|  |
| --- |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| CHUYÊN ĐỀ AN TOÀN HỆ THỐNG THÔNG TIN  **BÁO CÁO ĐỀ TÀI**  **TÌM HIỂU CÁC VẤN ĐỀ MÃ ĐỘC TRÊN THIẾT BỊ ANDROID** |
| Chuyên ngành: An toàn thông tin  *Sinh viên thực hiện*:  **Nguyễn Văn Hùng – AT180620**  *Giảng viên hướng dẫn:*  **ThS. Hoàng Thanh Nam**  Khoa ATTT – Học viện KTMM |
| Hà Nội, 2024 |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| CHUYÊN ĐỀ AN TOÀN HỆ THỐNG THÔNG TIN  **BÁO CÁO ĐỀ TÀI**  **TÌM HIỂU CÁC VẤN ĐỀ MÃ ĐỘC TRÊN THIẾT BỊ ANDROID** |
| Chuyên ngành: An toàn thông tin  *Sinh viên thực hiện*:  **Nguyễn Văn Hùng – AT180620**    *Giảng viên hướng dẫn:*  **ThS. Hoàng Thanh Nam**  Khoa ATTT – Học viện KTMM |
| Hà Nội, 2024 |

**HỌC VIỆN KỸ THUẬT MẬT MÃ**

KHOA AN TOÀN THÔNG TIN

BÁO CÁO CHUYÊN ĐỀ AN TOÀN HỆ THỐNG THÔNG TIN

**ĐỀ TÀI:**

**TÌM HIỂU VỀ CÁC VẤN ĐỀ MÃ ĐỘC TRÊN THIẾT BỊ ANDROID**

Nhận xét của cán bộ hướng dẫn:..............................................................................

..................................................................................................................................

..................................................................................................................................

..................................................................................................................................

..................................................................................................................................

..................................................................................................................................

..................................................................................................................................

..................................................................................................................................

..................................................................................................................................

Điểm chuyên cần:.....................................................................................................

Điểm báo cáo:...........................................................................................................

**Xác nhận của cán bộ hướng dẫn**

# **MỤC LỤC**

[**DANH MỤC TỪ VIẾT TẮT** 4](#_Toc186267382)

[**DANH MỤC BẢNG** 5](#_Toc186267383)

[**LỜI CẢM ƠN** 7](#_Toc186267384)

[**LỜI NÓI ĐẦU** 8](#_Toc186267385)

[**CHƯƠNG 1. GIỚI THIỆU CHUNG VẤN ĐỀ** 9](#_Toc186267386)

[**1.1. Mã độc.** 9](#_Toc186267387)

[**1.1.1. Khái niệm mã độc.** 9](#_Toc186267388)

[**1.1.2. Phân loại theo NIST.** 9](#_Toc186267389)

[**1.1.3 Thực trạng mã độc hiện nay.** 12](#_Toc186267390)

[**1.2. Tổng quan về Android.** 13](#_Toc186267391)

[**1.2.1 Kiến trúc về hệ điều hành Android.** 13](#_Toc186267392)

[**1.2.2. Các thành phần chính của ứng dụng Android.** 14](#_Toc186267393)

[**1.2.3. Lựa chọn phương pháp trên Androi.** 15](#_Toc186267394)

[**1.2.4 Khai thác tính năng.** 16](#_Toc186267395)

[**CHƯƠNG 2. TỔNG QUÁT HÓA VỀ MÃ ĐỘC ANDROID** 17](#_Toc186267396)

[**2.1. Sự phát triển mã độc.** 17](#_Toc186267397)

[**2.2. Các họ mã độc** 19](#_Toc186267398)

[**2.3. Mã độc tấn công và lẩn tránh** 21](#_Toc186267399)

[**CHƯƠNG 3. MỘT SỐ PHƯƠNG PHÁP PHÂN LOẠI MÃ ĐỘC ANDROID** 24](#_Toc186267400)

[**3.1. Phát hiện dựa trên chữ kí (Signature base).** 24](#_Toc186267401)

[**3.2. Phát triển dựa trên hành vi (Behavior based).** 24](#_Toc186267402)

[**3.3. Phát triển dựa trên học máy (Machine learning).** 25](#_Toc186267403)

[**CHƯƠNG 4. PHÂN TÍCH THUẬT TOÁN** 30](#_Toc186267404)

[**4.1. Thuật toán Neural Network(NN).** 30](#_Toc186267405)

[**4.2. Thuật toán Support Vector Classifier(SVC).** 31](#_Toc186267406)

[**4.3. Thuật toán thuật toán di truyền(genetic algorithm-GA).** 32](#_Toc186267407)

[**CHƯƠNG 5. THỰC NGHIỆM** 33](#_Toc186267408)

[**5.1. Tổng quan.** 33](#_Toc186267409)

[**5.2. Phân tích thiết kế hệ thống** 34](#_Toc186267410)

[**5.3. Cấu trúc** 36](#_Toc186267411)

[**KẾT LUẬN** 44](#_Toc186267412)

[**Tài liệu tham khảo** 45](#_Toc186267413)

[**Phụ lục** 46](#_Toc186267414)

# **DANH MỤC TỪ VIẾT TẮT**

|  |  |
| --- | --- |
| SVC | Support Vector Classifirer |
| NIST | National Institue of Standards and Technology |
| NN | Neural Network |
| GA | Genetic Algorithm |
| HDH | Hệ điều hành |
| APK | Android Package |

# **DANH MỤC BẢNG**

[Bảng các họ mã độc phổ biến trên hệ thống Android. 19](#_Toc186197563)

[Bảng điểm yếu điểm mạnh của các phương pháp phát hiện mã độc trên Android. 28](#_Toc186197564)

[Bảng mô tả chi tiết các Usecase của hệ thống. 34](#_Toc186197565)

**DANH MỤC HÌNH VẼ**

[Hình 1. Kiến trúc hệ điều hành android 14](#_Toc186196858)

[Hình 2. Qúa trình lựa chọn tính năng 15](#_Toc186196859)

[Hình 3. Tóm tắt về trính xuất đặc trưng trên Android 16](#_Toc186196860)

[Hình 4. Thị phần toàn cầu 18](#_Toc186196861)

[Hình 5. Phát triển các kĩ thuật lẩn tránh qua nhiều nghiên cứu 22](#_Toc186196862)

[Hình 6. Phân loại phân tích bảo mật phần mềm độc hại Android 24](#_Toc186196863)

[Hình 7. Phân loại các phương pháp học máy 26](#_Toc186196864)

[Hình 8. Giao diện cơ bản 33](#_Toc186196865)

[Hình 9. Mô hình usecase 34](#_Toc186196866)

[Hình 10. Một ví dụ về thông tin đầu vào của một tệp apk 38](#_Toc186196867)

# **LỜI CẢM ƠN**

Em xin chân thành gửi lời cảm ơn đến thầy Hoàng Thanh Nam đã tạo điều kiện, cung cấp kiến thức chuyên sâu, trong thời gian thực hiện đề tài đã tận tình hướng dẫn, giúp đỡ, chỉ bảo. Đồng thời, em xin gửi lời cảm ơn đến các thầy cô trong Học viện Kỹ thuật mật mã đã truyền đạt cho chúng em những kiến thức từ cơ bản đến nâng cao suốt những năm học qua, giúp em có được những kiến thức cơ bản để có thể hoàn thành được đề tài này.

Trong quá trình nghiên cứu và thực hiện đề tài, em đã rất cố gắng để hoàn thành đề tài tốt nhất có thể, nhưng do chưa có nhiều kinh nghiệm thực tế nên còn nhiều thiếu sót trong dự án. Vậy nên chắc chắn sẽ không thể tránh khỏi những thiếu sót khi thực hiện đề tài, chúng em rất mong nhận được sự góp ý từ thầy để có thể hoàn thiện tốt hơn đề tài của mình.

Em xin chân thành cảm ơn!

# **LỜI NÓI ĐẦU**

Ngày nay, hệ điều hành Android trở nên phổ biến hơn bao giờ hết và là nền tảng chính cho hàng tỷ thiết bị di động trên toàn thế giới. Với sự gia tăng về số lượng và mức độ phức tạp của các ứng dụng, Android không chỉ mang lại tiện ích vượt trội mà còn trở thành mục tiêu hấp dẫn cho các cuộc tấn công mạng, đặc biệt là các loại mã độc nhằm khai thác dữ liệu cá nhân và tài nguyên thiết bị.

Các cuộc tấn công bằng mã độc trên Android có thể gây ra nhiều hậu quả nghiêm trọng, từ việc đánh cắp thông tin cá nhân đến việc kiểm soát thiết bị từ xa và chiếm đoạt các tài khoản tài chính. Điều này không chỉ đe dọa quyền riêng tư của người dùng mà còn làm suy yếu độ tin cậy của hệ điều hành Android. Các loại mã độc phổ biến trên Android như ransomware, trojan, spyware, và adware đã và đang là mối đe dọa lớn cho người dùng trên toàn cầu.

Trước tình hình đó, việc nghiên cứu và tìm hiểu về mã độc trên Android là điều cần thiết để người dùng có thể tự bảo vệ bản thân. Nội dung này sẽ cung cấp cái nhìn toàn diện về các loại mã độc thường gặp trên Android, cách thức chúng xâm nhập và gây hại, đồng thời đề xuất những biện pháp phòng chống hiệu quả nhằm tăng cường bảo mật cho thiết bị di động. Qua đó, người dùng sẽ có thể chủ động và an tâm hơn trong việc sử dụng thiết bị Android trong môi trường số hiện đại.

# **CHƯƠNG 1. GIỚI THIỆU CHUNG VẤN ĐỀ**

## **1.1. Mã độc.**

### **1.1.1. Khái niệm mã độc.**

Malware (Malicious software) hay còn gọi là mã độc (Malicious code) là tên gọi chung cho các phần mềm được thiết kế, lập trình đặc biệt để gây hại cho máy tính hoặc làm gián đoạn môi trường hoạt động mạng. Mã độc thâm nhập vào một hệ thống máy tính mà không có sự đồng ý của nạn nhân.

Mã độc hại còn được định nghĩa là “một chương trình (program) được chèn một cách bí mật vào hệ thống với mục đích làm tổn hại đến tính bí mật, tính toàn vẹn hoặc tính sẵn sàng của hệ thống”.

Nhiều người sử dụng máy tính vẫn thường dùng thuật ngữ Virus để chỉ chung cho các loại mã độc hại nhưng thực chất mã độc hại bao gồm nhiều loại khác nhau.

### **1.1.2. Phân loại theo NIST.**

Theo NIST (National Institue of Standards and Technology).

1. *Virus*

Với cách định nghĩa, phân loại này, Virus là một loại mã độc hại (Malicious code) có khả năng tự nhân bản và lây nhiễm chính nó vào các file, chương trình hoặc máy tính. Như vậy Virus máy tính phải luôn luôn bám vào một vật chủ (đó là file dữ liệu hoặc file ứng dụng) để lây lan.

Compiled Virus là Virus mà mã thực thi của nó đã được dịch hoàn chỉnh bởi một trình biên dịch để nó có thể thực thi trực tiếp từ hệ điều hành.

Interpreted Virus là một tổ hợp của mã nguồn mã chỉ thực thi được dưới sự hỗ trợ của một ứng dụng cụ thể hoặc một dịch vụ cụ thể trong hệ thống. Một cách đơn giản, Virus kiểu này chỉ là một tập lệnh, cho đến khi ứng dụng gọi thì nó mới được thực thi. Macro Virus, Scripting Virus là các Virus nằm trong dạng này. Macro Virus rất phổ biến trong các ứng dụng Microsoft Office khi tận dụng khả năng kiểm soát việc tạo và mở file để thực thi và lây nhiễm. Sự khác nhau giữa Macro Virus và Scripting Virus là: Macro Virus là tập lệnh thực thi bởi một ứng dụng cụ thể, còn Scripting Virus là tập lệnh chạy bằng một dịch vụ của hệ điều hành.

*b.Worm*

Worm cũng là một chương trình có khả năng tự nhân bản và tự lây nhiễm trong hệ thống tuy nhiên nó có khả năng “tự đóng gói”, điều đó có nghĩa là Worm không cần phải có “file chủ” để mang nó khi nhiễm vào hệ thống.

Tấn công của Worm có đặc trưng là lan rộng cực kỳ nhanh chóng do không cần tác dụng của con người (như khởi động máy, copy file hay đóng/mở file).

Worm có thể chia làm 2 loại:

* Network Service Worm lan truyền bằng cách lợi dụng các lỗ hổng bảo mật của mạng, của hệ điều hành hoặc của ứng dụng.
* Mass Mailing Worm là một dạng tấn công qua dịch vụ mail, tuy nhiên nó tự đóng gói để tấn công và lây nhiễm chứ không bám vào vật chủ là email.

*c.Trojan Horse*

Trojan Horse không tự nhân bản tuy nhiên nó lây vào hệ thống với biểu hiện rất ôn hoà nhưng thực chất bên trong có ẩn chữa các đoạn mã với mục đích gây hại. Trojan có thể lựa chọn một trong 3 phương thức để gây hại:

* Tiếp tục thực thi các chức năng của chương trình mà nó bám vào, bên cạnh đó thực thi các hoạt động gây hại một cách riêng biệt.
* Tiếp tục thực thi các chức năng của chương trình mà nó bám vào, nhưng sửa đổi một số chức năng để gây tổn hại hoặc che dấu các hành động phá hoại khác.
* Thực thi luôn một chương trình gây hại bằng cách núp dưới danh một chương trình không có hại.

*d.Malicious Mobile Code*

Là một dạng mã phần mềm có thể được gửi từ xa vào để chạy trên một hệ thống mà không cần đến lời gọi thực hiện của người dùng hệ thống đó. Malicious Mobile Code được coi là khác với Virus, Worm ở đặc tính là nó không nhiễm vào file và không tìm cách tự phát tán. Thay vì khai thác một điểm yếu bảo mật xác định nào đó, kiểu tấn công này thường tác động đến hệ thống bằng cách tận dụng các quyền ưu tiên ngầm định để chạy mã từ xa. Các công cụ lập trình như Java, ActiveX, JavaScript, VBScript là môi trường tốt cho Malicious Mobile Code.

