HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Đàm Tiến Đạt

PHƯƠNG PHÁP PHÁT HIỆN LỖ HỔNG MÃ NGUỒN DỰA TRÊN TẬP LUẬT

Chuyên ngành: Hệ thống thông tin

Mã số: 8.48.01.04

TÓM TẮT ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

HÀ NỘI - NĂM 2024

Đề án tốt nghiệp được hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học: PGS. TS. Đỗ Xuân Chợ

Phản biện 1: PGS.TS. Hoàng Xuân Dậu

Phản biện 2: PGS.TS. Nguyễn Long Giang

Đề án tốt nghiệp sẽ được bảo vệ trước Hội đồng chấm đề án tốt nghiệp thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: 10 giờ 00 ngày 20 tháng 03 năm 2024

Có thể tìm hiểu đề án tốt nghiệp tại:

- Thư viện của Học viện Công nghệ Bưu chính Viễn thông.

MỞ ĐẦU

1. Lý do chọn đề tài:

Trong thời đại số hóa hiện nay, bảo mật trở thành một ưu tiên hàng đầu. Sự bất an về lỗ hổng mã nguồn có thể dẫn đến việc rò rỉ dữ liệu, mất cơ hội kinh doanh và ảnh hưởng đến uy tín của tổ chức. Nghiên cứu về phương pháp phát hiện lỗ hổng mã nguồn có thể giúp cải thiện bảo mật.

Dựa trên tập luật để phát hiện lỗ hổng mã nguồn có thể là một hướng tiếp cận đáng quan tâm, đặc biệt khi có sẵn nhiều kiến thức về lỗ hổng mã nguồn trong tập luật. Nghiên cứu về cách áp dụng tập luật để tự động phát hiện lỗ hổng mã nguồn có thể giúp giảm thiểu công sức thủ công trong việc kiểm tra mã nguồn.

Nghiên cứu về phương pháp phát hiện lỗ hổng mã nguồn dựa trên tập luật có thể đóng góp vào nguồn kiến thức về bảo mật phần mềm và có thể được chia sẻ với cộng đồng bảo mật để cải thiện phương pháp phát hiện và khắc phục lỗ hổng mã nguồn. Việc chọn đề tài ***"Phát hiện lỗ hổng mã nguồn dựa trên tập luật"*** thông qua đề tài này sẽ là giải pháp tối ưu để phát hiện các lỗ hổng mã nguồn từ đó giúp nâng cao an toàn ứng dụng và an toàn dữ liệu của các phần mềm.

2. Tổng quan về vấn đề nghiên cứu

Hiện nay nhiều phần mềm ứng dụng cho phép người dùng có thể truy cập và xem được các thông tin thông qua internet. Các phần mềm ứng dụng này hầu hết vẫn tồn tại các lỗ hổng và dễ bị khai thác trước các cuộc tấn công qua internet. Dựa vào việc sử dụng tập luật và công cụ để tìm ra các vấn đề tiềm ẩn trong mã nguồn của phần mềm, như các lỗ hổng bảo mật, sai sót lập trình, hoặc các vấn đề liên quan đến chuẩn mã hóa. Nếu phần mềm ứng dụng không được bảo vệ và khắc phục các lỗ hổng một cách thích hợp và chuẩn xác, kẻ tấn công có thể lợi dụng để xâm nhập vào hệ thống, đánh cắp, làm mất hay phá hủy cơ sở dữ liệu của hệ thống hoặc bản thân chính phần mềm.

3. Mục tiêu nghiên cứu của đề tài

Mục tiêu nghiên cứu của đề tài là tìm hiểu, nghiên cứu giải pháp, công cụ để phát hiện các lỗ hổng của một phần mềm chứa các lỗi bảo mật.

4. Đối tượng và phạm vi nghiên cứu của đề tài

Đối tượng nghiên cứu của đề tài sẽ là công cụ hỗ trợ mã nguồn mở MobSF giúp phát hiện các lỗ hổng mã nguồn của phần mềm (cụ thể trong đề tài này là phần mềm Egov Quảng Nam phiên bản Android). Đề tài này tập trung nghiên cứu các phần sau:

- Nghiên cứu tổng quan về môi trường cài đặt phần mềm, các dạng tấn công và các lỗ hổng để khai thác và tấn công phổ biến.
- Tìm hiểu tập luật để rà quét và phát hiện lỗ hổng phần mềm.
- Nghiên cứu quá trình rà quét lỗ hổng bằng công cụ của một phần mềm

5. Phương pháp nghiên cứu của đề tài

- Phương pháp nghiên cứu: Kết hợp giữa lý thuyết và thực nghiệm thực tế rà quét và phát hiện các lỗ hổng mã nguồn và dựa trên tập luật bằng công cụ.

