|  |  |  |
| --- | --- | --- |
|  | BAN CƠ YẾU CHÍNH PHỦ  “HỌC VIỆN KỸ THUẬT MẬT MÃ” | Mẫu 2 |

BÁO CÁO CHUYÊN ĐỀ SỐ 4.2.1

“Phương pháp trích xuất đặc trưng tĩnh”

NHIỆM VỤ: “Nghiên cứu và ứng dụng nền tảng học sâu để xây dựng hệ thống phát hiện mã độc trực tuyến”.

Mã số: 06/2022/CB.

Cơ quan chủ trì: Học viện Kỹ thuật Mật mã

Chủ nhiệm: ThS. Lê Đức Thuận

Hà Nội - 2023

|  |  |  |
| --- | --- | --- |
|  | BAN CƠ YẾU CHÍNH PHỦ  “HỌC VIỆN KỸ THUẬT MẬT MÃ” |  |

BÁO CÁO CHUYÊN ĐỀ SỐ 4.2.1

“Phương pháp trích xuất đặc trưng tĩnh”

NHIỆM VỤ: “Nghiên cứu và ứng dụng nền tảng học sâu để xây dựng hệ thống phát hiện mã độc trực tuyến”.

Mã số: 06/2022/CB.

Cơ quan chủ trì: Học viện Kỹ thuật Mật mã

Chủ nhiệm: ThS. Lê Đức Thuận

|  |  |
| --- | --- |
| **Người thực hiện chuyên đề** | **Cơ quan chủ trì** |
| *(Họ tên và chữ ký)* | *(Họ tên và chữ ký)* |

Hà Nội - 2023

MỤC LỤC

[MỤC LỤC 1](#_Toc129697622)

[DANH MỤC HÌNH ẢNH 2](#_Toc129697623)

[PHÂN TÍCH, TRÍCH XUẤT ĐẶC TRƯNG TĨNH MÃ ĐỘC ANDROID 3](#_Toc129697624)

[1.1. Các thành phần trong ứng dụng android 3](#_Toc129697625)

[1.2. Phân tích gom nhóm họ mã độc của các tập tin 3](#_Toc129697626)

[1.3. Trích xuất đặc trưng bằng phương pháp tĩnh 7](#_Toc129697627)

[1.4. Kết luận 16](#_Toc129697628)

[TÀI LIỆU THAM KHẢO 17](#_Toc129697629)

# DANH MỤC HÌNH ẢNH

[Hình 1. Khởi tạo môt trường cho andropytool 5](#_Toc129697604)

[Hình 2. Truy cập vào thư mục andropytool 5](#_Toc129697605)

[Hình 3. Tiến hành phân tích với cơ sở dữ liệu của totalvirus 6](#_Toc129697606)

[Hình 4. Truy cập vào tập tin av\_class 6](#_Toc129697607)

[Hình 5. Lấy dữ liệu phân tích về 7](#_Toc129697608)

[Hình 6. Họ virus được nhận diện với totalvirus 7](#_Toc129697609)

[Hình 7. Các thư mục được tạo sau khi chạy phân tích tĩnh 9](#_Toc129697610)

[Hình 8. Cấu trúc tập tin được trích xuất trong Features\_files 10](#_Toc129697611)

[Hình 9. Packege name trong phân tích tĩnh 10](#_Toc129697612)

[Hình 10. Permission trong phân tích tĩnh 10](#_Toc129697613)

[Hình 11. Main Activity trong phân tích tĩnh 11](#_Toc129697614)

[Hình 12. API làAPI calls trong phân tích tĩnh 11](#_Toc129697615)

[Hình 13. Strings trong phân tích tĩnh 12](#_Toc129697616)

[Hình 14. API Packages trong phân tích tĩnh 12](#_Toc129697617)

[Hình 15. Intents trong phân tích tĩnh 12](#_Toc129697618)

[Hình 16. Activities trong phân tích tĩnh 13](#_Toc129697619)

[Hình 17. Services trong phân tích tĩnh 13](#_Toc129697620)

[Hình 18. FlowDroid trong phân tích tĩnh 14](#_Toc129697621)