*e.Tracking Cookie*

Là một dạng lạm dụng cookie để theo dõi một số hành động duyệt web của người sử dụng một cách bất hợp pháp. Cookie là một file dữ liệu chứa thông tin về việc sử dụng một trang web cụ thể nào đó của web-client. Mục tiêu của việc duy trì các cookie trong hệ thống máy tính nhằm căn cứ vào đó để tạo ra giao diện, hành vi của trang web sao cho thích hợp và tương ứng với từng web-client. Tuy nhiên tính năng này lại bị lạm dụng để tạo thành các phần mềm gián điệp (spyware) nhằm thu thập thông tin riêng tư về hành vi duyệt web của cá nhân.

*f.Phần mềm gián điệp (Spyware)*

Là loại phần mềm chuyên thu thập các thông tin từ các máy chủ (thông thường vì mục đích thương mại) qua mạng Internet mà không có sự nhận biết và cho phép của chủ máy. Một cách điển hình, Spyware được cài đặt một cách bí mật như là một bộ phận kèm theo của các phần mềm miễn phí (freeware) và phần mềm chia sẻ (shareware) mà người ta có thể tải về từ Internet. Một khi đã cài đặt, Spyware điều phối các hoạt động của máy chủ trên Internet và lặng lẽ chuyển các dữ liệu thông tin đến một máy khác (thường là của những hãng chuyên bán quảng cáo hoặc của các tin tặc). Phần mềm gián điệp cũng thu thập tin tức về địa chỉ thư điện tử và ngay cả mật khẩu cũng như là số thẻ tín dụng. Khác với Worm và Virus, Spyware không có khả năng tự nhân bản.

*g.Phần mềm quảng cáo (Adware)*

Phần mềm quảng cáo, rất hay có ở trong các chương trình cài đặt tải từ trên mạng. Một số phần mềm vô hại, nhưng một số có khả năng hiển thị thông tin lên màn hình, cưỡng chế người sử dụng.

*h.Attacker Tool*

Là những bộ công cụ tấn công có thể sử dụng để đẩy các phần mềm độc hại vào trong hệ thống. Các bộ công cụ này có khả năng giúp cho kẻ tấn công có thể truy nhập bất hợp pháp vào hệ thống hoặc làm cho hệ thống bị lây nhiễm mã độc hại. Khi được tải vào trong hệ thống bằng các đoạn mã độc hại, Attacker Tool có thể chính là một phần của đoạn mã độc đó hoặc nó sẽ được tải vào hệ thống sau khi nhiễm.

*i.Phishing*

Là một hình thức tấn công thường có thể xem là kết hợp với mã độc hại. Phishing là phương thức dụ người dùng kết nối và sử dụng một hệ thống máy tính giả mạo nhằm làm cho người dùng tiết lộ các thông tin bí mật về danh tính (ví dụ như mật khẩu, số tài khoản, thông tin cá nhân…). Kẻ tấn công Phishing thường tạo ra trang web hoặc email có hình thức giống hệt như các trang web hoặc email mà nạn nhân thường hay sử dụng như trang của ngân hàng, của công ty phát hành thẻ tín dụng… Email hoặc trang web giả mạo này sẽ đề nghị nạn nhân thay đổi hoặc cung cấp các thông tin bí mật về tài khoản, về mật khẩu… Các thông tin này sẽ được sử dụng để trộm tiền trực tiếp trong tài khoản hoặc được sử dụng vào các mục đích bất hợp pháp khác.

*j.Virus Hoax*

Là các cảnh báo giả về Virus. Các cảnh bảo giả này thường núp dưới dạng một yêu cầu khẩn cấp để bảo vệ hệ thống. Mục tiêu của cảnh báo Virus giả là cố gắng lôi kéo mọi người gửi cảnh báo càng nhiều càng tốt qua email. Bản thân cảnh báo giả là không gây nguy hiểm trực tiếp nhưng những thư gửi để cảnh báo có thể chữa mã độc hại hoặc trong cảnh báo giả có chứa các chỉ dẫn về thiết lập lại hệ điều hành, xoá file làm nguy hại tới hệ thống. Kiểu cảnh báo giả này cũng gây tốn thời gian và quấy rối bộ phận hỗ trợ kỹ thuật khi có quá nhiều người gọi đến và yêu cầu dịch vụ.

### **1.1.3 Thực trạng mã độc hiện nay.**

Do sự phát triển nhanh chóng của các thiết bị đầu cuối thông minh di động, Android đã trở thành nền tảng tính toán được sử dụng phổ biến nhất trên các điện thoại thông minh. Theo báo cáo từ TrendForce, trong năm 2020, tổng cộng 1,25 tỷ điện thoại thông minh đã được sản xuất, và Android chiếm 78,4% thị phần.Tuy nhiên, do sự phân bố rộng rãi và bản chất mã nguồn mở, các ứng dụng Android có thể được truy cập từ các bên thứ ba tiềm ẩn nguy cơ độc hại bên cạnh Cửa hàng Android chính thức, điều này khiến nền tảng trở thành mục tiêu cho các cuộc tấn công mã độc. Theo Báo cáo đặc biệt về mã độc Android 2019 được phát hành bởi 360 Security vào ngày 28 tháng 2, 2020, nền tảng này đã chặn khoảng 1,809 triệu mẫu mã độc mới trên các thiết bị đầu cuối di động trong năm 2019, và trung bình mỗi ngày có khoảng 5.000 mẫu mã độc di động mới bị chặn.

Các bài đánh giá về phát hiện mã độc trên Android đã được đề xuất trong nhiều nghiên cứu trước đây, đã nghiên cứu các mô hình học máy để phát hiện mã độc trên Android trong những năm gần đây nhưng đã bỏ qua việc xem xét các hạn chế của các mô hình học máy được đề cập. Bakour, Ünver, và Ghanemđã cung cấp một phân loại rộng cho tất cả các hướng nghiên cứu. Các tổng quan về phân tích tĩnh và phát hiện truyền thống đều được đưa vào bài viết này để đưa ra kết luận từ các góc nhìn khác nhau.

Tuy nhiên, nhằm giải quyết các hạn chế của các nghiên cứu đã thu thập nhưng lỗi thời, sự bỏ qua tầm quan trọng của các thuật toán lựa chọn đặc trưng và các vấn đề của các mô hình học máy, bài viết này cung cấp một cái nhìn toàn diện hơn về các công trình nghiên cứu với khối lượng lớn các tài liệu liên quan từ năm 2015 đến 2021.

Theo hiểu biết của tôi, rất ít bài đánh giá đã trình bày một cách có hệ thống các nghiên cứu tập trung vào bốn giai đoạn xử lý đặc trưng và các mô hình phân loại mã độc Android theo phân loại kỹ thuật phát hiện và loại hình học máy. So với các nghiên cứu khác, công trình này chỉ ra một số lĩnh vực bị bỏ qua trong các nghiên cứu trước đây trong lĩnh vực phát hiện mã độc Android, như các hạn chế và thách thức của các mô hình học máy. Ngoài ra, các bộ dữ liệu Android được áp dụng rộng rãi và các công cụ giải mã cũng được tổng hợp. Các nghiên cứu liên quan đến mã độc Android được thu thập trong bài viết này có thể cung cấp tài liệu tham khảo quý giá và mở rộng hướng nghiên cứu cho các nhà nghiên cứu trong tương lai.

## **1.2. Tổng quan về Android.**

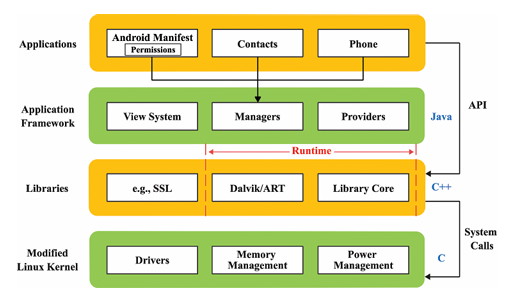
### **1.2.1 Kiến trúc về hệ điều hành Android.**

Hệ điều hành Android là một hệ điều hành mã nguồn mở dựa trên Linux dành cho nền tảng di động, được phát hành bởi Google. Ngày nay, Android đang được cập nhật nhanh chóng mặc dù kiến trúc chính của hệ điều hành Android vẫn không thay đổi.

Kiến trúc của Android có thể được chia thành các lớp sau: Lớp nhân Linux sửa đổi**,** Thư viện**,** Lớp thư viện runtime hệ thống, và Lớp khung ứng dụng như được minh họa trong Hình 1.

Modified Linux Kernel: Các dịch vụ hệ thống cốt lõi do Android cung cấp được xây dựng dựa trên hệ thống Linux, như bảo mật, quản lý năng lượng và trình điều khiển. Đóng vai trò như một lớp trừu tượng giữa phần cứng và phần mềm, nhân Linux đã được sửa đổi giúp ẩn đi các chi tiết của lớp phần cứng và cung cấp dịch vụ cho các lớp trên nhằm giảm sự phụ thuộc lẫn nhau.

Libraries: Cơ chế sandbox quy trình dựa trên Linux trong lớp thư viện của Android là một trong những nền tảng của toàn bộ thiết kế bảo mật. Dựa vào lớp nhân Linux đã được sửa đổi để thực hiện các chức năng cơ bản như quản lý luồng và quản lý bộ nhớ, máy ảo Dalvik được tối ưu hóa để chạy hiệu quả nhiều phiên bản máy ảo đồng thời trong bộ nhớ hạn chế. Mỗi ứng dụng Android được thực thi dưới dạng một quy trình Linux với một phiên bản của máy ảo Dalvik.



Hình 1. Kiến trúc hệ điều hành android

System Runtime: Từ góc độ kiến trúc tổng thể, nhà phát triển chỉ có quyền kiểm soát chủ động đối với lớp System Runtime và các cấu trúc phía trên nó, do đó việc phát hiện phần mềm độc hại trên Android cũng nên tập trung vào cùng vị trí này. Nó có thể được phân loại thành thư viện hệ thống và môi trường chạy Android. Thư viện lõi của môi trường chạy Android cung cấp hầu hết các API, chẳng hạn như Android OS, Android.net và Android.media.

Application Framework Layer: Android có một lớp Application Frameworkcung cấp nhiều API phục vụ cho việc phát triển ứng dụng Android. Các nhà phát triển có thể tự do sử dụng các API này để xây dựng ứng dụng của mình, nhưng phải tuân thủ các giới hạn bảo mật trong quá trình triển khai của framework.

### **1.2.2. Các thành phần chính của ứng dụng Android.**

Bốn thành phần chính của Android bao gồm Activity**,** Service**,** Broadcast Receiver, và Content Provider, chúng có mối liên hệ chặt chẽ với hành vi của phần mềm độc hại trên Android.

Activity:Activity cung cấp cho người dùng một cửa sổ đồ họa để thực hiện các hành động như nhấn nút, hiển thị khối văn bản, hoặc nhập liệu. Người dùng tương tác với ứng dụng bằng cách chạm vào các thành phần này. Activity thường đóng vai trò là một lớp trung gian giữa người dùng và các chức năng của ứng dụng, chịu trách nhiệm truyền tải ý định của người dùng.

Service: Service thường được sử dụng để xử lý các logic tốn thời gian ở chế độ nền, do đó nhiều hành vi độc hại có liên quan đến **Service** vì tính chất không hiển thị với người dùng. **Service** không chạy trong một tiến trình riêng biệt mà phụ thuộc vào tiến trình của ứng dụng nơi nó được tạo ra.

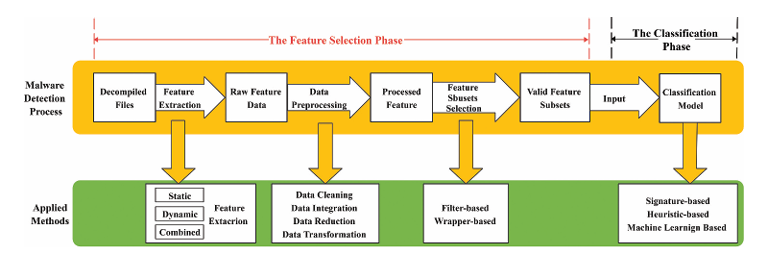
Broadcast Receiver:Là một cơ chế được sử dụng rộng rãi để truyền thông tin giữa các ứng dụng, Broadcast Receiver thường được các nhà phát triển phần mềm độc hại sử dụng để theo dõi các sự kiện liên quan đến thông tin nhạy cảm. Broadcast Receiver lọc, nhận và phản hồi các tín hiệu phát đi. Broadcast Receiver cho phép ứng dụng Android phản ứng với các sự kiện bên ngoài, chẳng hạn như bật nguồn điện thoại, nhận tin nhắn hoặc cuộc gọi.

Content Provider: Content Provider có thể hỗ trợ phần mềm độc hại trên Android thực hiện hành vi xấu bằng cách sử dụng quyền chia sẻ dữ liệu. Content Provider hỗ trợ lưu trữ và đọc dữ liệu giữa nhiều ứng dụng, hoạt động như một cơ sở dữ liệu cho các ứng dụng, cho phép truy cập vào dữ liệu được chia sẻ, chẳng hạn như danh bạ và tin nhắn, tạo điều kiện cho các nhà phát triển phần mềm độc hại khai thác.

Các quyền cần thiết và từng thành phần trong bốn thành phần được sử dụng trong một ứng dụng Android đều phải được khai báo trong tệp AndroidManifest.xml. Do đó, việc phân tích tệp AndroidManifest.xml có thể cung cấp cái nhìn tổng quan về chức năng cũng như hành vi độc hại của ứng dụng. Tệp AndroidManifest.xml thường được sử dụng như một chỉ số hỗ trợ để kết hợp với các phương pháp phân tích khác nhằm phát hiện phần mềm độc hại.