CHƯƠNG 1: TỔNG QUAN VỀ PHÁT HIỆN LỖ HỔNG MÃ NGUỒN DỰA TRÊN TẬP LUẬT

1.1. Tổng quan chung về lỗ hổng bảo mật và vấn đề phát hiện lỗ hổng bảo mật

1.1.1. Tổng quan về phát hiện và đánh giá lỗ hổng bảo mật

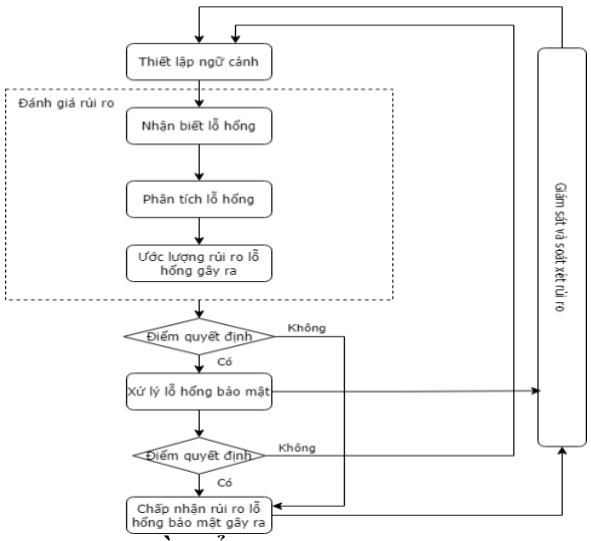
Tất cả các tài sản đều có giá trị quan trọng trong hoạt động của tổ chức và cần được bảo vệ. Do thông tin tồn tại và được lưu trữ dưới nhiều hình thức khác nhau, nên tổ chức phải có các biện pháp bảo vệ phù hợp để hạn chế rủi ro.

Bên cạnh những rủi ro về an toàn thông tin do bị tấn công phá hoại có chủ đích thông qua các lỗ hổng bảo mật, tổ chức cũng có thể gặp phải những rủi ro đối với thông tin, dữ liệu trong ứng dụng. Do đó, ngoài các biện pháp kỹ thuật hiện có, đơn vị phát triển cần nghiên cứu và áp dụng các công cụ phát hiện lỗ hổng phù hợp để giảm thiểu rủi ro.

1.1.2. Quy trình quản lý rủi ro

Khái niệm quản lý và xác định rủi ro các hệ thống của tổ chức được hiểu như là việc áp dụng các biện pháp xử lý, nhằm tiết giảm đầu tư vào nguồn lực, đảm bảo dự phòng, đánh giá và kiểm soát các rủi ro có thể. Với mục tiêu về các nghiệp vụ của tổ chức không bị ảnh hưởng, sai lệch. Kết quả mong muốn cho quản lý rủi ro là kiểm soát và điều hành tốt các hoạt động của tổ chức.

Sơ đồ các luồng :



Hình 1.1. Sơ đồ thể hiện quy trình quản lý rủi ro

Luồng thực hiện quy trình:

Bước 1: Thiết lập ngữ cảnh, đầu vào, đầu ra và công cụ rà quét lỗ hổng.

Bước 2: Nhận biết lỗ hổng nhằm xác định nguy cơ, các mối đe dọa tiềm ẩn, các điểm yếu đang tồn tại bên trong ứng dụng, có thể lợi dụng cho các mục đích xấu.

Bước 3: Phân tích rủi ro nhằm đánh giá tác động về an toàn bảo mật thông tin từ các nguy cơ đã nhận biết được ở Bước 2.

Bước 4: Ước lượng được rủi ro dựa trên các phân tích ở Bước 3 nhằm đánh giá mức độ rủi ro.

Bước 5: Từ các rủi ro đã phân tích thông qua các lỗ hổng đã phát hiện được xác định

các lỗ hổng có nguy cơ cao để quyết định xử lý lỗ hổng. Nếu nguy cơ cao sẽ được yêu cầu khắc phục và đánh giá lại sau khi đã khắc phục xong còn các nguy cơ còn lại sẽ được khắc phục sau.

Bước 6: Sau khi đã xử lý lỗ hổng nguy cơ cao sẽ tiếp tục được đánh giá để quyết định có chấp nhận được các rủi ro lỗ hổng gây ra. Nếu chấp nhận được thì phần mềm, ứng dụng sẽ được thông qua còn nếu không sẽ quay lại Bước 1 để đánh giá lại.

Tất cả các bước từ thiết lập ngữ cảnh đến đánh giá rủi ro và kết luận chấp nhận rủi ro đều được giám sát và soát xét kỹ lưỡng nhằm đảm bảo sự chặt chẽ và an toàn cho cả quy trình.

Thiết lập ngữ cảnh bao gồm đầu vào, đầu ra, phương án thực hiện của ứng dụng cần đánh giá:

Đầu vào: Đơn vị phát triển chuẩn bị tập tin đóng gói cài đặt cho thiết bị Android (.apk) bản cuối để đánh giá rủi ro an toàn thông tin.

Đầu ra: Báo cáo thống kê các cảnh báo về lỗ hổng bảo mật và giải trình khắc phục của đơn vị phát triển.