# PHÂN TÍCH, TRÍCH XUẤT ĐẶC TRƯNG TĨNH MÃ ĐỘC ANDROID

## Các thành phần trong ứng dụng android

Các tập tin trên nền tảng Android có phần mở rộng là APK được phân tích để trích xuất và lựa chọn các đặc trưng. Tập tin có phần mở rộng APK là định dạng sử dụng cho việc phân phối và cài đặt các phần mềm ứng dụng điện thoại có hệ điều hành Android. Bản chất của tập tin apk là một tệp tin nén với cấu trúc như sau:

* META-INIF/: chứa thông tin mô tả từ tập tin jar của Java
* res/: chứa các tài liệu nguồn
* libs/: chứa các thư viện .so từ phần mềm phát triển.
* androidManifest.xml: chứa các tập tin cấu hình về quyền truy cập và dịch vụ.
* classes.dex: chứa mã byte-code.
* resources.asc: chứa mã nguồn nhị phân sau khi biên dịch.

## Phân tích gom nhóm họ mã độc của các tập tin

Andropytool là một công cụ để trích xuất cả hai tính năng tĩnh và động từ các ứng dụng Android [12]. Androidpytool giúp cho các nhà phân tích mã độc trên Android trích xuất được các đặc trưng để có thể mô hình hóa hành vi của các mẫu mã độc, từ đó có thể phân biệt được các loại mã độc khác nhau và mã sạch.

Andropytool tích hợp các công cụ nổi tiếng như AndroGuard, Droidbox, FlowDroid, AVClass, VirusTotal, Strace. Do đó Androidpytool cho phép người phân tích thu được nhiều đặc trưng khác nhau như: lời gọi hàm (API) , quyền được khai báo, quyền yêu cầu sử dụng (permission), nhãn (các họ mã độc) thu được từ các công cụ chống mã độc khác được tích hợp trong Virus Total, source-Sinks, các lời gọi API được gọi trong thời gian thực , các hoạt động truy cập, các hoạt động của người dùng v.v.

Andropytool tích hợp các công cụ phân tích khác nhau và các công cụ xử lý ứng dụng Android, từ đó cung cấp các đặc trưng liên quan tới các hành vi và đặc điểm riêng của từng mã độc.

* **Các bước để phân tích gom nhóm họ mã độc**

**B1: Lọc các tập tin định dạng .APK**. Việc lọc này dùng để kiểm tra xem tập tin APK được đưa vào liệu có phải là ứng dụng Android hợp lệ hay không.

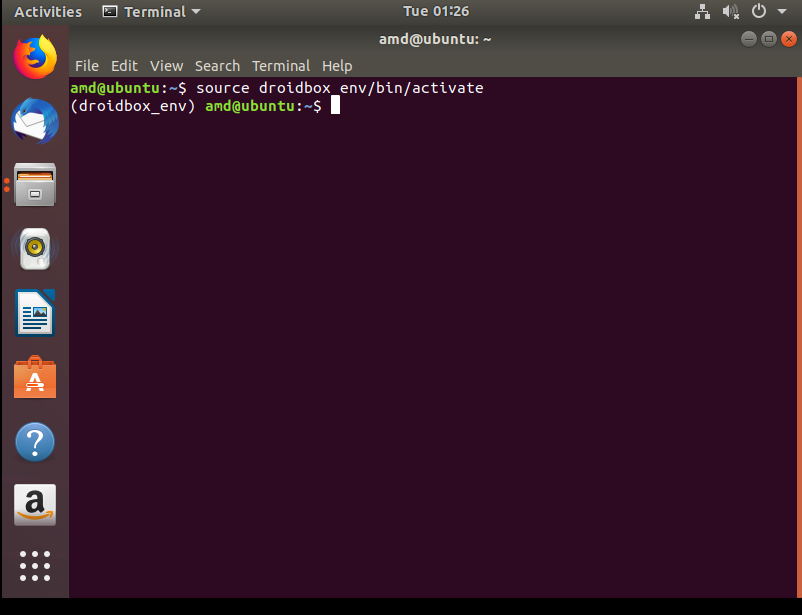
**B2: Phân tích VirusTotal.** Với mỗi tập tin được đưa vào VirusTotal sau khi được phân tích sẽ trả về kết quả từ hơn 60 công cụ phát hiện mã độc (Antivirus) khác nhau.

**B3: Phân đoạn dữ liệu.** Phân loại dựa trên báo cáo VirusTotal được sử dụng để chia bộ mẫu thành hai bộ: mẫu mã độc (có ít nhất một công cụ Antivirus cho kết quả tập tin đưa vào là mã độc) và mẫu mã sạch.