### **1.2.3. Lựa chọn phương pháp trên Androi.**

Lựa chọn phương pháp cải thiện hiệu quả phát hiện phần mềm độc hại bằng cách loại bỏ các tính năng dư thừa và không liên quan trong quá trình phát hiện phần mềm độc hại trên Android. Hình phác họa quá trình của giai đoạn lựa chọn tính năng.

****

Hình 2. Qúa trình lựa chọn tính năng

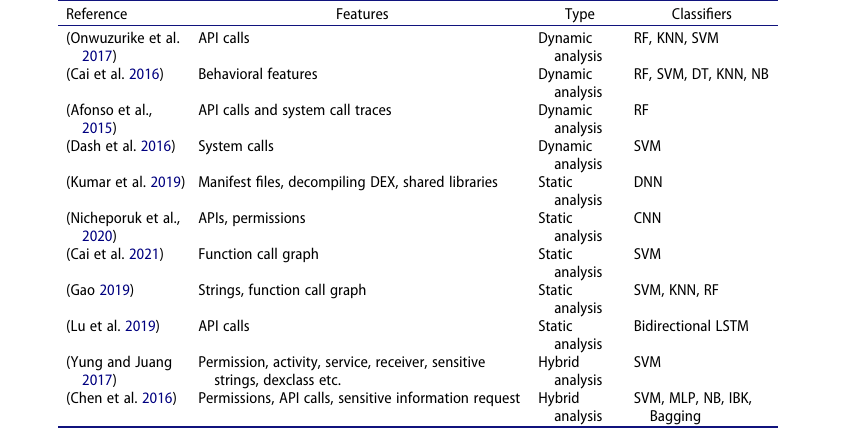
Để chứng minh tầm quan trọng của việc lựa chọn đặc trưng, kiểm tra hiệu quả của việc lựa chọn đặc trưng trong phát hiện phần mềm độc hại bằng cách sử dụng các thuật toán học máy có giám sát và không giám sát, với hoặc không có lựa chọn đặc trưng. Lấy độ chính xác dự đoán của thuật toán làm chỉ số hiệu suất, kết quả cho thấy tỷ lệ phát hiện tốt nhất đạt được là khi sử dụng thuật toán học có giám sát kết hợp với lựa chọn đặc trưng. So với việc áp dụng mà không có lựa chọn đặc trưng, độ chính xác chính đã tăng từ 54,56% lên 74,5%, cho thấy tầm ảnh hưởng của việc lựa chọn đặc trưng.

### **1.2.4 Khai thác tính năng.**

Theo cách tiếp cận trích xuất tính năng bằng cách sử dụng tính năng tĩnh, tính năng động hoặc cả hai, công nghệ phát hiện phần mềm độc hại trên Android có thể được phân loại thành phân tích động, phân tích tĩnh và phân tích kết hợp.

Dynamic analysis (phân tích động).Phân tích động là một phương pháp chạy chương trình trong môi trường sandbox và theo dõi hành vi của chuỗi lệnh gọi API, lệnh hệ thống, lưu lượng mạng và dữ liệu CPU để giám sát luồng dữ liệu trong quá trình chạy chương trình, từ đó tiết lộ hành vi thực tế của chương trình gần với tình huống thực tế hơn. Tuy nhiên, phương pháp này không được sử dụng rộng rãi do yêu cầu lượng lớn tài nguyên và tốc độ phát hiện chậm trong khi chạy chương trình.

Static analysis (phân tích tĩnh). được thực hiện bằng cách phân tích các tệp Android và trích xuất thông tin như các quyền được yêu cầu, chuỗi opcode và lệnh gọi API, v.v. Phát hiện tĩnh được sử dụng rộng rãi trong lĩnh vực phát hiện phần mềm độc hại trên Android nhờ nhiều tính năng tùy chọn dễ trích xuất.



Hình 3. Tóm tắt về trính xuất đặc trưng trên Android

Hybrid analysis (Phân tích lai). Sự kết hợp giữa phân tích động và tĩnh có thể giúp việc phát hiện phần mềm độc hại trên Android chính xác và hiệu quả hơn.

Hơn nữa, API thường được sử dụng để phát hiện phần mềm độc hại trên Android trong cả phân tích đặc tính tĩnh và phân tích đặc tính động**.** Sau khi xử lý chuỗi API hoặc đồ thị gọi hàm và đưa chúng vào RNN (Mạng nơ-ron hồi tiếp), đường dẫn hành vi của phần mềm có thể được phát hiện, điều này khác biệt so với phần mềm an toàn.

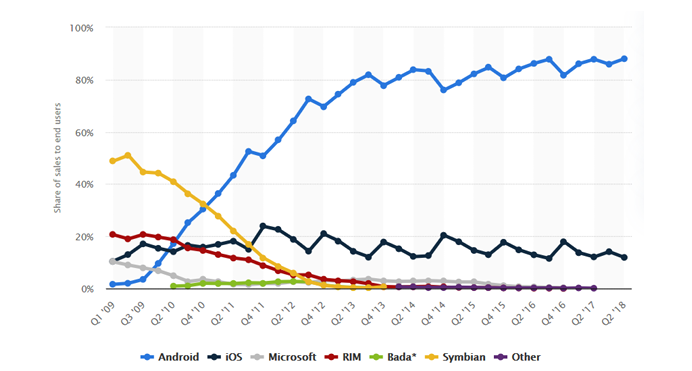
# **CHƯƠNG 2. TỔNG QUÁT HÓA VỀ MÃ ĐỘC ANDROID**

Trong phần này, sẽ trình bày một lịch sử ngắn gọn về phần mềm độc hại (malware) và sự phát triển của nó. Trọng tâm chính của chúng tôi sẽ là phần mềm độc hại trên Android, nhưng không thể tránh khỏi việc cung cấp một cái nhìn tổng quan ngắn gọn về phần mềm độc hại trên thiết bị di động nói chung. Chúng tôi cũng sẽ thảo luận về các cuộc tấn công và các chiến thuật lẩn tránh mà phần mềm độc hại sử dụng để tránh bị phát hiện.

## **2.1. Sự phát triển mã độc.**

Phần mềm độc hại trên thiết bị di động đã trải qua một chặng đường dài kể từ khi sâu di động đầu tiên, được thiết kế để lây nhiễm các thiết bị Nokia thuộc dòng 60. Ban đầu, sự lây nhiễm này có vẻ vô hại khi sâu chỉ hiển thị từ "Caribe" trên màn hình. Sâu này lây lan bằng cách sử dụng Bluetooth để tự truyền sang các thiết bị Bluetooth lân cận như máy in, điện thoại di động, v.v. . Vào thời điểm đó, hệ điều hành Symbian rất phổ biến, trở thành một thị trường tiềm năng lớn cho các nhà phát triển phần mềm độc hại di động. Đáng chú ý, năm 2005 chứng kiến sự sụt giảm đáng kể trong thị phần của Symbian, điều này có thể là do sự lây lan của Cabir trên các điện thoại sử dụng hệ điều hành Symbian .Năm 2005, Cabir đã được tiếp nối bởi CommWarrior, một loại virus có khả năng tự phát tán qua cả MMS lẫn Bluetooth. Virus này được thiết kế cho nền tảng Symbian 60 và đã lây nhiễm hơn 100.000 thiết bị di động bằng cách gửi hơn 450.000 tin nhắn MMS . Việc phát tán này mang lại giá trị tài chính cho việc phát triển phần mềm độc hại vì mỗi tin nhắn MMS được gửi đi sẽ khiến chủ sở hữu thiết bị phải chịu một khoản phí từ nhà mạng.Lợi ích tài chính này tiếp tục được khai thác bởi một trojan có tên RedBrowser, được phát hiện vào năm 2006. Trojan này được thiết kế để tận dụng dịch vụ SMS với phí cao, khi mỗi tin nhắn được gửi đi thường khiến chủ sở hữu thiết bị mất 5 USD . RedBrowser đánh dấu một bước ngoặt trong sự phát triển của phần mềm độc hại di động, vì đây là phần mềm độc hại đầu tiên có thể lây nhiễm điện thoại di động trên nhiều hệ điều hành khác nhau bằng cách khai thác điểm yếu trong Java 2 Micro Edition (J2ME), vốn được hỗ trợ rộng rãi .Hai năm tiếp theo, 2007 và 2008, là giai đoạn trầm lắng trong việc phát triển các mối đe dọa phần mềm độc hại di động mới, và sự phát triển phần mềm độc hại phi thương mại gần như ngừng hẳn . Một ví dụ về phần mềm độc hại phi thương mại là sâu di động chuyển sang việc đánh cắp dữ liệu, khi tội phạm mạng bắt đầu nhắm mục tiêu vào các game thủ trực tuyến để chiếm đoạt mật khẩu, tài sản trong trò chơi và các nhân vật ảo .

Vào năm 2009, phần mềm độc hại botnet trên di động có tên Yxes đã được phát hiện và trở thành tiêu điểm tin tức như một trong những phần mềm độc hại đầu tiên dành cho hệ điều hành Symbian OS 9. Phát hiện này đánh dấu một bước ngoặt khác trong sự phát triển của phần mềm độc hại di động cũng như đổi mới công nghệ, vì đây là phần mềm độc hại đầu tiên có khả năng gửi tin nhắn SMS và truy cập Internet .Từ năm 2009, phần mềm độc hại trên di động đã tăng trưởng theo cấp số nhân nhờ vào những cải tiến công nghệ, mang lại các phương thức mới để khai thác lợi nhuận, cùng với sự gia tăng khả năng tiếp cận chợ đen để mua bán thông tin bị đánh cắp, và sự hợp tác giữa các nhà phát triển phần mềm độc hại trong việc trao đổi mã nguồn và lỗ hổng hệ thống .Theo Apvrille và cộng sự , năm 2010 được coi là thời kỳ công nghiệp của phần mềm độc hại di động, khi lĩnh vực này chuyển từ hoạt động của các cá nhân sang tổ chức tội phạm mạng có tổ chức, với động lực tài chính tăng mạnh mẽ. Quý đầu tiên của cùng năm cũng chứng kiến sự gia tăng mạnh mẽ về mức độ phổ biến của Android. Thị phần toàn cầu của Android đã tăng từ 1,6% trong quý 1 năm 2009 lên 88% trong cùng kỳ năm 2010. Hình 3 dưới đây minh họa xu hướng phổ biến này.



Hình 4. Thị phần toàn cầu

Việc sử dụng ngày càng tăng của nền tảng Android đã khiến hệ thống này trở nên dễ bị tổn thương trước các loại phần mềm độc hại mới và tinh vi. Những phần mềm độc hại này được thiết kế để tránh bị phát hiện và gây ra những hậu quả nghiêm trọng cho các nạn nhân. Các tác giả phần mềm độc hại gây ra thiệt hại lên tới hàng tỷ đô la mỗi năm . Do đó, đây trở thành một lĩnh vực đầy hấp dẫn khi số lượng người dùng hệ thống Android tiếp tục gia tăng.Theo ước tính của Amro, cứ mỗi 10 giây lại phát hiện một biến thể phần mềm độc hại mới. Tốc độ này không chỉ đáng báo động mà còn đòi hỏi nguồn lực khổng lồ để phát hiện và vô hiệu hóa. Tỷ lệ gia tăng của phần mềm độc hại lẩn tránh đã nhảy vọt lên hơn 2000% vào năm 2017, và xu hướng này được dự đoán sẽ tiếp tục.Các cuộc tấn công quy mô lớn đang được thực hiện, khiến hàng nghìn người dùng trở thành nạn nhân và chịu tổn thất nặng nề. Các tác giả phần mềm độc hại đã sử dụng những kỹ thuật tinh vi để vượt qua các yêu cầu xác thực được đưa vào như là các biện pháp bảo mật. Một số phần mềm độc hại thậm chí tấn công tất cả các thiết bị xác thực, làm lộ ra những lỗ hổng trong hệ thống xác thực.

Hồi chuông cảnh báo đã được gióng lên, chủ yếu do sự gia tăng các trường hợp ransomware, nơi các tác giả phần mềm độc hại chiếm quyền kiểm soát hệ thống Android và yêu cầu tiền chuộc để trả lại quyền truy cập. Các tác giả phần mềm độc hại thường sử dụng tiền điện tử không thể truy vết để che giấu hành tung của mình.Ngoài các mục tiêu phổ biến của phần mềm độc hại như khóa màn hình thiết bị, ransomware còn hỗ trợ các mục đích xấu khác như xóa dữ liệu, đặt lại cài đặt bảo mật, theo dõi GPS và đánh cắp thông tin cá nhân. Tuy nhiên, ngay cả khi các mối đe dọa phần mềm độc hại ngày càng gia tăng, các công ty chống phần mềm độc hại và Google vẫn đang đầu tư nguồn lực để phát triển các kỹ thuật phát hiện ngày càng hiệu quả.Các cơ sở dữ liệu quan trọng đã được xây dựng theo thời gian và đóng vai trò đáng kể trong việc nâng cao bảo mật Android cho hàng triệu người dùng. Tuy nhiên, có thể suy luận rằng các tác giả phần mềm độc hại dường như luôn đi trước một bước.

## **2.2. Các họ mã độc**

Dựa trên khả năng của phần mềm độc hại đó, chúng được phân loại thành các họ khác nhau. Số lượng ứng dụng độc hại đã tăng đều đặn kể từ năm 2010 và xu hướng này dự kiến sẽ tiếp tục. Năm 2013 con số này làước tính khoảng 500.000, tăng lên hơn 2,5 triệu ứng dụng vào năm 2015. Con số này tăng lên 3,5 triệu vào năm 2017 và dự kiến sẽ tiếp tục tăng với các phần mềm độc hại phức tạp hơn và có sức tàn phá cao hơn được mong đợi. Phần mềm tống tiền là một trong những dạng phần mềm độc hại được báo cáo nhiều nhất và chiếm gần 30% số trường hợp trong năm 2017.