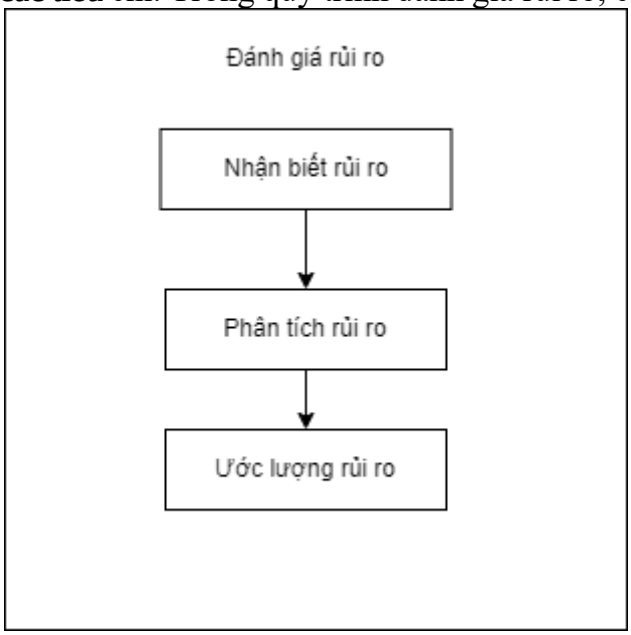
Phương án thực hiện: Thiết lập môi trường và công cụ rà quét lỗ hổng bảo mật sau đó từ báo cáo chi tiết của công cụ tổng hợp thành báo cáo các lỗ hổng cần khắc phục.

Nhằm xác định các nguy cơ có thể xảy ra với hệ thống, xác định các điểm yếu đang tồn tại, đánh giá mức độ tiềm ẩn, rủi ro có thể gặp phải. Từ đó, có thể phân loại và đánh giá mức độ theo thứ tự các rủi ro.

Đầu vào: Bao gồm tiêu chí, phạm vi, giới hạn thực hiện đánh giá rủi ro.

Hành động: Cần xác định định tính hoặc định lượng, sắp xếp các tiêu chí theo mức độ ưu tiên và các mục tiêu liên quan.

Đầu ra: Báo cáo tổng hợp danh sách những rủi ro và lỗ hổng bảo mật đã được sắp xếp theo thứ tự ưu tiên theo các tiêu chí. Trong quy trình đánh giá rủi ro, có ba hoạt động như sau:



Hình 1.2. Các bước đánh giá rủi ro

a) Bước 1: Nhận biết, xác định

Nhận biết rủi ro nhằm xác định nguy cơ, các mối đe dọa tiềm ẩn, các điểm yếu đang tồn tại bên trong ứng dụng, có thể lợi dụng cho các mục đích xấu. Từ đó xác định được phương thức, các mối đe dọa và tác động có thể ảnh hưởng thiệt hại đến tài sản, hệ thống của tổ chức.

Mục đích là xác định nguyên nhân có thể gây ra thiệt hại tiềm ẩn và hiểu được lí do, phương thức, thời điểm, không gian mà thiệt hại có thể xảy ra. Các hoạt động gồm:

b) Bước 2: Phân tích

Phân tích rủi ro nhằm xác định, đánh giá mức độ ảnh hưởng đến ứng dụng, xác định nguyên nhân và bản chất rủi ro. Xác định đánh giá tác động về an toàn bảo mật thông tin từ các nguy cơ đã dự báo từ trước. Từ đó đánh giá mức độ tác động đến tổ chức hoặc tài sản của tổ chức.

c) Bước 3: Ước lượng

Ước lượng được căn cứ trên các phân tích nhằm xác định mức độ rủi ro, đánh giá mức độ rủi ro có nguy cao hay không.

1.1.3. Đánh giá rủi ro

Thông qua phân tích hiện trạng tại thời điểm đánh giá của ứng dụng bao gồm thông tin ứng dụng, lỗ hổng bảo mật tồn tại, các biện pháp an toàn bảo mật đang dùng, từ đó xác định rủi ro và đánh giá ảnh hưởng của rủi ro tạo nên cho hệ thống, đồng thời đưa ra giải pháp nhằm giảm thiểu rủi ro.

Mục đích của đánh giá rủi ro là thông báo cho những người ra quyết định và hỗ trợ các phản ứng rủi ro bằng cách xác định:

- Các lỗ hổng cả bên trong ứng dụng.
- Tác động (tức là gây hại) cho các tổ chức có thể xảy ra do khả năng đe dọa khai thác lỗ hổng.
- Khả năng gây hại sẽ xảy ra.

1.1.4. Tại sao phải đánh giá rủi ro lỗ hổng bảo mật

Các doanh nghiệp, tổ chức ở mọi quy mô, lĩnh vực sản xuất, cơ quan quản lý nhà nước… đều có nguy cơ về tấn công mạng, đánh cắp dữ liệu, mã hóa, lộ thông tin khách hàng hoặc thông tin nội bộ, dẫn đến phải đối mặt với những rủi ro phức tạp và lớn hơn. Không những thế, việc khắc phục hậu quả sẽ rất tốn kém về nhân lực, thời gian, tiền bạc và uy tín của đơn vị. Vì vậy, tại sao phải đánh giá rủi ro lỗ hổng bảo mật mã nguồn gây ra? Có 2 lý do chính như sau: Xác định mức độ an ninh của ứng dụng, phòng ngừa giảm thiểu rủi ro từ sớm và đảm bảo an toàn trong tương lai của phần mềm.