**B4: Thực thi FlowDroid.** Trong bước này, công cụ phân tích Android DroidBox được thực hiện cho từng mẫu.

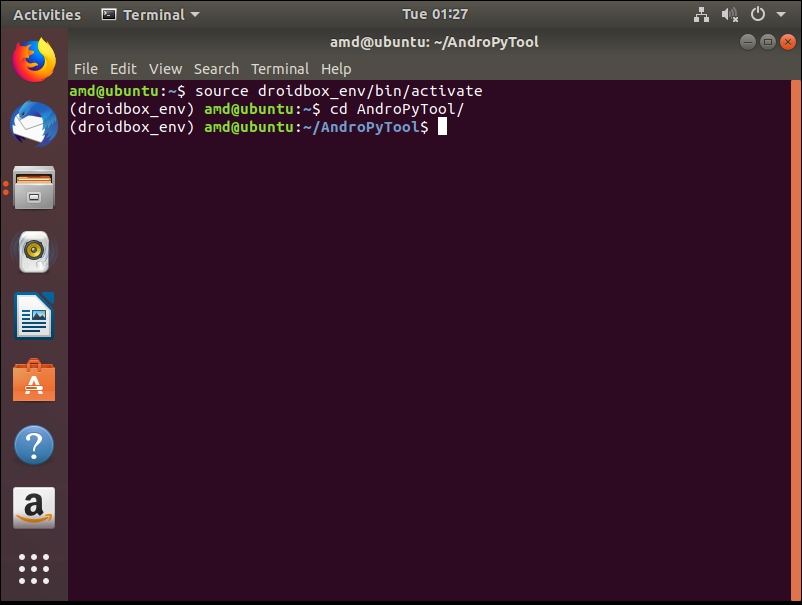
**B5: Trích xuất thông tin.** Tất cả các thông tin liên quan tới tập tin Android sẽ được trích xuất. Các thông tin này có thể được lưu thành 03 định dạng: .CSV, JSON và Mongo DB. Mỗi tập tin định dạng này sẽ lưu thông tin của một tập tin APK.

Dựa vào các bước phân tích như trên, sử dụng ubuntu để chạy thực nghiệm việc gom nhóm họ mã độc bằng TotalVirus được tích hợp trong Androidpytool như sau:



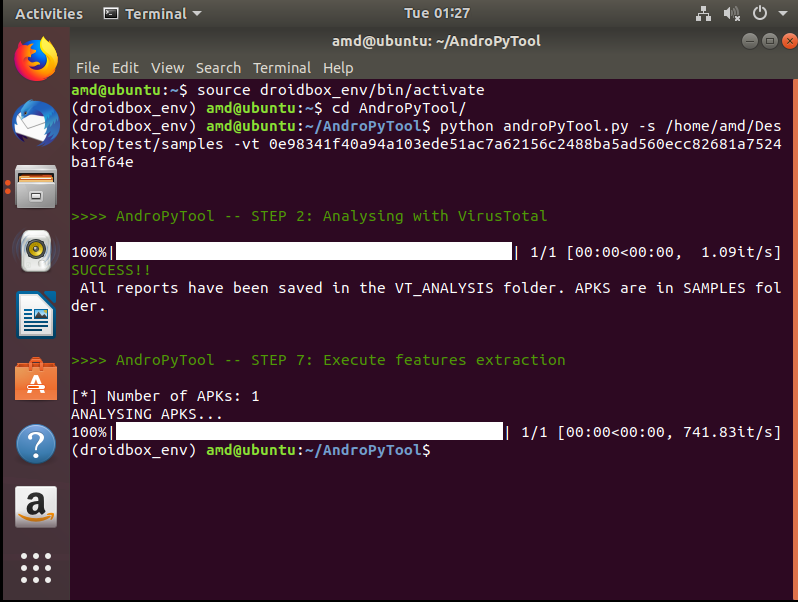
1. Khởi tạo môt trường cho andropytool

Bước khởi tạo này chính là bước kích hoạt môi trường. Để có thể vào được chương trình Andropytool.