Các phiên bản tiếp theo của phần mềm độc hại có nhiều khả năng trốn tránh bị phát hiện hơn và nhiều tính năng có hại hơn. Hầu hết các họ phần mềm độc hại đều đã được xác định và phát hiện. Sự nhận dạng này đã chứng kiến một giảm số lượng các họ phần mềm độc hại mới được báo cáo hàng năm. Tuy nhiên, các biến thể của phần mềm độc hại trong các họ tiếp tục gia tăng. Một số phần mềm độc hại này đã tìm được đường đến cửa hàng Google Play, trong khi phần lớn chúng đến từ thị trường bên thứ ba.

Bảng các họ mã độc phổ biến trên hệ thống Android.

|  |  |
| --- | --- |
| **Họ phần mềm độc hại** | **Mô tả** |
| AnserverBot | Ứng dụng máy chủ bị nhiễm hiển thị một hộp thoại mới để yêu cầu và nâng cấp ứng dụng mới. Ứng dụng đã tải xuống có chứa tải trọng ẩn. Tải trọng giao tiếp với máy chủ từ xa để nhận lệnh |
| BaseBridge (AdSMS) | Khi ứng dụng máy chủ bị lây nhiễm được cài đặt, nó sẽ lừa người dùng nâng cấp ứng dụng. Ứng dụng đã tải xuống đã được cài đặt và sẽ liên lạc với máy chủ từ xa để tải xuống tệp cấu hình. Tệp chứa các số đặc biệt mà phần mềm độc hại sẽ gọi hoặc gửi tin nhắn SMS |
| BeanBot | Phần mềm độc hại gửi dữ liệu thiết bị đến máy chủ từ xa. Nó cũng gửi tin nhắn SMS cao cấp |
| Pjapps | Phần mềm độc hại giao tiếp với máy chủ từ xa và gửi lịch sử và dấu trang của Duyệt. Nó cũng gửi tin nhắn SMS được xếp hạng cao cấp. |
| BGSERV | Phần mềm độc hại gửi SMS được xếp hạng cao cấp. Nó cũng liên lạc với một máy chủ từ xa để gửi thông tin cá nhân. |
| CruseWin (CruseWind) | Phần mềm độc hại gửi SMS được xếp hạng cao cấp. Nó cũng có thể tự nâng cấp. |
| DroidCoupon | Nó sử dụng một cách khai thác gốc đơn giản—"Rage Against the Cage" trong Android 2.2 trở về trước, ẩn Nền tảng, vì vậy rất khó để phát hiện ra nó. Phần mềm độc hại làm rò rỉ thông tin cá nhân của người dùng. |
| DroidDeluxe | Cài đặt công cụ khôi phục mật khẩu và Công cụ này sẽ không hoạt động trên Android 2.3 với thông báo: "This application has stopped.” |
| DroidDream (DORDRAE) | Nó ngụy trang dưới dạng các ứng dụng như công cụ theo dõi pin, công cụ liệt kê nhiệm vụ. Phần mềm độc hại gửi thông tin của thiết bị đến máy chủ từ xa. |
| DreamLight | Dịch vụ có tên "CoreService" được bắt đầu khi thiết bị bị nhiễm nhận được một cuộc gọi điện thoại. Phần mềm độc hại gửi thông tin của thiết bị đến máy chủ từ xa. |
| DroidKungFu (LeNa) | Phần mềm độc hại đã root thiết bị. Phần mềm độc hại gửi thông tin của thiết bị đến máy chủ từ xa. |
| Smssend(fake player) | Phần mềm độc hại gửi SMS được xếp hạng cao cấp. |
| Gamblersms | Phần mềm độc hại yêu cầu người dùng cung cấp số điện thoại và địa chỉ email. Thông tin này sau đó bị lạm dụng. |
| Geinimi | Phần mềm độc hại trích xuất thông tin từ thiết bị và gửi nó đến một địa điểm từ xa. Hacker có thể thực thi các lệnh từ xa, chẳng hạn như gửi SMS và gọi điện thoại. |
| GGTracker | Phần mềm độc hại lừa người dùng tải xuống ứng dụng tiết kiệm pin thông qua quảng cáo độc hại. Sau khi cài đặt, nó sẽ gửi tin nhắn văn bản cao cấp. |
| GoldDream | Phần mềm độc hại có khả năng bot khi giao tiếp với máy chủ từ xa để tìm nạp lệnh hoặc gửi các thông tin cá nhân. Nó chặn các tin nhắn SMS đến và đi chuyển tiếp chúng đến máy chủ từ xa. |
| GPSSMSSpy (mobinauten, SmsHowU, SMS spy) | Phần mềm độc hại gửi tin nhắn SMS giá cao. |
| Jifake | Phần mềm độc hại được cài đặt khi người dùng mở liên kếttải xuống apk từ một trang web độc hại nhưng người dùng không thấy bất kỳ tệp nào đang được tải xuống. Phần mềm độc hại gửi tin nhắn SMS giá cao. |
| Plankton | Phần mềm độc hại được cài đặt cùng với một ứng dụng máy chủ bị nhiễm. Nó thu thập dữ liệu của người dùng và gửi đến máy chủ trung tâm. |

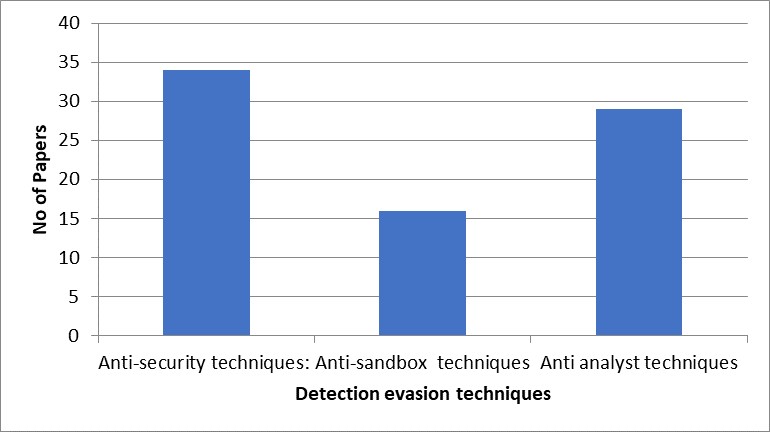
## **2.3. Mã độc tấn công và lẩn tránh**

Các ứng dụng độc hại trên Android được phát triển với ý đồ xấu, có thể gây ra những tổn thất nghiêm trọng. Trong bối cảnh này, các nhà phát triển phần mềm độc hại luôn tìm kiếm các kỹ thuật lẩn tránh để tránh bị phát hiện và thực hiện các kế hoạch xấu xa của mình [40]. Các kỹ thuật lẩn tránh ngày càng trở nên phức tạp và tinh vi hơn, đồng thời xuất hiện trong nhiều họ phần mềm độc hại khác nhau.

Có ba kỹ thuật lẩn tránh phổ biến nhất:

* Kỹ thuật chống bảo mật (Anti-security techniques): Tránh bị phát hiện bởi phần mềm chống phần mềm độc hại bằng cách ẩn mình trong một tiến trình hợp pháp đang chạy trên thiết bị.
* Kỹ thuật chống môi trường thử nghiệm (Anti-sandbox techniques): Phát hiện phân tích tự động và do đó tránh thực thi hành vi độc hại.
* Kỹ thuật chống phân tích (Anti-analyst techniques): Sử dụng các công cụ giám sát như làm rối mã (obfuscation) để ngăn chặn kỹ thuật phân tích ngược (reverse engineering).

Những chiến thuật lẩn tránh này đã được đề cập rộng rãi trong các nghiên cứu hiện có.



Hình 5. Phát triển các kĩ thuật lẩn tránh qua nhiều nghiên cứu

Packing (Đóng gói): Mặc dù các ứng dụng độc hại được đóng gói đã tồn tại từ lâu, nhưng các xu hướng gần đây đã cho thấy các kỹ thuật đóng gói tinh vi hơn, gây khó khăn trong việc phát hiện. Kỹ thuật đóng gói cho phép các ứng dụng nén các tệp nhằm giảm kích thước của ứng dụng, sau đó giải nén và thực thi các tệp này trong bộ nhớ khi chạy. Đây là một kỹ thuật hợp pháp, nhưng cũng cho phép các ứng dụng độc hại nén mã độc trong các tệp classes.dex được mã hóa và giải nén chúng trong thời gian chạy để phục vụ cho mục đích xấu.

Multidex Applications (Ứng dụng Multidex): Các phương pháp phát hiện phần mềm độc hại thông thường tập trung vào tệp Dalvik Executable (DEX) trong hệ thống Android. Các ứng dụng thường có một tệp DEX duy nhất chứa mã thực thi. Tuy nhiên, một số phần mềm độc hại trên Android được thiết kế theo cách mà payload (mã thực thi độc hại) được chia thành hai tệp DEX. Kỹ thuật này có thể được sử dụng như một cách đơn giản để lẩn tránh nhằm tránh bị phát hiện; điều này đặc biệt đúng đối với các phương pháp phát hiện tĩnh phân tích một tệp DEX và bỏ sót tệp DEX thứ hai, có thể chứa một phần của mã độc.

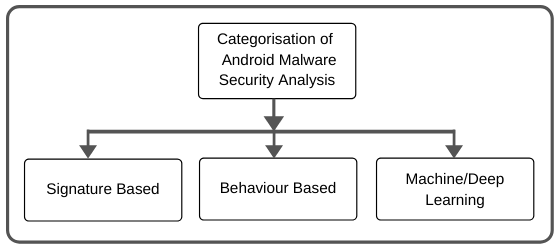
Instant Run-Based Malware (Phần mềm độc hại dựa trên Instant Run): Instant Run là một tính năng của Android cho phép các nhà phát triển phát hành các bản cập nhật nhanh hơn thông qua một ứng dụng gỡ lỗi, mà đưa vào một tệp zip vào ứng dụng để phục vụ mục đích cập nhật [42]. Các tác giả phần mềm độc hại đang ẩn payload (mã thực thi độc hại) vào các tệp zip được sử dụng bởi tính năng instant-run để tránh bị phát hiện. Cách tiếp cận này chỉ khả dụng cho Android Lollipop và các phiên bản sau và chỉ có thể được sử dụng trên các ứng dụng cài đặt thông qua bên ngoài (side-loading).

Malformed Manifest Files (Tệp Manifest bị hỏng): Kỹ thuật lẩn tránh này sử dụng các giá trị kỳ lạ trong tệp AndroidManifest.xml và tệp tài nguyên. Phương pháp này có thể làm rối các công cụ phát hiện tĩnh và tránh bị phát hiện.

Chattr (Chattr, một lệnh của Linux**):** Chattr đang được sử dụng để khóa phần mềm độc hại trên các hệ thống Android, chủ yếu là nơi có quyền truy cập root. Tiện ích chattr có thể được đóng gói vào một ứng dụng dưới dạng mã hóa và được sử dụng để khóa ứng dụng vào thư mục hệ thống. Khi một ứng dụng bị khóa, bất kỳ nỗ lực nào để xóa nó ngay cả khi có quyền truy cập root đều không thành công.

# **CHƯƠNG 3. MỘT SỐ PHƯƠNG PHÁP PHÂN LOẠI MÃ ĐỘC ANDROID**

Phần mềm độc hại được sử dụng để nhắm vào các loại ứng dụng khác nhau trong hệ thống Android. Các ứng dụng bao gồm ứng dụng giải trí, công cụ xã hội, ứng dụng năng suất, công cụ giao tiếp và các trò chơi xếp hình, trong số đó. Nghiên cứu hiện tại về bảo mật ứng dụng Android đã phát triển đáng kể với các kỹ thuật mới được sử dụng để phân loại. Phân tích bảo mật ứng dụng Android được phân loại tùy thuộc vào bản chất của phần mềm độc hại và các kinh nghiệm trước đây với các mẫu tương tự. Các kỹ thuật phát hiện và phân loại phần mềm độc hại trên Android hiện tại được chia thành ba loại chính, như được hiển thị trong Hình 6 dưới đây:



Hình 6. Phân loại phân tích bảo mật phần mềm độc hại Android

## **3.1. Phát hiện dựa trên chữ kí (Signature base).**

Dựa trên việc khớp mẫu, tính năng phát hiện dựa trên chữ ký sẽ duy trì thư viện chữ ký phần mềm độc hại chứa chữ ký duy nhất cho từng phần mềm độc hại Android đã biết. Thư viện chữ ký phần mềm độc hại bao gồm các thuộc tính khác nhau như tên tệp, chuỗi nội dung hoặc byte, được các chuyên gia xác định theo cách thủ công hoặc được tạo tự động. Nó phát hiện mẫu Android bằng cách kiểm tra xem có chữ ký phần mềm độc hại phù hợp trong thư viện hay không.

Công nghệ này tiện lợi nhất và được sử dụng phổ biến do tốc độ phát hiện nhanh và độ chính xác cao. Tất cả phần mềm độc hại Android được ghi trong thư viện phần mềm độc hại đều có thể được phát hiện chính xác. Tuy nhiên, điểm bất lợi là việc duy trì thư viện chữ ký phần mềm độc hại tốn nhiều thời gian và không áp dụng được để phát hiện phần mềm độc hại mới.