- Có đánh giá tổng quan toàn bộ phần mềm, bao gồm các thành phần về mã nguồn.
- Đưa ra được các giải pháp khắc phục, nâng cấp và cập nhật các bản nâng cấp ứng dụng có nguy cơ.
- Xác định và đánh giá được các mức độ rủi ro về ứng dụng. Giúp đơn vị phát triển có thể quyết định nâng cấp trong tương lai.
- Ứng dụng hiện tại đã triển khai giải pháp đảm bảo an ninh nào, đánh giá tình hình và đề xuất triển khai nếu chưa đáp ứng được yêu cầu.
- Phân loại mức độ rủi ro của ứng dụng, từ đó xác định các mục tiêu chính cho việc phòng ngừa, giảm thiểu rủi ro trong tương lai.

1.2. Phương pháp phát hiện lỗ hổng dựa trên tập luật

1.2.1. Công cụ và kỹ thuật phát hiện lỗ hổng

Phát hiện lỗ hổng mã nguồn là một quá trình quan trọng để đảm bảo tính bảo mật của ứng dụng và hệ thống. Dưới đây là một số công cụ và kỹ thuật phổ biến để phát hiện lỗ hổng mã nguồn:

Các Công Cụ Phát Hiện Lỗ Hổng:

1. **SonarQube:** SonarQube là một công cụ mã nguồn mở giúp phát hiện lỗ hổng mã nguồn thông qua việc thực hiện kiểm tra tự động với nhiều ngôn ngữ lập trình.
2. **Checkmarx:** Checkmarx là một công cụ chuyên nghiệp được sử dụng để phân tích mã nguồn và phát hiện các lỗ hổng bảo mật.
3. **Fortify Static Code Analyzer:** Fortify của Micro Focus cung cấp một công cụ phân tích tĩnh mã nguồn để tìm kiếm và phát hiện các lỗ hổng bảo mật.
4. **Veracode:** Veracode là một dịch vụ kiểm tra mã nguồn tự động được tích hợp vào chuỗi phát triển liên tục để phát hiện và báo cáo lỗ hổng mã nguồn.
5. **Brakeman:** Brakeman là một công cụ dành cho ứng dụng Ruby on Rails để phát hiện lỗ hổng bảo mật trong mã nguồn Ruby.
6. **Bandit:** Bandit là một công cụ kiểm tra lỗ hổng bảo mật trong mã nguồn Python, tập trung vào việc tìm kiếm lỗ hổng liên quan đến an ninh mã nguồn.
7. **OWASP Dependency-Check:** OWASP Dependency-Check kiểm tra các phụ thuộc của ứng dụng để phát hiện các thư viện đã được công bố về lỗ hổng bảo mật.

Kỹ Thuật Phát Hiện Lỗ Hổng Mã Nguồn:

- Kiểm tra tĩnh (Static Analysis): Sử dụng công cụ để kiểm tra mã nguồn mà không thực hiện thực thi chương trình. Các công cụ này tìm kiếm lỗ hổng từ mã nguồn trước khi chạy ứng dụng.
- Kiểm tra động (Dynamic Analysis): Thực hiện kiểm tra lỗ hổng trong mã nguồn trong quá trình chạy ứng dụng. Các kỹ thuật này thường bao gồm penetration testing và fuzz testing.
- Kiểm tra tự động (Automated Testing): Sử dụng kịch bản kiểm thử tự động để kiểm tra mã nguồn một cách định kỳ và tự động, giúp phát hiện lỗ hổng một cách nhanh chóng.
- Kiểm tra bảo mật phần mềm bên thứ ba: Sử dụng dịch vụ của các công ty chuyên về bảo mật phần mềm để kiểm tra mã nguồn và đưa ra báo cáo chi tiết về các lỗ hổng có thể xảy ra.
- Kiểm tra quy mô lớn (Large-Scale Analysis): Sử dụng công nghệ big data và machine learning để phân tích lỗ hổng mã nguồn trên quy mô lớn, giúp xác định các xu hướng và mô hình bảo mật.

Việc sử dụng một sự kết hợp giữa các công cụ và kỹ thuật này thường là lựa chọn tốt nhất để đảm bảo mức độ bảo mật cao cho mã nguồn. Mỗi công cụ có những ưu điểm và hạn chế riêng, và việc kết hợp sử dụng nhiều công cụ có thể giúp đảm bảo tính toàn vẹn và an toàn của ứng dụng Android.

1.2.2. Mô tả và phân loại tập luật

Tập luật là tập hợp các quy tắc mà hệ thống hoặc mạng so sánh để quyết định nên làm gì hoặc hành động nào được chấp thuận.

Ví dụ:

- Tập luật tường lửa: cho phép hoặc từ chối một gói dữ liệu khi đi qua.
- Tập luật IDS: cảnh báo khi có xâm nhập bất hợp pháp tới hệ thống.
- Tập luật AV: xóa file khi xác định có chứa mã độc.