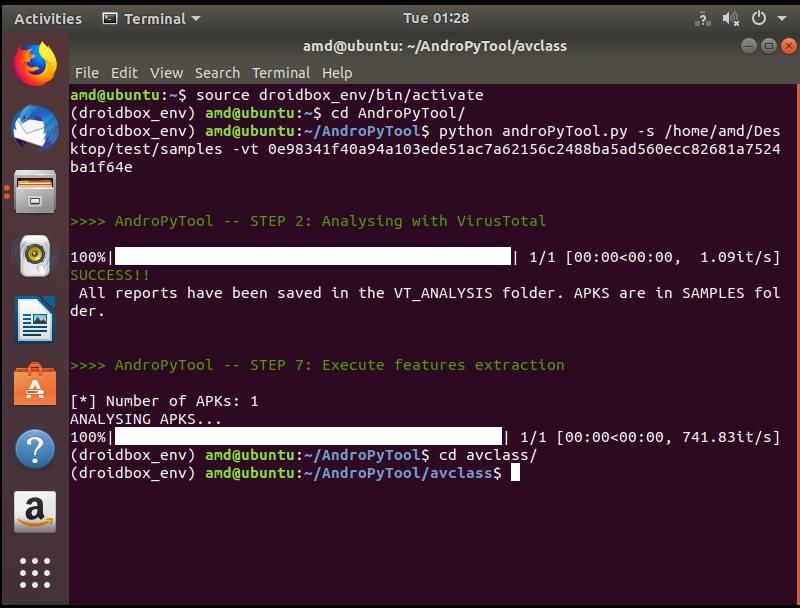
1. Truy cập vào thư mục andropytool

Trong hình 2, sử dụng câu lệnh cd AndroPytool để vào thư mục AndroPytool. Từ đó tiến hành phân tích một tập tin apk như trong hình 3.



1. Tiến hành phân tích với cơ sở dữ liệu của totalvirus

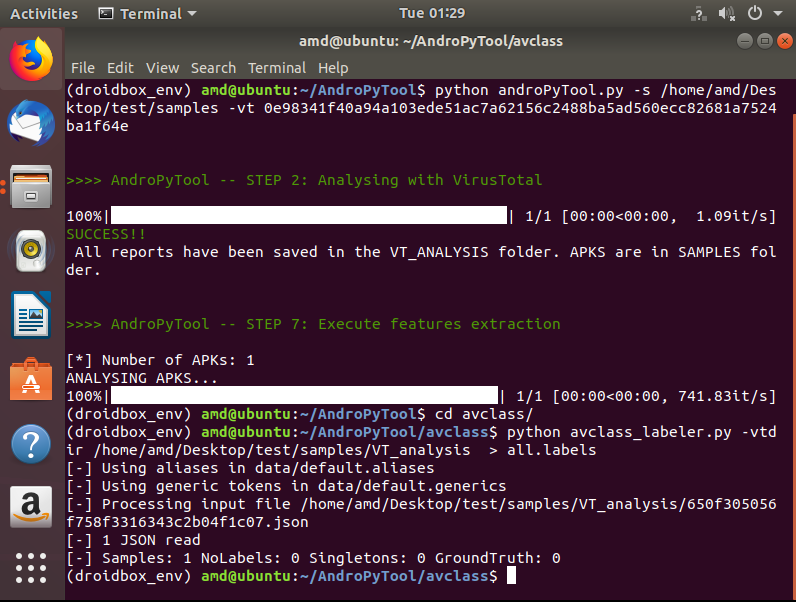
Trong quá trình phân tích tập tin, AndroPytool sẽ phân tích và gán tập tin này vào một nhãn. Giá trị nhãn này được gán dựa vào av\_class được truy cập vào thư mục như hình 4.



1. Truy cập vào tập tin av\_class

Sau khi truy cập vào avclass, bắt đầu chạy lệnh sau như hình 5:

Python avclass\_labeler.py -vtd ir /home/amd/Desktop/test/samples/VT\_analysis > all.labels



1. Lấy dữ liệu phân tích về

Sau khi phân tích xong tập tin, kết quả được lưu vào tập tin all.labels, trong đó gồm 2 thành phần: tên tập tin và nhãn được gán.



1. Họ virus được nhận diện với totalvirus

Quá trình gán nhãn của tập tin này được dùng kết hợp với các đặc trưng của tập tin. Từ đó làm đầu vào trong quá trình huấn luyện mô hình sử dụng học máy (được nêu trong chuyên đề 2) và so sánh với các nhãn được chẩn đoán trong giai đoạn kiểm tra.

Trong mục 1.3 và 1.4 tiếp theo, sẽ mô tả cách thức trích xuất đặc trưng tĩnh và động và nêu ý nghĩa của các đặc trưng đó.