## **3.2. Phát triển dựa trên hành vi (Behavior based).**

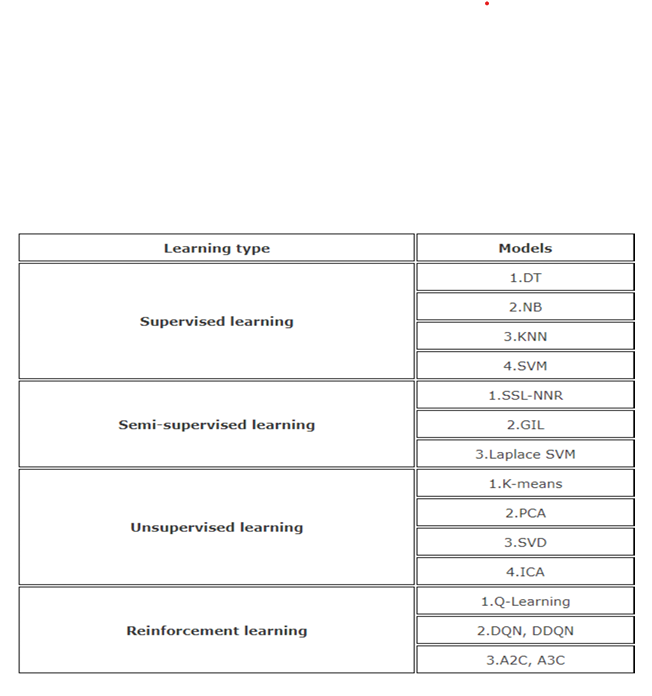
Phát hiện dựa trên hành vi( dựa trên sự bất thường), nhấn mạnh đến khả năng xác định phần mềm độc hại không xác định. Phương pháp này so sánh các đặc điểm của các mẫu chưa xác định với các họ phần mềm độc hại đã biết và mỗi họ phần mềm độc hại được thể hiện bằng một bộ quy tắc được xác định để khai thác kinh nghiệm và kiến thức chung về phần mềm. Nó được coi là phần mềm độc hại khi các đặc điểm của mẫu được phát hiện phù hợp với quy tắc của một họ phần mềm độc hại. Các bộ quy tắc đã biết bao gồm các thuộc tính như tính năng cấu trúc phần mềm, lệnh gọi API, chuỗi mã hoạt động và quy tắc tích hợp nhiều chế độ xem, v.v.

Các kỹ thuật phát hiện dựa trên hành vi có khả năng tự phát hiện phần mềm độc hại không xác định và ủng hộ việc sử dụng nhiều phương pháp để xác định sự khác biệt giữa phần mềm độc hại và phần mềm lành tính. Nó bù đắp cho sự thiếu hụt của khả năng phát hiện truyền thống và cũng có thể xác định phần mềm độc hại không xác định nhưng có nhược điểm là tỷ lệ lỗi cao hơn đối với phần mềm độc hại zero-day.

## **3.3. Phát triển dựa trên học máy (Machine learning).**

Học máy đào tạo người học bằng cách điều chỉnh các tham số để đưa ra dự đoán tốt nhất. Nghiên cứu hiện tại đã chứng minh rằng học máy là một phương pháp hiệu quả và đầy hứa hẹn để phát hiện phần mềm độc hại trên Android. Trong những năm gần đây, nhiều công trình phát hiện phần mềm độc hại đã cố gắng khai thác công nghệ máy học để tìm kiếm bước đột phá trong việc phát hiện phần mềm độc hại không xác định trên Android. Các phần sau đây sẽ giới thiệu chi tiết về công nghệ phát hiện dựa trên học máy.

Trong nỗ lực giải quyết vấn đề thiếu khả năng xác định phần mềm độc hại không xác định hoặc phần mềm độc hại chưa từng biết đối với các phương pháp truyền thống, học máy được sử dụng rộng rãi để phát hiện phần mềm độc hại trên Android trong nghiên cứu những năm gần đây. Học máy có thể được chia đại khái thành năm loại: chủ nghĩa tượng trưng, chủ nghĩa bayes, chủ nghĩa kết nối, chủ nghĩa tiến hóa và chủ nghĩa tương tự hành vi, theo khái niệm cơ bản Theo loại học, học máy được sử dụng để phát hiện phần mềm độc hại trên Android có thể được chia thành bốn loại được mô tả trong hình sau:



Hình 7. Phân loại các phương pháp học máy

**Học có giám sát( Supervised learning):** Dữ liệu huấn luyện được gắn nhãn danh mục là đầu vào của các mô hình học máy trong học tập có giám sát. Đó là nhiệm vụ phân loại khi học có giám sát đưa ra các dự đoán rời rạc về nhiều thứ khác nhau, nhiệm vụ hồi quy khi học có giám sát đưa ra dự đoán về các giá trị liên tục.

**Học tập không giám sát( Unsupervised learning):** Mô hình dự đoán được huấn luyện thông qua các tập dữ liệu chưa được gắn nhãn, với chủ đề là khám phá và suy ra các kết nối tiềm năng từ dữ liệu chưa được gắn nhãn trong quá trình học tập không giám sát. Các nhiệm vụ điển hình là phân cụm và giảm kích thước.

**Học bán giám sát( Semi-supervised learning):** Kết hợp với học có giám sát và học không giám sát, chỉ một số phần của dữ liệu huấn luyện được gắn nhãn là học bán giám sát. Nó tìm hiểu cấu trúc bên trong của dữ liệu và sau đó tổ chức dữ liệu một cách hợp lý để dự đoán chỉ với một vài bộ dữ liệu được đánh dấu.

**Học tăng cường( Reinforcement learning):** Học tăng cường có thể được áp dụng cho các tính năng chọn lọc của Android, sử dụng kết quả phân loại của dữ liệu đầu vào làm phản hồi cho mô hình phân loại, với nguyên tắc tác nhân tối ưu hóa hành động tiếp theo của mình để tối đa hóa giá trị phần thưởng.

Ba loại mô hình và thuật toán chính được sử dụng để phát hiện phần mềm độc hại trên Android như sau: loại thứ nhất (1)- (6) là mô hình học máy truyền thống, mô hình thứ hai là Neural Network và Deep Learning(7)-(8) và mô hình thứ ba sử dụng mô hình học tập tổng hợp (9) kết hợp nhiều bộ phân loại để phát hiện phần mềm độc hại trên Android.

1, **Linear Model:** Các hàm tuyến tính đơn giản và có khả năng diễn giải cao sử dụng các tính năng của Android làm đầu vào được áp dụng để đưa ra dự đoán về phần mềm độc hại. Mô hình tuyến tính điển hình bao gồm hồi quy logistic và hồi quy tuyến tính, với điểm khác biệt là hồi quy logistic là giải quyết vấn đề phân loại trong khi hồi quy tuyến tính giải quyết các vấn đề hồi quy (Zhang và cộng sự 2019) cung cấp một phương pháp gián tiếp để chẩn đoán bất thường bằng cách xây dựng một mô hình tuyến tính chuyên dụng để ước tính cục bộ phân số bất thường được tạo ra bởi mô hình hộp đen.

2, **Support Vector Machine:** Nó cho thấy sự cải thiện đáng kể trong việc giám sát hiệu quả mức tiêu thụ tài nguyên khi chạy phần mềm độc hại Android bằng Support Vector Machine (SVM). SVM là tìm một siêu phẳng (Boswell 2002) phân chia hoàn hảo dữ liệu n chiều thành hai loại. (Faiz, Hussain và Marchang 2020) đã áp dụng SVM bằng cách sử dụng các tính năng được trích xuất từ quyền của Android, bộ thu phát sóng và API để phát hiện phần mềm độc hại trên Android, với độ chính xác phân loại cao nhất là 98,55% mà PersonaCateg-SVM đạt được.

3, **Naive Bayes:** Dựa trên định lý Bayes, Naive Bayes (NB) giả định rằng tác động của một giá trị thuộc tính lên một lớp nhất định là độc lập với giá trị của các thuộc tính khác (Leung 2007). (Alqahtani, Zagrouba và Almuhaideb 2019) đã cung cấp bài đánh giá chi tiết về các trình phát hiện máy học, tóm tắt chi tiết NB, SVM và DNN được áp dụng trong việc phát hiện phần mềm độc hại trên Android.

4, **Decision Tree:** Là một trong những mô hình học có giám sát được áp dụng phổ biến nhất trong lý luận quy nạp, Decision Tree(DT) xây dựng cấu trúc cây giống như sơ đồ từ dữ liệu huấn luyện. (Lashkari và cộng sự 2018) đã sử dụng RF, KNN và DT làm công cụ phân loại phát hiện phần mềm độc hại trên Android để so sánh, với mỗi thuật toán machine learning được đào tạo, kiểm tra và đánh giá với cùng các tính năng được chọn.

5**, K-Nearest Neighbor:** Là một mô hình học có giám sát, K Nearest Neighbor (KNN) có thể thu được kết quả phân loại phần mềm độc hại trên Android thông qua đo khoảng cách Euclide trong không gian hình học giữa các giá trị riêng khác nhau (Ray 2019).

6, **K-means Clustering:** Thuật toán K-means Clustering là thuật toán học không giám sát thường được áp dụng trong phân loại dòng phần mềm độc hại trên Android (Ilham, Abderrahim và Abdelhakim 2018). Cho một tập hợp gồm N điểm dữ liệu Rd và số nguyên K trong không gian D chiều thực, cần tìm điểm trung tâm trong N điểm dữ liệu, do đó giảm thiểu khoảng cách bình phương trung bình của mỗi điểm dữ liệu đến tâm gần nhất của nó (Kanungo và cộng sự 2002).

7, **Neural Network:** Bao gồm một số lượng lớn các nơ-ron nhân tạo được kết nối, Mạng nơ-ron (NN) sử dụng các nơ-ron để phản ánh tín hiệu nhận được và trọng lượng để thể hiện cường độ của tín hiệu (Gershenson 2003). Các thuật toán mạng thần kinh được sử dụng phổ biến nhất là Perceptron Neural Network, Hopfield Neural Network và Self-Organized Map.

8, **Deep Learning:** Với nhiều mức biểu diễn dữ liệu thu được bằng cách soạn thảo các mô-đun phi tuyến chuyển đổi mức biểu diễn thành mức biểu diễn cao hơn và trừu tượng hơn (LeCun, Bengio và Hinton 2015), Deep learning bắt nguồn từ NN (Du và cộng sự 2016) như được minh họa trong (Trích dẫn Qiu2020). Nó thường được sử dụng để phát hiện phần mềm độc hại trên Android khi các tính năng được chuyển thành hình ảnh.

9, **Ensemble learning:** Nhiều bộ phân loại đã được kết hợp để cải thiện độ chính xác khi phát hiện phần mềm độc hại trên Android trong quá trình học tập tổng hợp (Zhao và cộng sự 2018) (Rana và Sung 2020). Cụ thể hơn, Ensemble learning mô tả cách kết hợp người học. Một phương pháp tổng hợp phân loại mới dựa trên cấu trúc đa cấp đã được đề xuất bởi (Yerima và Sezer 2019), đào tạo các trình phân loại Android cơ bản ở cấp độ thấp hơn để tạo mô hình, sử dụng một bộ thuật toán sắp xếp để chọn phân loại cuối cùng và gán trọng số của kết quả dự đoán của bộ phân loại được chọn theo độ chính xác dự đoán của các bộ phân loại cơ bản ở mức cao hơn. Tuy nhiên, việc áp dụng phương pháp học tập hợp sẽ tốn kém về mặt tính toán vì mỗi tệp APK phải được phân tích bằng nhiều trình phát hiện. Để giải quyết vấn đề, (Birman và cộng sự 2019) đã áp dụng học tăng cường sâu để tự động khởi động và dừng các trình phân loại cơ sở, sử dụng DNN để tự động xác định xem có đủ thông tin để phân loại một tệp APK nhất định hay không.

Bảng điểm yếu điểm mạnh của các phương pháp phát hiện mã độc trên Android.

|  |  |  |
| --- | --- | --- |
| **Loại** | **Điểm mạnh** | **Điểm yếu** |
| Linear Model | Nhanh và chính xác | Tiền đề của thuật toán là nghiêm ngặt và không thể xử lý các tính năng đa chiều của Android |
| Support Vector Machine | Dễ dàng giải quyết ở quy mô nhỏ; các vấn dề chiều thấp và phi tuyến tính | Lượng lớn dữ liệu tiền xử lý; nhạy cảm với các giá trị bị thiếu |
| Decision | Có khả năng xử lí mô hình giá trị quy mô lớn và bị thiếu. | Overfirring; không hỗ trợ online |
| Naive Bayes | Nhanh và đơn giản | Không khả dụng khi các tính năng được trích xuất có tương quan |
| Deep Learning | Giải quyết các vấn đề khó khăn bằng mô hình phức tạp với nhiều tham số hơn | Hiệu quả đào tạo mô hình phức tạp thấp; Dễ dàng overfitting |
| Random Forest | Dễ để thực hiện; chi phí tính toán thấp; khả năng khái quát hóa mạnh mẽ | Hiệu suất ban đầu của Random forest không đạt yêu cầu |
| Neural Network | Độ chính xác cao và khả năng chịu lỗi mạnh | Yêu cầu nhiều dữ liệu đào tạo; các thông số và cấu trúc liên kết rất khó lựa chọn |
| K-nearest Neighbor | Có thể được thực hiện mà không cần ước tính tham số; thích hợp để giải quyết các đa dạng vấn đề | Nhạy cảm với tập dữ liệu; khối lượng tính toán tương đối lớn |
| K-means Clustering | Chi phí thấp | Kết quả bị ảnh hưởng bởi cài đặt ban đầu; nhạy cảm với nhiễu và các giá trị rời rạc |

# **CHƯƠNG 4. PHÂN TÍCH THUẬT TOÁN**

## **4.1. Thuật toán Neural Network(NN).**

Mạng Nơ-ron (Neural Network - NN) là một thuật toán học máy được sử dụng trong bài toán phân loại và dự đoán. Nó được lấy cảm hứng từ cách hoạt động của hệ thần kinh trong não người và bao gồm một số lớp nơ-ron nhân tạo liên kết với nhau để xử lý thông tin. NN được sử dụng để tìm ra các quy luật phức tạp trong dữ liệu và tự động học hồi quy hoặc phân loại.