Rà soát đánh giá phần mềm nhằm:

- Tìm ra những lỗi hoặc lỗ hổng bảo mật: lỗ hổng bảo mật, các vi phạm chính sách.
- Tìm ra các thiếu sót hoặc không hiệu quả làm ảnh hưởng đến hiệu suất của phần mềm.

Tập luật để phát hiện lỗ hổng mã nguồn phần mềm thường được thiết kế để kiểm tra

mã nguồn của phần mềm và xác định các vấn đề bảo mật, lỗi lập trình, và các tiêu chí chất lượng mã khác. Dưới đây là một số chi tiết về các quy tắc thường được áp dụng để phát hiện lỗ hổng mã nguồn:

1. Quy tắc về Bảo mật:

- Kiểm tra xác thực và ủy quyền: Phát hiện mã nguồn không an toàn liên quan đến xác thực và ủy quyền.
- Kiểm tra xâm phạm bảo mật: Phát hiện các lỗ hổng bảo mật phổ biến như SQL injection, Cross-Site Scripting (XSS), và các kỹ thuật tấn công khác.

2. Quy tắc về Hiệu suất:

- Kiểm tra lợi dụng tài nguyên: Phát hiện mã nguồn có thể gây ra lợi dụng tài nguyên không cần thiết hoặc tiêu tốn quá mức.

3. Quy tắc về Độ tin cậy:

- Phòng tránh chống mã độc hại: Kiểm tra mã nguồn để phát hiện và ngăn chặn việc tích hợp mã độc hại.

4. Quy tắc về Kiểm soát biên:

- Kiểm tra kiểm soát biên: Phát hiện các lỗ hổng liên quan đến kiểm soát biên không an toàn.

5. Quy tắc về Tiêu chuẩn mã:

- Kiểm tra định dạng mã: Đảm bảo rằng mã nguồn tuân theo các quy tắc về định dạng để dễ đọc và duy trì.
- Kiểm tra sự hiện diện của mã chết: Loại bỏ đoạn mã không sử dụng để giảm kích thước của ứng dụng.

6. Quy tắc về Ngôn ngữ và API:

- Kiểm tra sử dụng API nhạy cảm: Đảm bảo rằng các API nhạy cảm được sử dụng đúng cách và an toàn.

7. Quy tắc về Giao diện người dùng (UI):

- Kiểm tra sử dụng tài nguyên giao diện: Đảm bảo rằng tài nguyên giao diện được sử dụng đúng cách và không có lỗi.

1.2.3. Khó khăn trong phát hiện lỗ hổng bảo mật

Với sự phát triển liên tục về công nghệ thông tin trên thế giới đã giúp các tổ chức, đơn vị giảm thiểu được sức người trong hầu hết các lĩnh vực. Nhưng bên cạnh đó, các tổ chức tội phạm công nghệ cao ngày càng tinh vi và nguy hiểm khi liên tục sử dụng công nghệ cao để sử dụng cho mục đích phá hoại, đánh cắp thông tin, gián điệp... Đòi hỏi các đơn vị cần phải tập trung đầu tư nhiều vào công tác đảm bảo an toàn thông tin của các phần mềm và ứng dụng. Để thực hiện được điều đó, cần phải thực hiện công tác thường xuyên dò quét, xác định các nguy cơ tiềm ẩn để có biện pháp khắc phục, tăng cường an toàn thông tin hệ thống và mã nguồn của các phần mềm, ứng dụng. Vì vậy, các tổ chức đơn vị cần phải có những đầu tư chi phí cho phát hiện các lỗ hổng bảo mật cho hệ thống của mình, trong đó bao gồm các chi phí về: Phần mềm, con người, quy trình ... những thách thức không nhỏ đối với các đơn vị khi gặp khó khăn trong việc lựa chọn nhà cung cấp về giải pháp đảm bảo an ninh hệ thống. Trong khi đó các tội phạm công nghệ cao ngày càng tinh vi, đòi hỏi các giải pháp dò quét cũng phải chủ động trong việc xác định các nguy cơ tiềm ẩn, hỗ trợ người đứng đầu đơn vị có thể ra những quyết định, chính sách phù hợp. Công nghệ không ngừng phát triển, việc quản lý rủi ro an toàn thông tin luôn gặp nhiều khó khăn vì các nguy cơ luôn có thể xảy ra, điều tốt nhất các đơn vị phát triển cần phải có những đầu tư nghiêm túc phù hợp với tình hình thực tế.