## Trích xuất đặc trưng bằng phương pháp tĩnh

Trích xuất đặc trưng bằng phương pháp tĩnh là quá trình phân tích mã nguồn của chương trình ứng dụng, điều này có nghĩa rằng ứng dụng không được thực thi. Phân tích tĩnh có thể tìm các điểm yếu trong mã nguồn dẫn đến lỗ hổng bảo mật. Tuy nhiên để có thể đánh giá được điểm yếu trong mã nguồn sẽ cần phải tích hợp thêm các công cụ khác.

Android ra mắt năm 2008 với phiên bản đầu tiên là Android 1.0 và cho đến nay Android đã có phiên bản 10.0 với tên gọi Android Pie. Cùng với sự phát triển đó, Android đang là hệ điều hành mã nguồn mở cho di động phát triển mạnh mẽ nhất với hàng triệu ứng dụng nên sẽ là mục tiêu hàng đầu cho các hacker lợi dụng để lấy thông tin, điều khiển hệ thống, phá hoại, v.v.

Như đã biết, ứng dụng trên android khi được xây dựng để sẵn sàng cài đặt đã được mã hóa thành mã máy có tên Dalvik code, trong khi đó đặc trưng của phân tích tĩnh là phân tích mã nguồn của ứng dụng. Do đó, việc sử dụng phân tích tĩnh vào phân tích mã độc android phải qua một quá trình dịch ngược để có thể tiến hành phân tích. Để quá trình dịch ngược diễn ra một cách đơn giản nhất, người lập trình có thể dùng một số công cụ như: baksmali, Apktool, Dex2jar, Dexjump,… hoặc một số công cụ do những nhóm nghiên cứu mã độc trên android đưa ra. Trong phần này, công cụ được dùng để phân tích tĩnh là AndroPytool, công cụ này được tích hợp 2 công cụ khác bên trong là: FlowDroid và Androguard.

**FlowDroid** [9] là hệ thống phân tích tĩnh đầu tiên mà có thể phân tích đầy đủ context, flow, field và object-sensitive trong mô hình hoàn chỉnh của Android, bao gồm xử lý các các lời gọi, do nhà phát triển Widget UI xác định trong ứng dụng. FlowDroid tối đa hóa độ chính xác và thu hồi thông tin, mục đích giảm thiểu thông tin mất mát dẫn đến cảnh báo sai. Để có được hiệu suất ở mức ổn định thì FlowDroid sử dụng phân tích bí danh on-demand. Việc phân tích này được đề xuất bởi Andromeda[10] nhưng cải thiện hơn về độ chính xác. Năm 2013 FlowDroid đã có mã nguồn mở (open source) và đã được các nhà phát triển công cụ Antivirus lựa chọn sử dụng.

**Androguard** là một công cụ đươc viết bằng hoàn toàn bằng Python để có thể xủ lý được các tập tin Andoid:

* DEX, ODEX
* APK
* Android’s binary xml
* Android resource
* Dissemble DEX/ODEX bytecodes
* Decompiler for DEX/ODEX files
* **Quá trình thực hiện phân tích**

Trong trích xuất bằng phương pháp tĩnh và phương pháp động (sẽ được nhắc tới ở phần tiếp theo) được trình bày sau đây sẽ được chạy trên hệ điều hành Ubuntu phiên bản 18.04 sau khi đã cài đặt AndroPyTool [2]. Việc thực hiện phân tích và trích xuất được chạy trên trình terminal. Phân tích bao gồm các bước:

* *Bước 1:* Kích hoạt môi trường để phân tích và trích xuất.
* *Bước 2:* Di chuyển đến thư mục AndroPyTool,
* *Bước 3:* Thực hiện câu lệnh để phân tích tĩnh và trích xuất các đặc trưng với cấu trúc lệnh như sau: *python androPyTool.py –s <PATH\_FOLDER\_APK> -fw*

Sau khi thực hiện phân tích tĩnh sẽ có các thư mực chứa kết quả phân tích và trích xuất được các đặc trưng của tập tin *.apk*

|  |
| --- |
|  |
| 1. Các thư mục được tạo sau khi chạy phân tích tĩnh |

Trong đó:

* Thư mục **Features\_file** chức các file dạng mở rộng .json chứa kết quả phân tích tĩnh sử dụng Androguard và kết quả của Flowdroid.