**Nguyên lý hoạt động của NN:**

NN hoạt động bằng cách xây dựng nhiều lớp nơ-ron (lớp ẩn) với các kết nối giữa chúng. Các lớp nơ-ron này thực hiện các phép biến đổi trên dữ liệu đầu vào để tạo ra đầu ra dự đoán. Mỗi lớp nơ-ron bao gồm nhiều đơn vị nơ-ron (neurons) được kết nối với các đơn vị trong lớp trước đó. Các trọng số và hệ số tự do trong các kết nối này được điều chỉnh trong quá trình huấn luyện để tối ưu hóa hiệu suất của mô hình.

*Ưu điểm của Neural Network:*

Khả năng học mô hình phi tuyến tính: NN có khả năng học các quy luật phức tạp và phi tuyến tính trong dữ liệu, điều này giúp nó phù hợp với nhiều loại bài toán phân loại và dự đoán.

Khả năng học từ dữ liệu không cân bằng: NN có khả năng xử lý các tập dữ liệu không cân bằng, trong đó có ít mẫu dữ liệu thuộc một lớp so với lớp khác.

Độ linh hoạt trong việc thiết kế kiến trúc: Chúng ta có thể điều chỉnh kiến trúc của NN bằng cách thêm hoặc loại bỏ các lớp, nơ-ron, và điều chỉnh các tham số để tối ưu hóa mô hình cho từng bài toán cụ thể.

*Hạn chế của Neural Network:*

Yêu cầu lượng dữ liệu lớn: NN thường yêu cầu lượng dữ liệu lớn để đạt hiệu suất tốt. Trong trường hợp dữ liệu hạn chế, có nguy cơ overfitting, khi mô hình học quá nhiều chi tiết từ dữ liệu huấn luyện.

Khó để hiểu và giải thích: Kiến trúc phức tạp của NN khiến nó khó để giải thích quyết định của mô hình, đặc biệt đối với bài toán phân loại mã độc, việc xác định lý do tại sao một tệp tin được coi là độc hại có thể khó khăn.

Đòi hỏi tài nguyên tính toán nhiều: NN có thể đòi hỏi lượng tính toán lớn và có thể cần phần cứng mạnh để huấn luyện và triển khai.

**Áp dụng thuật toán Neural Network vào giải bài toán phát hiện mã độc:**

Trong bài toán phát hiện mã độc, NN hoạt động bằng cách sử dụng một kiến trúc mạng nơ-ron để học cách phân loại các tệp tin hoặc đoạn mã thành hai lớp: mã độc và không mã độc. Mô hình NN cần được huấn luyện trên một tập dữ liệu huấn luyện gồm đặc trưng của các tệp tin hoặc đoạn mã, chẳng hạn như thông tin về cấu trúc, hàm, biến, chuỗi, dòng lệnh, và nhiều đặc trưng khác.

Sau khi mô hình NN đã được huấn luyện, nó có thể được sử dụng để dự đoán xem một tệp tin hoặc đoạn mã mới có phải là mã độc hay không. Kết quả dự đoán của mô hình sẽ giúp xác định tính độc hại của tệp tin hoặc đoạn mã và có thể được sử dụng để đưa ra quyết định về việc chặn hoặc xử lý chúng.

## **4.2. Thuật toán Support Vector Classifier(SVC).**

Support Vector Classifier (SVC) là một thuật toán học máy được sử dụng trong bài toán phân loại. Nó dựa trên ý tưởng của Support Vector Machine (SVM) và được sử dụng để tìm ra một đường phân chia tối ưu giữa các lớp dữ liệu.

**Nguyên lý hoạt động của SVC:**

Thuật toán SVC hoạt động bằng cách tìm ra một siêu phẳng trong không gian đặc trưng, có thể phân tách các điểm dữ liệu của các lớp khác nhau. Siêu phẳng này được chọn sao cho khoảng cách từ các điểm dữ liệu gần nhất đến siêu phẳng là lớn nhất, đây được gọi là Margin. Các điểm dữ liệu gần nhất này được gọi là Support Vectors.

SVC có thể được áp dụng cho các bài toán phân loại nhị phân và đa lớp. Đối với bài toán phân loại đa lớp, SVC sử dụng phương pháp One-vs-One hoặc One-vs-Rest để chuyển đổi bài toán thành các bài toán phân loại nhị phân.

*Một số ưu điểm của thuật toán SVC bao gồm:*

Hiệu suất tốt trong việc phân loại dữ liệu tuyến tính và phi tuyến tính.

Khả năng xử lý các tập dữ liệu lớn.

*Tuy nhiên, SVC có một số hạn chế như:*

Yêu cầu các điểm dữ liệu phải được phân loại chính xác.

Nhạy cảm với nhiễu và dữ liệu không cân bằng.

Đòi hỏi thời gian huấn luyện lâu đối với các tập dữ liệu lớn.

**Áp dụng thuật toán vào giải bài toán phát hiện mã độc.**

Trong bài toán phát hiện mã độc, SVC hoạt động bằng cách tìm ra một siêu mặt phân chia tối ưu để phân loại các tệp tin hoặc đoạn mã thành hai lớp: mã độc và không mã độc.

Đầu tiên, ta cần xây dựng một tập dữ liệu huấn luyện gồm đặc trưng của các tệp tin hoặc đoạn mã này cần được trích xuất, ví dụ như thông tin về cấu trúc, hàm, biến, chuỗi, dòng lệnh, v.v.

Sau đó, SVC sẽ tìm cách tạo ra một siêu mặt phân chia trong không gian đặc trưng sao cho các điểm dữ liệu thuộc lớp mã độc và không mã độc được phân tách xa nhau nhất có thể. Siêu mặt phân chia này được xác định bởi một tập hỗ trợ (support vectors), đó là các điểm dữ liệu gần nhất với siêu mặt phân chia.

Khi có một tệp tin hoặc đoạn mã mới cần kiểm tra, SVC sẽ sử dụng siêu mặt phân chia đã học để dự đoán xem nó có phải là mã độc hay không. Nếu tệp tin hoặc đoạn mã nằm về phía dương của siêu mặt phân chia, nó sẽ được dự đoán là mã độc. Ngược lại, nếu nó nằm về phía âm của siêu mặt phân chia, nó sẽ được dự đoán là không mã độc.

## **4.3. Thuật toán thuật toán di truyền(genetic algorithm-GA).**

Mục tiêu chính của thuật toán di truyền trong dự án này là tối ưu hóa lựa chọn đặc trưng (features) từ một tập dữ liệu đa dạng của các ứng dụng Android. Thuật toán di truyền sẽ tạo ra các "chromosomes" (các tập con của các đặc trưng) và đánh giá tính thích nghi của chúng dựa trên khả năng phân loại tính độc hại của các ứng dụng.

Thuật toán di truyền giúp xác định các đặc trưng quan trọng nhất để phân loại ứng dụng là độc hại hoặc không độc hại. Điều này giúp cải thiện hiệu suất của mô hình phân loại bằng cách tạo ra các "chromosomes" chỉ chứa các đặc trưng quan trọng nhất.

Sau khi đã xác định các đặc trưng quan trọng, thuật toán di truyền sẽ sử dụng chúng để phân loại các ứng dụng. Sự phân loại dựa trên mô hình máy học (Neural Network hoặc Support Vector Classifier) và các đặc trưng đã được chọn. Việc tối ưu hóa lựa chọn đặc trưng cũng có thể ảnh hưởng đến hiệu suất mô hình máy học. Mô hình máy học sẽ được cung cấp với các đặc trưng quan trọng hơn và có khả năng phân loại ứng dụng tốt hơn.

Đây có thể xem như thuật toán tiền xử lý việc lựa chọn các thuộc tính nào sẽ xem xét trong quá trình áp dụng NN, SVC và các thuật toán học máy mở rộng khác.

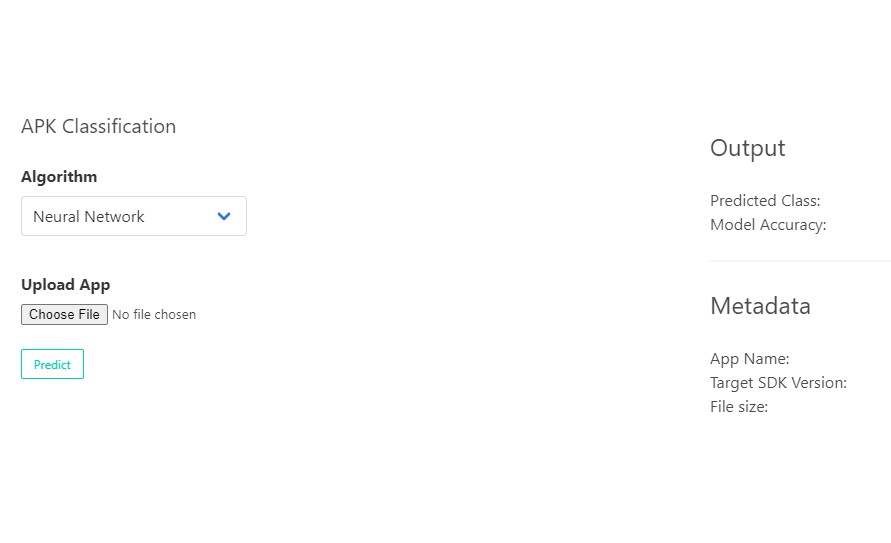
Đây có thể xem như thuật toán tiền xử lý việc lựa chọn các thuộc tính nào sẽ xem xét trong quá trình áp dụng NN, SVC và các thuật toán học máy mở rộng khác.

# **CHƯƠNG 5. THỰC NGHIỆM**

## **5.1. Tổng quan.**

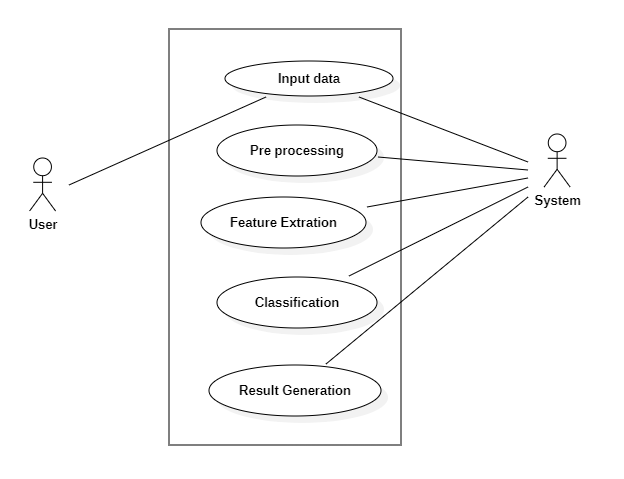
Thực nghiệm bao gồm tìm hiểu một số thuật toán như ANN, SVC,... áp dụng vào bài toán tạo ra một chương trình có khả năng đọc thông tin của các tệp tin định dạng .apk, phân tích thông tin và phân loại tệp tin cài đặt có chứa mã độc hay không. Cụ thể hóa bằng việc tạo một ứng dụng Fask Web hiển thị giao diện người dùng.

Ứng dụng sử dụng machine learning để xây dựng một hệ thống phát hiện mã độc. Nó sử dụng đặc trưng từ các tệp tin mã độc và không phải mã độc để huấn luyện mô hình dự đoán. Sau đó, áp dụng cụ thể hai thuật toán Artificial Neural Network(ANN) và Support Vector Machine(SVC). Các mô hình được huấn luyện bở dữ liệu có sẵn về các trường hợp thông tin quan trọng của một file apk và được sử dụng để phát hiện mã độc trong các tệp tin apk nhập vào.



Hình 8. Giao diện cơ bản

## **5.2. Phân tích thiết kế hệ thống**



Hình 9. Mô hình usecase

Bảng mô tả chi tiết các Usecase của hệ thống.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên UseCase | Mô tả chức năng | Tác nhân liên quan |
| 1 | Nhập dữ liệu | Người dùng cung cấp tệp tin apk cần phân tích | Người dùng tải lên tệp tin cần phân tích  Hệ thống tiếp nhận và kiểm tra định dạng tệp tin |
| 2 | Tiền xử lí dữ liệu | Hệ thống làm sạch dữ liệu, loại bảo các thông tin không cần thiết và chuẩn bị tệp tin để phân tích đặc trưng | Hệ thống thực hiện tự động các bước tiền xử lí |
| 3 | Trích xuất đặc trưng | Hệ thống trích xuất các đặc trưng quan trọng từ tệp tin, các đặc trưng này chuyển đổi thành vector số để sử dụng trong quá trình phân loại | Hệ thống phân tích và chuyển đổi dữ liệu thành đặc trưng số |
| 4 | Phân loại | Sử dụng học các mô hình học máy (ANN hoặc SVC) đã được huấn luyện để phân tích tập tin: Dự đoán tệp tin có chứa mã độc hay không và đưa ra kết quả với độ chính xác dựa trên mô hình | Thực hiện phân loại tệp tin bằng mô hình học máy |
| 5 | Tạo kết quả | Hệ thống hiển thị kết quả cuối cùng cho người dùng: kết luận tệp tin có chứa mã độc hay không, thông tin. | Hệ thống trả kết quả dự đoán  Người dùng xem kết quả qua giao diện |

**Mô tả hệ thống**

**a. Nhập dữ liệu (Input data)**

Người dùng tải lên các tệp tin apk cần kiểm tra.

Hệ thống kiểm tra định dạng tệp và lưu trữ tạm thời để xử lí

**b. Tiền xử lý dữ liệu (Pre-processing)**

Dữ liệu đầu vào được làm sạch và chuẩn bị để phân tích

Các tệp tin không hợp lệ hoặc không cần thiết sẽ bị loại bỏ

Hệ thống giải nén các tệp apk để lấy dữ liệu cần phần tích (như quyền truy cập, mã người bytecode)

**c. Trích xuất đặc trưng (Feature Extraction)**

Hệ thống phân tích và trích xuất đặc trưng quan trọng từ tệp tin apk:

Quyền truy cập (Permissions): Các quyền mà ứng dụng yêu cầu (như quyền truy cập tin nhắn, vị trí, camera).