1.3. Kết luận chương 1

Các cuộc tấn công mạng, tội phạm công nghệ cao, gián điệp không ngừng gia tăng nhằm mục đích phá hoại, lấy cắp dữ liệu, bí mật của tổ chức doanh nghiệp và cả của nhà nước. Chúng có thể gây ra những hậu quả vô cùng nghiêm trọng đến các đơn vị, tổ chức ở mọi vùng miền, quốc gia, lãnh thổ trên thế giới. Trong thời gian tới, dự báo về tình hình an ninh thông tin có những diễn biến khó lường, đặc biệt là các tội phạm công nghệ cao sử dụng trí tuệ nhân tạo trong lây nhiễm mã độc, tấn công với mục đích xấu nhằm đánh cắp, mã hóa thông tin. Vì vậy, để có thể phòng ngừa, giảm thiểu rủi ro, đòi hỏi các tổ chức doanh nghiệp phải quan tâm đến công tác đầu tư các hệ thống dò quét lỗ hổng bảo mật dùng cho nhận biết, đánh giá, phân loại… Từ đó có thể có thể phát hiện các lỗ hổng, điểm yếu của ứng dụng, kết xuất báo cáo đánh giá chi tiết và mức độ nghiêm trọng phục vụ công tác đảm bảo an toàn bảo mật thông tin. Trên các cơ sở lý thuyết đó, chương tiếp theo sẽ trình bày chi tiết về bài toán đặt ra và hướng giải quyết.

CHƯƠNG 2: PHƯƠNG PHÁP PHÁT HIỆN LỖ HỔNG MÃ NGUỒN DỰA TRÊN TẬP LUẬT

2.1. Đánh giá MobSF với một số công cụ khác

Các công cụ rà quét lỗ hổng mã nguồn, bao gồm cả MobSF, thường có một số điểm chung và khác nhau. Dưới đây là một số điểm chung và khác nhau giữa MobSF và các công cụ rà quét lỗ hổng mã nguồn khác:

Điểm chung:

Phát hiện lỗ hổng bảo mật: Cả MobSF và các công cụ rà quét lỗ hổng mã nguồn khác đều được thiết kế để phát hiện các lỗ hổng bảo mật trong mã nguồn của ứng dụng, như lỗ hổng về quyền truy cập, mã độc, lỗ hổng mã nguồn mở, và các vấn đề khác liên quan đến bảo mật.

Hỗ trợ nhiều ngôn ngữ lập trình: Cả MobSF và các công cụ khác thường hỗ trợ nhiều ngôn ngữ lập trình phổ biến, bao gồm Java, Python, JavaScript, C/C++, và các ngôn ngữ khác. Một số công cụ chỉ hỗ trợ một loại ngôn ngữ lập trình như công cụ Brakeman chỉ hỗ trợ ngôn ngữ Ruby

Tính linh hoạt và mở rộng: Các công cụ này đều có tính linh hoạt và có thể được mở rộng thông qua các plugin hoặc tích hợp với các công cụ khác để tạo ra quy trình kiểm thử bảo mật tự động.

Điểm khác nhau:

Phạm vi hỗ trợ: MobSF thường chủ yếu tập trung vào ứng dụng di động (Android và iOS), trong khi các công cụ khác có thể hỗ trợ phân tích mã nguồn cho các loại ứng dụng khác nhau như web, desktop, và các hệ thống nhúng.

Tính năng chuyên sâu: Mỗi công cụ có các tính năng chuyên sâu riêng biệt, chẳng hạn như MobSF tập trung vào các vấn đề bảo mật cụ thể cho ứng dụng di động, trong khi các công cụ khác có thể tập trung vào kiểm tra chất lượng mã, phân tích mã nguồn mở, và nhiều hơn nữa.

Cộng đồng và hỗ trợ: Mức độ hỗ trợ từ cộng đồng và sự phát triển của các công cụ có thể khác nhau. Một số công cụ có cộng đồng lớn và tích cực hơn, trong khi các công cụ khác có thể có ít hỗ trợ hơn.

Giấy phép và chi phí: Các công cụ có thể được phân phối dưới các giấy phép khác nhau, bao gồm các phiên bản miễn phí và phiên bản trả phí với các tính năng mở rộng và hỗ

trợ cao cấp hơn.

Ngữ cảnh đầu vào: Đối với MobSF là tệp APK để rà quét còn các công cụ khác sẽ tích hợp trong các công cụ phát triển ứng dụng để rà quét trực tiếp mã nguồn trong quá trình phát triển.

Tóm lại, MobSF và các công cụ rà quét lỗ hổng mã nguồn khác có nhiều điểm chung trong việc phát hiện lỗ hổng bảo mật, nhưng cũng có những khác biệt quan trọng về phạm vi hỗ trợ, tính năng, cộng đồng, và giấy phép. Lựa chọn giữa chúng phụ thuộc vào nhu cầu cụ thể của dự án và mục tiêu bảo mật. MobSF thường được sử dụng cho mục đích kiểm tra bảo mật cá nhân hoặc nhóm nhỏ hơn, và có thể yêu cầu kiến thức chuyên sâu hơn về bảo mật ứng dụng di động.

2.2. Tổng quan về công cụ phát hiện lỗ hổng mã nguồn

Công cụ phát hiện lỗ hổng mã nguồn là các ứng dụng được thiết kế để kiểm tra mã nguồn của phần mềm để xác định và báo cáo về các lỗ hổng bảo mật, tiềm ẩn, hoặc các vấn đề khác có thể ảnh hưởng đến tính toàn vẹn và an toàn của ứng dụng. Dưới đây là một tổng quan về công cụ phát hiện lỗ hổng mã nguồn:

Chức Năng Cơ Bản:

Phân Tích Tĩnh:

Kiểm tra lỗ hổng bảo mật: Các công cụ này kiểm tra mã nguồn để phát hiện và báo cáo về các lỗ hổng bảo mật như SQL injection, cross-site scripting (XSS), tràn bộ đệm, và nhiều vấn đề khác.