|  |
| --- |
|  |
| 1. Cấu trúc tập tin được trích xuất trong Features\_files |

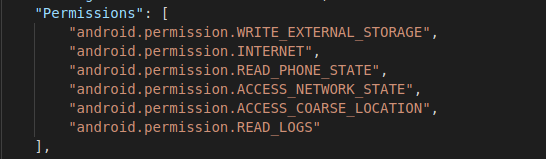
Đối với việc chọn các đặc trưng trong phân tích tĩnh sẽ được tập trung vào các tập tin trong thư mục **Features\_files** và đặc biệt là object “**Static\_analysis**”

Cấu trúc của object “**Static\_analysis**” gồm:

* **Package name**

1. Packege name trong phân tích tĩnh

Mọi ứng dụng trên Android đều có một số định danh (ID) để phân biệt ứng dụng trên thiết bị và trên Google Play Store và khi ứng dụng được phát hành công khai thì tên này không thể thay đổi được. Hay nói một cách khác thì giá trị của *Package name* chính là chữ ký của ứng dụng từ nhà phát hành và nó là duy nhất [3].

* **Permission**

1. Permission trong phân tích tĩnh

Android triển khai Permission System buộc các nhà phát triển phải khai báo tài nguyên quan trọng về bảo mật ứng dụng có thể truy cập và các hoạt động quan trọng bảo mật mà ứng dụng có thể thực hiện [4]. Sau đó, tại thời điểm chạy, hệ điều hành chặn mọi truy cập không được khai báo hoặc cố gắng hoạt động bất hợp pháp. Trong thực tế, trước khi ứng dụng được cài đặt, các Permission cần được sử dụng trong ứng dụng sẽ hiển thị cho người dùng dưới dạng danh sách. Tuy nhiên, người dùng đơn thuần không đủ chuyên môn để có thể biết liệu một ứng dụng có độc hại hay không bằng cách chỉ đọc permission. Hơn nữa một số lớn người dùng thậm chí không đọc danh sách permission mà chỉ cần cài đặt. Trong trường hợp này, Permission System không giúp người dùng bảo vệ họ khỏi ứng dụng độc hại. Chính vì vậy việc trích xuất permission trong một ứng dụng android để lấy làm đặc trưng là một điều quan trọng.

* **Main activity**

|  |
| --- |
|  |
| 1. Main Activity trong phân tích tĩnh |

Là activity ban đầu được chỉ định khi bắt đầu chạy ứng dụng.

* **API calls**

|  |
| --- |
|  |
| 1. API làAPI calls trong phân tích tĩnh |

API calls được các nhà phát triển sử dụng để tương tác với hệ điều hành Android [5]. Do có nhiều API, vì vậy nên xem xét các API cụ thể thường được sử dụng dụng bởi phân mềm độc hại và cho phép chúng truy cập tài nguyên cần được bảo vệ trong điện thoại. Và có một số API đáng ngờ hay được sử dụng trong các phần mềm mã độc như: getSubscackerId(), getDeviceId(), getDescription(),v.v. Qua đó việc phân tích và lấy các API Calls có thể xác định chắc chắn hơn về ứng dụng này có mang mã độc hay không.

* **Strings**

|  |
| --- |
|  |
| 1. Strings trong phân tích tĩnh |

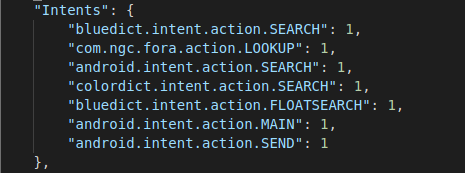
Strings là một trong những đối tượng quan trọng để có thể nhận thấy rằng đây có phải một ứng dụng độc hại hay không [6]. Các ứng dụng Android phụ thuộc rất nhiều vào *strings* để thực hiện hành vi nhạy cảm hoặc độc hại chẳng hạn như đọc thông tin liên lạc, ID thiết bị, truy cập bộ nhớ, gọi API ẩn và sử dụng máy chủ từ xa. Hơn nữa phân tích chuỗi bổ sung cho các đối tượng phân tích khác, và góp phần cải thiện độ chính xác của họ mã độc. Đặc biệt là chuỗi URL để mã độc gửi thông tin về máy chủ thông qua URL đó.