Chuỗi kí tự (Strings): Các chuỗi nghi ngờ chứa mã độc thường xuất hiện trong mã nguồn.

Mã nguồn (Bytecode): Các thông tin liên quan đến hành vi của ứng dụng.

Các đặc trưng được chuyển đổi thành định dạng số (vector số) để sử dụng trong mô hình học máy.

**d. Phân loại (Classification)**

Sử dụng các thuật toán học máy cụ thể là ANN và SVC.

Kết quả dự đoán được đưa ra dựa tên dữ liệu đã huấn luyện

**e. Tạo kết quả (Result Generation)**

Hệ thống trả về kết quả phân tích cho người dùng, bao gồm:

Kết luận: Tệp tin là mã độc hoặc không

Thông tin chi tiết: Các đặc trưng nghi ngờ, mức độ rủi ro, hoặc loại mã độc (nếu có).

## **5.3. Cấu trúc**

**a.File input.txt**

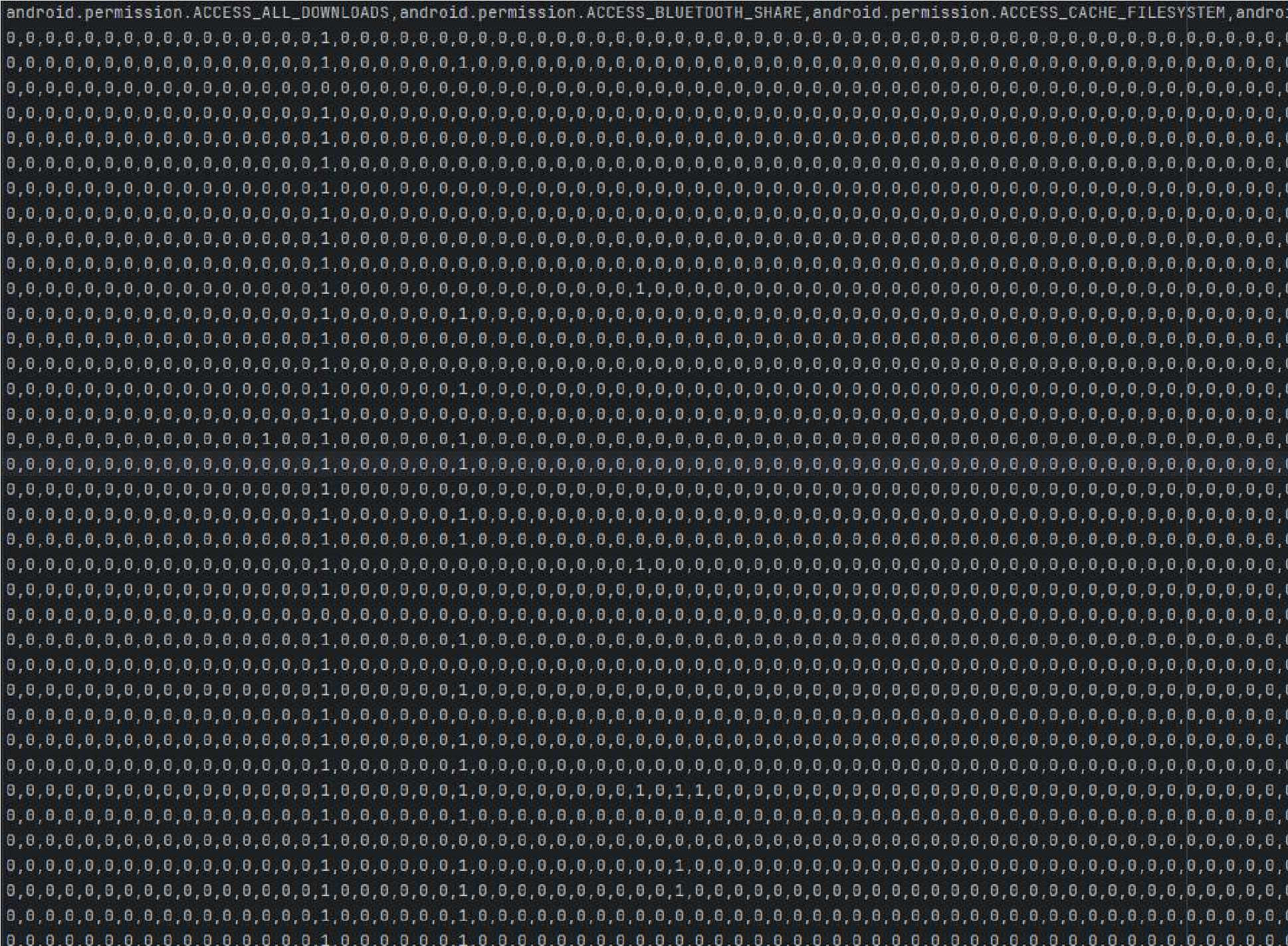
Cấu trúc input.txt, flie dữ liệu để dạy hay nạp dữ liệu về mã trong android có địn dạng file .txt, với rất nhiều trường thông tin của file, đại diện cho cấu hình của một mã hợp lệ hay không hợp lệ.

Các quyền này đóng vai tròn quan trọng trong việc kiểm soát truy cập và bảo mật của ứng dụng Android. Khi ài đặt một ứng dụng, hệ thống sẽ thông báo cho bạn có thể chấp nhận hoặc từ chối chúng ta. Ta dựa vào các thông tin này để phân loại ứng dụng có chứa mã độc hay không.

*Ví dụ một số thông tin cơ bản:*

* android.permission.ACCESS\_NETWORK\_STATE: Được sử dụng để kiểm tra tình trạng kết nối mạng.
* android.permission.INTERNET: Cho phép ứng dụng truy cập internet.
* android.permission.CAMERA: Quyền truy cập máy ảnh để chụp ảnh hoặc quay video.
* android.permission.RECORD\_AUDIO: Được sử dụng để ghi âm bằng microphone.
* android.permission.READ\_CONTACTS: Cho phép ứng dụng đọc thông tin liên lạc từ danh bạ.
* android.permission.ACCESS\_FINE\_LOCATION và android.permission.ACCESS\_COARSE\_LOCATION: Quyền truy cập vị trí cụ thể của thiết bị, dựa vào độ chính xác.
* android.permission.SEND\_SMS: Cho phép gửi tin nhắn SMS từ ứng dụng.
* android.permission.READ\_EXTERNAL\_STORAGE và android.permission.WRITE\_EXTERNAL\_STORAGE: Quyền truy cập và ghi dữ liệu vào bộ nhớ ngoại vi của thiết bị. o android.permission.BLUETOOTH và android.permission.BLUETOOTH\_ADMIN: Được sử dụng để quản lý Bluetooth và kết nối với thiết bị Bluetooth khác.
* android.permission.BATTERY\_STATS: Cho phép ứng dụng truy cập thông tin về tình trạng pin và tiêu hao năng lượng.
* android.permission.SYSTEM\_OVERLAY\_WINDOW: Cho phép vẽ lên cửa sổ của hệ thống, thường được sử dụng cho các ứng dụng đòi hỏi lớp vẽ đặc biệt.
* android.permission.READ\_SMS và android.permission.RECEIVE\_SMS: Cho phép đọc và nhận tin nhắn SMS.
* android.permission.READ\_PHONE\_STATE: Cho phép truy cập thông tin về trạng thái điện thoại, chẳng hạn như số IMEI.
* android.permission.WRITE\_SETTINGS: Quyền sửa đổi cài đặt hệ thống.
* android.permission.WAKE\_LOCK: Cho phép ứng dụng duy trì việc hoạt động của thiết bị khi màn hình tắt.
* android.permission.RECEIVE\_BOOT\_COMPLETED: Cho phép ứng dụng nhận sự kiện khởi động lại hệ thống.
* android.permission.RECEIVE\_WAP\_PUSH: Cho phép ứng dụng nhận các tin nhắn WAP Push, thường được sử dụng cho các tin nhắn đa phương tiện.
* android.permission.INSTALL\_PACKAGES: Cho phép ứng dụng cài đặt gói ứng dụng.

*Đối với những thuật toán khác nhau thì File dữ liệu ban đầu máy nhận vào để học không khác nhau, dữ liệu vào chỉ là tên của các trường thông tin đại diện cho các quyền của một tệp APK, việc các thuật toán đọc và xử lý file APK nhập vào mới có sự khác biệt.*



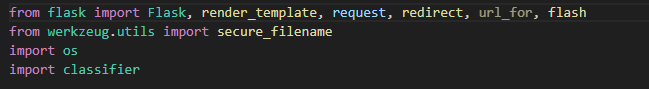
Hình 10. Một ví dụ về thông tin đầu vào của một tệp apk

**Cấu trúc source code**

Thực nghiệm này là một ứng dụng web Flask được sử dùng để phân loại các tệp apk (ứng dụng di động) là phần mềm độc hại (malware) hoặc không độc hại (benign). Ứng dụng cho phép người dùng tải lên một tệp apk và sau đó sử dụng hai thuật toán khác nhau để phân loại nó. Các thành phần chính bao gồm:

**File 1 (app.py):** Đây là tệp chính của ứng dụng Flask. Dưới đây là mô tả các thành phần quan trọng trong tệp này:

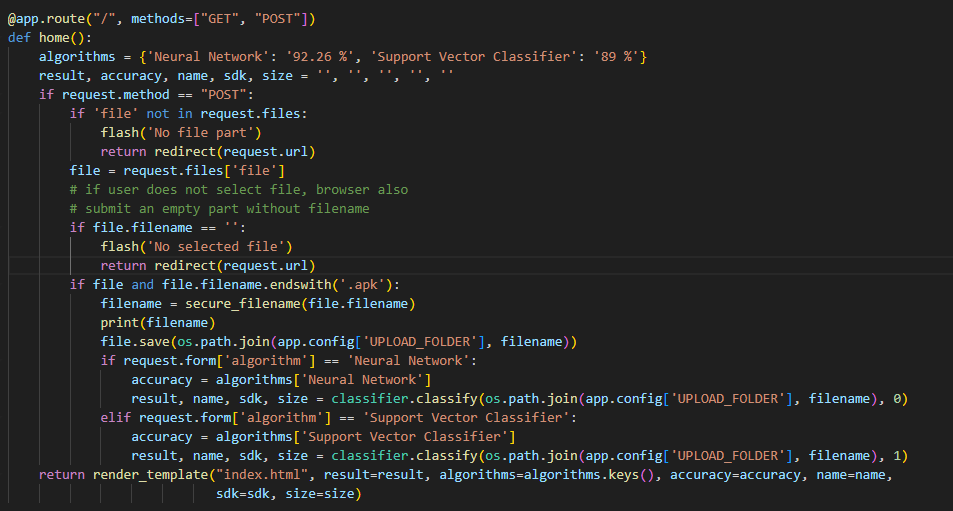
Import các thư viện Flask, Werkzeug, os và classifier



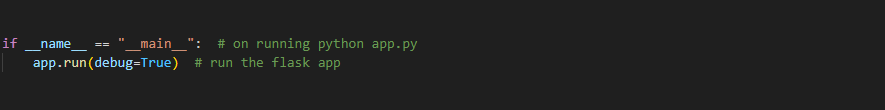
Cấu hình ứng dụng Flask với một thư mục lưu trữ tải lên và một mã bảo mật.



Xử lý tải lên tệp apk từ người dùng và thực hiện phân loại sử dụng các thuật toán máy học.

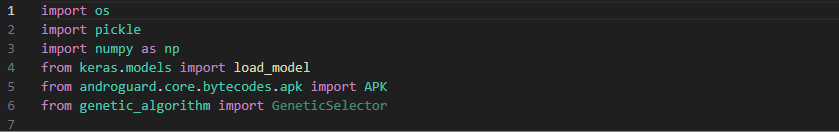


Trả về kết quả phân loại và hiện thị trang web kết quả và các tùy chọn thuật toán

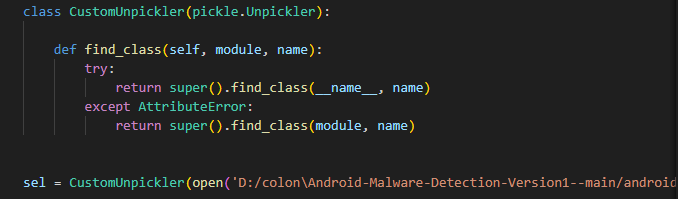


**File 2 (classifier.py):** Đây là một tệp chứa mã máy học để phân loại tệp apk. Các thành phần quan trọng trong tệp này bao gồm:

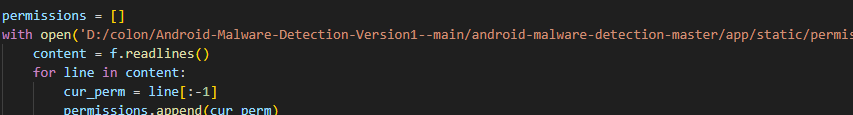
Sử dụng thư viện Keras để mô tải mô hình mạng nơ-ron (Neural Network) đã được cài sẵn.