Phân tích quy ẩn: Các công cụ có thể phân tích các luồng điều khiển, dữ liệu và cấu trúc của ứng dụng để xác định lỗ hổng tiềm ẩn mà không cần chạy ứng dụng.

Tích Hợp Trong Quy Trình Phát Triển:

Tích hợp với ide: Nhiều công cụ có thể tích hợp trực tiếp vào môi trường phát triển tích hợp (IDE) như Eclipse, Visual Studio, hoặc IntelliJ IDEA để hỗ trợ lập trình viên phát hiện lỗ hổng ngay từ khi viết mã.

Tích hợp trong quy trình CI/CD: Các công cụ thường có khả năng tích hợp vào các hệ thống liên tục tích hợp và triển khai (CI/CD) để kiểm tra mã nguồn trong quy trình tự động.

Loại Công Cụ:

Tự Động Hóa:

Static Application Security Testing (SAST): Các công cụ SAST kiểm tra mã nguồn mà không chạy ứng dụng. Chúng phân tích tập tin mã nguồn và báo cáo về lỗ hổng mà chúng tìm thấy.

Dynamic Application Security Testing (DAST): Ngược lại với SAST, DAST chạy ứng dụng để phân tích lỗ hổng trong môi trường thực tế. Nó kiểm tra ứng dụng đang chạy và tìm kiếm lỗ hổng từ cấp độ runtime.

Ưu Điểm:

- Phát Hiện Sớm: Cung cấp khả năng phát hiện sớm các lỗ hổng khi vẫn ở cấp độ mã nguồn.
- Tự Động Hóa: Các công cụ thường có thể tự động hóa việc kiểm tra mã nguồn, giảm áp lực cho nhóm phát triển.
- Tích Hợp Liên Tục: Các công cụ thích hợp với quy trình liên tục tích hợp, giúp duy trì mã nguồn an toàn và bảo mật trong quá trình phát triển.

Nhược Điểm:

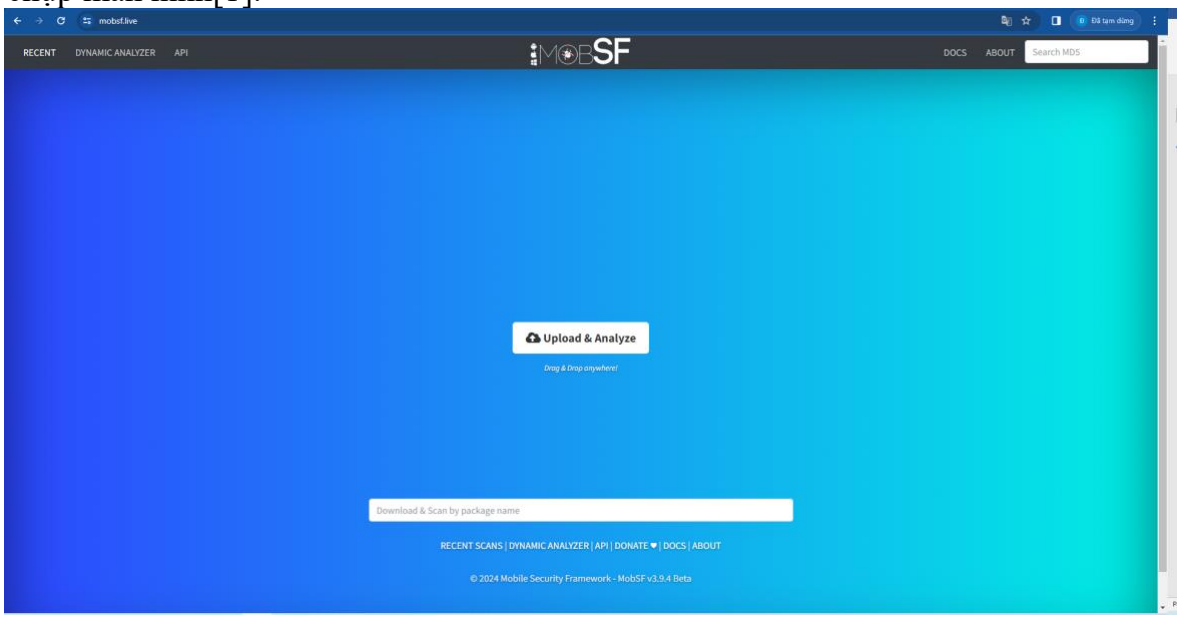
- False Positives: Có thể xuất hiện các kết quả giả mạo, làm tăng công việc kiểm tra và giảm độ chính xác.