* **API Packages**

Các API Packages được sử dụng trong ứng dụng, để biết được những API thuộc Packages nào và số lần được sử dụng.

|  |
| --- |
| 1. API Packages trong phân tích tĩnh |

* **Intents**



1. Intents trong phân tích tĩnh

Intent là một hệ thống nhắn tin phức tạp trong nền tảng Andoird, được coi là một cơ chế bảo mật để cản trở các ứng dụng từ việc truy cập và các ứng dụng khác. Các ứng dụng phải có quyền cụ thể để sử dụng intent. Đây là một cách kiểm soát những gì mà ứng dụng có thể làm trong Android. Intent được khai báo trong tập tin AndroidManifest.xml

* **Activities**

|  |
| --- |
| 1. Activities trong phân tích tĩnh |

Các *Activity* có trong ứng dụng Android, bên trong mỗi activity là các intent được sử dụng trong activity đó.

* **Services**

|  |
| --- |
| 1. Services trong phân tích tĩnh |

Services là một thành phần của ứng dụng có khả năng thực hiện các thao tác chạy kéo dài trong nền của ứng dụng và không cung cấp giao diện người dùng. Một thành phần ứng dụng khác có thể bắt đầu một dịch vụ và nói sẽ tiếp tục chạy ngầm ngay cả khi người dùng chuyển sang một ứng dùng khác [7]. Ngoài ra, một thành phần có thể gắn kết với một dịch vụ để tương tác với nó và thậm chí thực hiện truyền thông liên tiến trình (IPC).

* **FlowDroid**
* Thư mục **FlowDroid\_Ouputs** và **FlowDroid\_prossesed** lần lượt chứa kết quả của quá trình FlowDroid dưới dạng .json và kết quả của FlowDroid được chuyển từ dạng .json sang .csv
* Thư mục **Samples** chứa các files .apk được chạy để phân tích
* Thư **Select\_Features\_Static** chứa các đặc trưng được chọn của .apk để làm đặc trưng cho file đó (sẽ được trình bày trong phần lựa chọn đặc trưng)

|  |
| --- |
| 1. FlowDroid trong phân tích tĩnh |

Dangerous permissions thường gặp được biểu diễn trong bảng sau:

|  |  |  |
| --- | --- | --- |
| Permission group | Permission | Funcion |
| CALENDAR | READ\_CALENDAR | Used for runtime permissions related to user’s calendar |
| WRITE\_CALENDAR |
| CAMERA | CAMERA | Used for permissions that are associated with accessing camera or capturing images/video from the device |
| CONTACTS | READ\_CONTACTS | Used for runtime permissions related to contacts and profiles on this device. |
| WRITE\_CONTACTS |
| GET\_ACCOUNTS |
| LOCATION | ACCESS\_FINE\_LOCATION | Used for permissions that allow accessing the device location |
| ACCESS\_COARSE\_LOCATION |
| MICROPHONE | RECORD\_AUDIO | Used for permissions that are associated with accessing microphone audio from the device. |
| PHONE | READ\_PHONE\_STATE | Used for permissions that are associated with accessing body or environmental sensors |
| CALL\_PHONE |
| READ\_CALL\_LOG |
| WRITE\_CALL\_LOG |
| ADD\_VOICEMAIL |
| USE\_SIP |
| PROCESS\_OUTGOINGS\_CALLS |
| SENSORS | BODY\_SENSORS | Used for permissions that ar associated with accessing body or environmental sensors. |
| SMS | SEND\_SMS | Used for runtime permissions related to user’s SMS messages |
| RECEIVE\_SMS |
| READ\_SMS |
| STORAGE | RECEIVE\_WAP\_PUSH | Used for runtime permissions related to the shared external storage. |
| READ\_EXTERNAL\_STORAGE |
| WRITE\_EXTERNAL\_STORAGE |

Một số API package thường gặp trong các dataset của bài toán phát hiện mã độc trên android được mô tả trong bảng:

|  |  |
| --- | --- |
| API package | API package |
| java.lang.StringBuilder.toString | android.content.Context.getSystemService |
| java.lang.System | android.content.Context.startActivity |
| java.lang.Integer | java.lang.Thread |
| java.lang.String.substring | java.lang.Boolean |
| java.io | android.app |
| android.content.SharedPreferences | java.lang.String |
| android.content.Context.getPackageName | java.lang.StringBuilder |
| java.util | java.lang.Long |
| java.lang.String.length | android.view |
| android.content.Intent.putExtra | java.lang.Exception |