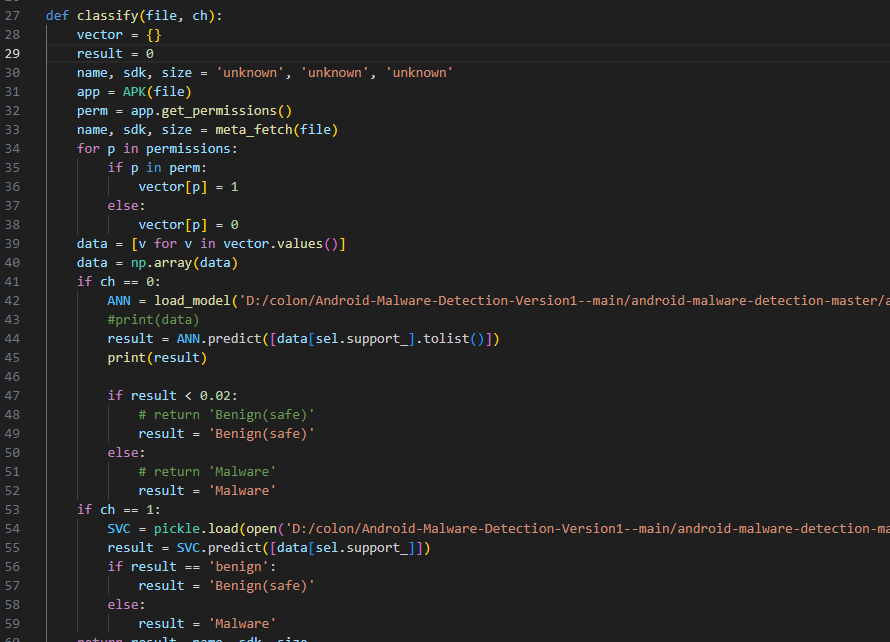
Sử dụng tệp ‘ga.pkl’ để tải một bộ lựa chọn đặc trưng (feature selection) từ thuật toán tối ưu di truyền.



Xác định danh sách quyền hạn (permissions) trong tệp permissions.txt

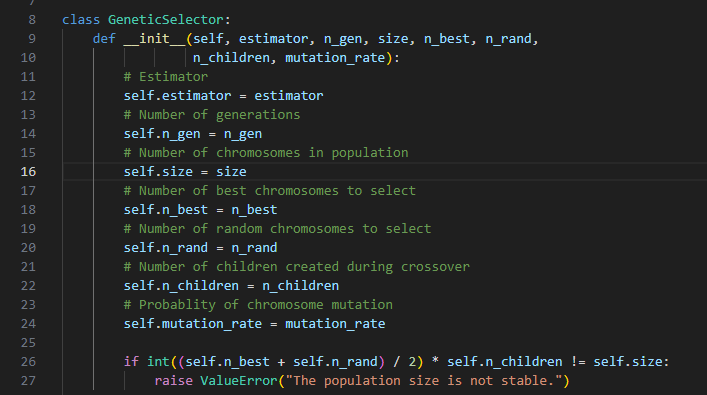


Sử dụng dữ liệu từ tệp apk để tạo một vector đặc trưng và sau đó sử dụng mô hình học máy để phân loại tệp apk.

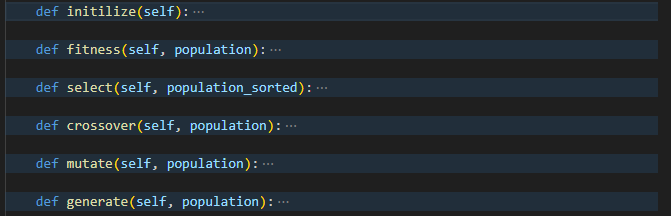


**File 3(genetic\_algorithm.py):** Đây là một tệp chứa mã máy học liên quan đến thauajt toán tối ưu hóa di truyền (genetic algorithm) được sử dụng để lựa chọn các đặc trưng quan trọng từ dữ liệu đầu vào.

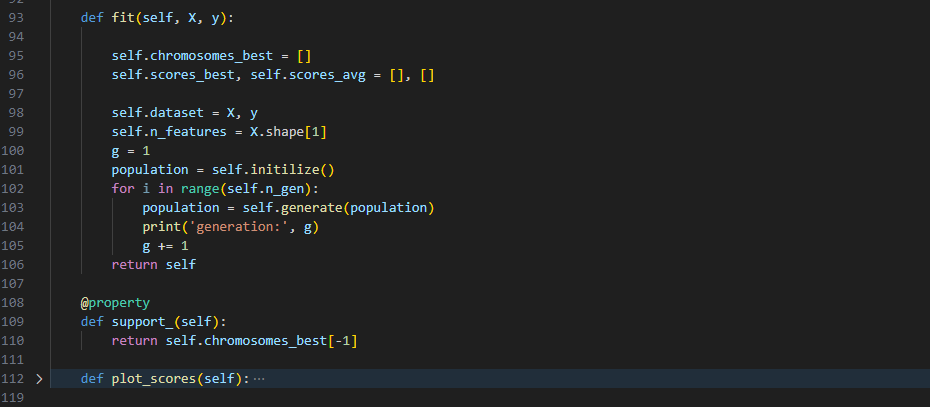
Tạo một lớp GeneticSelector để thực hiện việc lựa chọn đặc trưng sử dụng thuật toán di truyền.



Cung cấp các thông số như số thế hệ, kích thước, tỷ lệ số lượng tốt nhất, ngẫu nhiễn, và các tham số của di truyền khác. Mỗi thông tin được khởi tạo bởi những hàm riêng lẻ cũng như tổng thể, thuận lời cho việc thao tác sử dụng



Cung cấp phương thức fit để thực hiện quá trình tối ưu hóa di truyền trên dữ liệu đầu vào.



# **KẾT LUẬN**

Trong thời đại công nghệ phát triển vượt bậc, các thiết bị Android đã trở thành mục tiêu hấp dẫn cho các cuộc tấn công bằng mã độc do sự phổ biến và tính mở của nền tảng này. Việc tìm hiểu các vấn đề liên quan đến mã độc trên Android không chỉ giúp nâng cao nhận thức về an ninh mạng mà còn đóng vai trò quan trọng trong việc xây dựng các giải pháp bảo mật hiệu quả.

Qua đề tài, chúng ta đã làm rõ các khía cạnh quan trọng sau:

1. **Phân loại mã độc trên Android:** Mã độc có thể tồn tại dưới nhiều hình thức khác nhau gây ra những rủi ro nghiêm trọng như đánh cắp thông tin cá nhân, phá hoại hệ thống, hoặc gây tổn thất tài chính.
2. **Phương pháp tấn công phổ biến:** Hacker thường lợi dụng các lỗ hổng trong hệ điều hành Android, các ứng dụng bên thứ ba hoặc kỹ thuật xã hội để phát tán mã độc.

Bên cạnh đó, đề tài cũng tập trung vào việc ứng dụng học máy để phát hiện mã độc trong các tệp tin APK. Cụ thể:

* Việc sử dụng các thuật toán học máy như Artificial Neural Network (ANN) và Support Vector Machine (SVM) đã cho thấy tiềm năng mạnh mẽ trong việc phân tích và phát hiện mã độc với độ chính xác cao.
* Việc trích xuất và phân tích các đặc trưng từ tệp tin APK đóng vai trò quan trọng trong quá trình xây dựng mô hình học máy.

Nhìn chung, việc kết hợp giữa hiểu biết về mã độc và các công nghệ tiên tiến như học máy mở ra cơ hội lớn trong việc xây dựng các hệ thống bảo mật tự động, hiệu quả và đáng tin cậy. Tuy nhiên, các giải pháp hiện tại cũng đối mặt với nhiều thách thức như tốc độ phát triển của các biến thể mã độc mới, hạn chế trong dữ liệu huấn luyện, và nhu cầu cân bằng giữa hiệu suất và độ chính xác. Điều này đặt ra yêu cầu cần tiếp tục nghiên cứu và phát triển các phương pháp mới để ứng phó với các mối đe dọa ngày càng tinh vi.

Cuối cùng, đề tài không chỉ góp phần nâng cao nhận thức về an ninh trên nền tảng Android mà còn là nền tảng để nghiên cứu sâu hơn về việc phát hiện và ngăn chặn mã độc, hướng tới một môi trường sử dụng thiết bị di động an toàn hơn.

# **Tài liệu tham khảo**

Giáo trình môn học “Mã độc” của Học viện Kỹ thuật mật mã.

A Systematic Overview of Android Malware Detection

Android Malware Detection Using Static Analysis, Machine Learning and Deep Learning

# **Phụ lục**

**File app.py:**

from flask import Flask, render\_template, request, redirect, url\_for, flash

from werkzeug.utils import secure\_filename

import os

import classifier

app = Flask(\_\_name\_\_)

app.config['UPLOAD\_FOLDER'] = './static/upload/'

app.config['SECRET\_KEY'] = 'd3Y5d5nJkU6CdwY'

if os.path.exists(app.config['UPLOAD\_FOLDER']):

print("directory exists")

else:

os.makedirs(app.config['UPLOAD\_FOLDER'])

print("directory created")

@app.route("/", methods=["GET", "POST"])

def home():

algorithms = {'Neural Network': '92.26 %', 'Support Vector Classifier': '89 %'}

result, accuracy, name, sdk, size = '', '', '', '', ''

if request.method == "POST":

if 'file' not in request.files:

flash('No file part')

return redirect(request.url)

file = request.files['file']

# if user does not select file, browser also

# submit an empty part without filename

if file.filename == '':

flash('No selected file')

return redirect(request.url)

if file and file.filename.endswith('.apk'):

filename = secure\_filename(file.filename)

print(filename)

file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))

if request.form['algorithm'] == 'Neural Network':

accuracy = algorithms['Neural Network']

result, name, sdk, size = classifier.classify(os.path.join(app.config['UPLOAD\_FOLDER'], filename), 0)

elif request.form['algorithm'] == 'Support Vector Classifier':

accuracy = algorithms['Support Vector Classifier']

result, name, sdk, size = classifier.classify(os.path.join(app.config['UPLOAD\_FOLDER'], filename), 1)

return render\_template("index.html", result=result, algorithms=algorithms.keys(), accuracy=accuracy, name=name,

sdk=sdk, size=size)

if \_\_name\_\_ == "\_\_main\_\_": # on running python app.py

app.run(debug=True) # run the flask app

**File classifier.py**

import os

import pickle

import numpy as np

from keras.models import load\_model

from androguard.core.bytecodes.apk import APK

from genetic\_algorithm import GeneticSelector

class CustomUnpickler(pickle.Unpickler):

def find\_class(self, module, name):

try:

return super().find\_class(\_\_name\_\_, name)

except AttributeError:

return super().find\_class(module, name)

sel = CustomUnpickler(open('D:/colon\Android-Malware-Detection-Version1--main/android-malware-detection-master/app/static/models/ga.pkl', 'rb')).load()

permissions = []

with open('D:/colon/Android-Malware-Detection-Version1--main/android-malware-detection-master/app/static/permissions.txt', 'r') as f:

content = f.readlines()

for line in content:

cur\_perm = line[:-1]

permissions.append(cur\_perm)

def classify(file, ch):

vector = {}

result = 0

name, sdk, size = 'unknown', 'unknown', 'unknown'

app = APK(file)

perm = app.get\_permissions()

name, sdk, size = meta\_fetch(file)

for p in permissions:

if p in perm:

vector[p] = 1

else:

vector[p] = 0

data = [v for v in vector.values()]

data = np.array(data)

if ch == 0:

ANN = load\_model('D:/colon/Android-Malware-Detection-Version1--main/android-malware-detection-master/app/static/models/ANN.h5')

#print(data)

result = ANN.predict([data[sel.support\_].tolist()])

print(result)

if result < 0.02:

# return 'Benign(safe)'

result = 'Benign(safe)'

else:

# return 'Malware'

result = 'Malware'

if ch == 1:

SVC = pickle.load(open('D:/colon/Android-Malware-Detection-Version1--main/android-malware-detection-master/app/static/models/svc\_ga.pkl', 'rb'))

result = SVC.predict([data[sel.support\_]])

if result == 'benign':

result = 'Benign(safe)'

else:

result = 'Malware'

return result, name, sdk, size

def meta\_fetch(apk):

app = APK(apk)

return app.get\_app\_name(), app.get\_target\_sdk\_version(), str(round(os.stat(apk).st\_size / (1024 \* 1024), 2)) + ' MB'

File genetic\_algorithm.py

import os

import pickle

import numpy as np

from keras.models import load\_model

from androguard.core.bytecodes.apk import APK

from genetic\_algorithm import GeneticSelector

class CustomUnpickler(pickle.Unpickler):

def find\_class(self, module, name):

try:

return super().find\_class(\_\_name\_\_, name)

except AttributeError:

return super().find\_class(module, name)

sel = CustomUnpickler(open('D:/colon\Android-Malware-Detection-Version1--main/android-malware-detection-master/app/static/models/ga.pkl', 'rb')).load()

permissions = []

with open('D:/colon/Android-Malware-Detection-Version1--main/android-malware-detection-master/app/static/permissions.txt', 'r') as f:

content = f.readlines()

for line in content:

cur\_perm = line[:-1]

permissions.append(cur\_perm)

def classify(file, ch):

vector = {}

result = 0

name, sdk, size = 'unknown', 'unknown', 'unknown'

app = APK(file)

perm = app.get\_permissions()

name, sdk, size = meta\_fetch(file)

for p in permissions:

if p in perm:

vector[p] = 1

else:

vector[p] = 0

data = [v for v in vector.values()]

data = np.array(data)

if ch == 0:

ANN = load\_model('D:/colon/Android-Malware-Detection-Version1--main/android-malware-detection-master/app/static/models/ANN.h5')

#print(data)

result = ANN.predict([data[sel.support\_].tolist()])

print(result)

if result < 0.02:

# return 'Benign(safe)'

result = 'Benign(safe)'

else:

# return 'Malware'

result = 'Malware'

if ch == 1:

SVC = pickle.load(open('D:/colon/Android-Malware-Detection-Version1--main/android-malware-detection-master/app/static/models/svc\_ga.pkl', 'rb'))

result = SVC.predict([data[sel.support\_]])

if result == 'benign':

result = 'Benign(safe)'

else:

result = 'Malware'

return result, name, sdk, size

def meta\_fetch(apk):

app = APK(apk)

return app.get\_app\_name(), app.get\_target\_sdk\_version(), str(round(os.stat(apk).st\_size / (1024 \* 1024), 2)) + ' MB'