Một số API call khả nghi (suspicious) thường được sử dụng trong mã độc [75] được mô tả trong bảng:

|  |  |
| --- | --- |
| Group API | API suspicious |
| Privacy | getSimSerialNumber(), getSubricbierId(), getImei(), getDeviceId(), detLineNumber(), getNetworkOperator(), newOutgoingCalls() |
| SMS | sendTextMessages(), sendBroadcast(), sendDataMessage(), telephonySMS-Received(), content:sms() |
| Network | Httpclient.excute(), getOutputStream(), getInputStream(), getNetworkInfi(), httpUrlConnectionConn(), execHttpRequest(), SendRequestMethod(), setDataAndTypes() |
| Location | getLongitiude(), getLatitute(), getCellLocation(), requestLocationUpdate(), getFromLocation(), getLastLocationKnown() |
| Application | getInstallPackades(), installPackadges() |
| Component | startService(), setComponent() |
| Wifi | getWifiState(), getEnabled(), getWifiEnables() |
| File | URL(), getAssets(), OpenFileOutPut(), Browser:Bookmarks\_URL() |
| Function | Runtime.exec() |
| Obfuscation | DexClassLoder(), Cipher.getInstence() |

## Kết luận

Phương pháp tĩnh giúp cho quá trình phân tích nhanh, đưa ra được kết quả nhanh chóng. Trong phương pháp này giúp cho người dùng lấy được nhiều thông tin quan trọng như permission api, intent, service, v.v. Những đặc trưng này rất hữu dụng khi đưa vào các mô hình học máy, học sâu và cho kết quả phát hiện, phân lớp rất khả quan.

# TÀI LIỆU THAM KHẢO

1. Omer Tripp, Marco Pistoia, Patrick Cousot, Radhia Cousot, Salvatore Guarnieri, "An droneda: Accurate and scalable security analysis of web applications, Fundamental approaches to software engineering," in Fundamental approaches to software engineering. 16th international conference, FASE 2013, held as part of the European joint conferences on theory and practice of software, ETAPS 2013, Rome, Italy, Proceedings (pp. 220 - 225), March 2013.
2. H. Alatwi, “Android malware detection using category-based machine learning classiﬁers”, Rochester institute of technology, Masters thesis, June 2016.
3. "DeverloperAndroid," [Online]. Available: <https://developer.android.com/studio/build/application-id>.
4. Pham Thanh Giang, Nguyen Viet Duc, Pham Minh Vi, "Permission Analysis for Android Malware Detection, The Proceedings," in The Proceedings of the 7th VAST-AIST workshop “Research collaboration: review and prespective”, 11/2015.
5. Altyeb Altaher, Omar Mohammed Barukab, "Intelligent Hybrid Approach for Androi Malware Detection based on Permission and API Calls," in International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 8, No. 6, 2017.
6. Justin Del Vecchio, Feng Shen, Kenny M. Yee, Boyu Wang, Steven Y.Ko, Lukasz Ziarek, in String Analysis of Android Applications- AndroDialysis: Analysis of Android Intent Effectiveness in Malware Dectection- Ali Feizollahm, Nor Badrul Anuar, 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE).
7. "Securitybox" [Online]. Available: <https://securitybox.vn/2337/cac-ky-thuat-phan-tich-dong-muc-co-ban-phan-1/>.
8. "GitHub" [Online]. Available: <https://github.com/pjlantz/droidbox>.
9. D.-J. Wu, C.-H. Mao and T.-E. Wei, "Droid mat: Android malware detection through manifest and API calls tracing," in Seventh Asia Joint Conference on Information Security, pp. 62 - 69, 2012.
10. "GitHub" [Online]. Available: <https://github.com/idanr1986/droidmon>.
11. Nguyễn Thị Biên, “Khai phá luật kết hợp trong cơ sở dữ liệu đa phương tiện”, luân văn thạc sĩ ngành công nghệ phần mềm, Đại học công nghệ, 2012.
12. Chenglin Li, Keith Mills, Rui Zhu, Di Niu, Hongwen Zhang, Husam Kinawi, Android Malware Detection based on Factorization Machine, arXiv:1805.11843.
13. Daniel Arp, Michael Spreitzenbarth, Malte Huebner, Hugo Gascon, and Konrad Rieck ["Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket"](https://www.tu-braunschweig.de/Medien-DB/sec/pubs/2014-ndss.pdf), 21th Annual Network and Distributed System Security Symposium (NDSS), February 2014