- Phức Tạp Cho Người Mới: Sử dụng chúng có thể đòi hỏi sự hiểu biết sâu rộng về bảo mật và mã nguồn.
- Khả Năng Chấp Nhận: Các công cụ thường có khả năng chấp nhận tốt khi mã nguồn chưa được viết hoặc thiết kế đúng cách.

Việc sử dụng công cụ phát hiện lỗ hổng mã nguồn là một phần quan trọng trong chiến lược bảo mật phần mềm để đảm bảo rằng mã nguồn được viết và duy trì một cách an toàn và bảo mật.

2.3. Nghiên cứu về công cụ MobSF

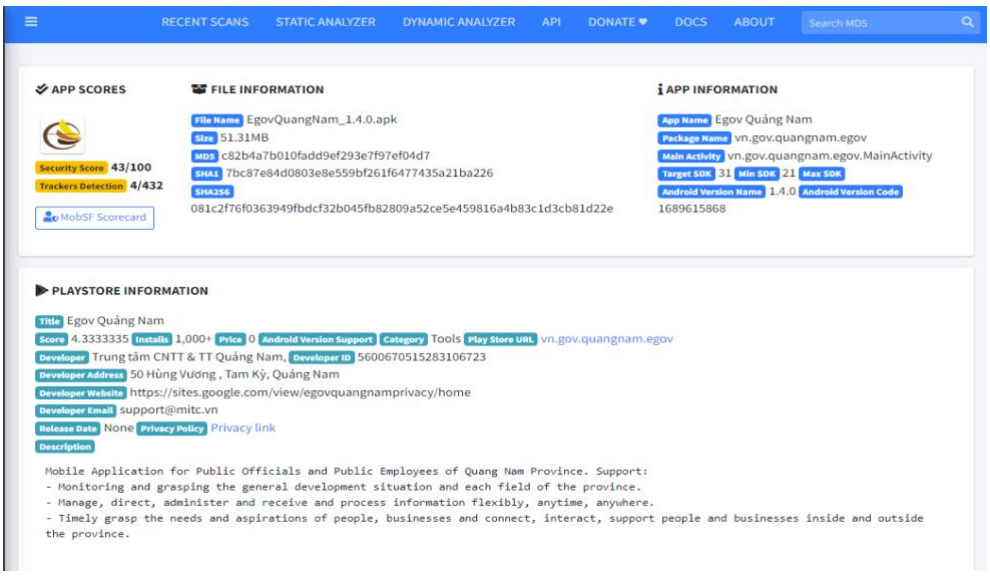
MobSF là một công cụ mã nguồn mở được phát triển bởi Ajin Abraham được sử dụng để phân tích tự động APK. Đây là tập hợp các công cụ chạy dưới một giao diện, thực hiện các tác vụ riêng lẻ như Jadx, apktool, v.v. và hiển thị kết quả dưới một giao diện chung. Các báo cáo này cũng có thể được tải xuống ở định dạng PDF và phân tích chi tiết cùng với các ảnh chụp màn hình[1].



Hình 2.1. Giao diện MobSF

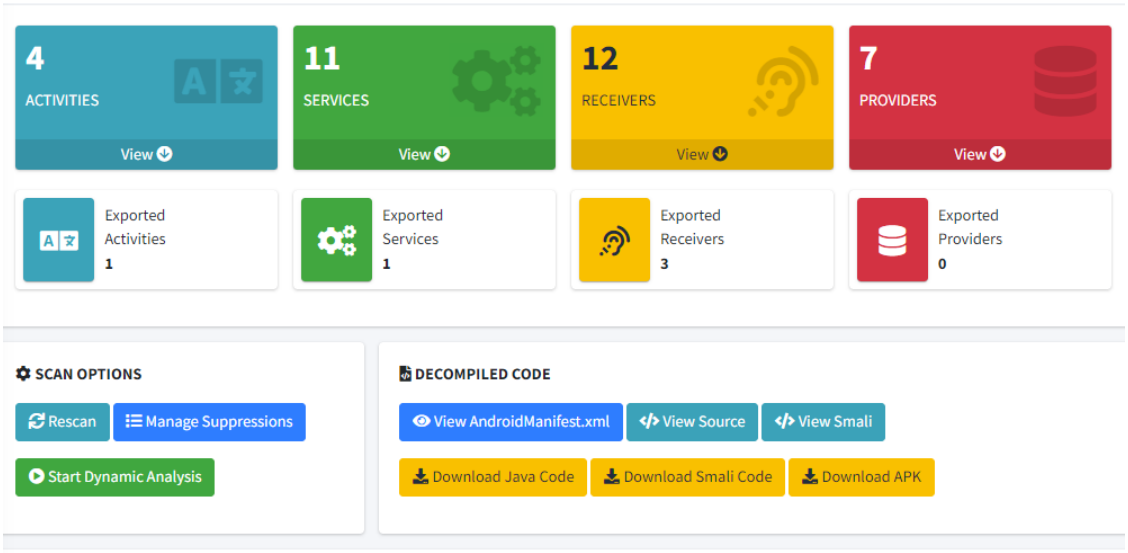
2.3.1. Báo cáo và các kết quả rà quét của công cụ MobSF

Sau khi thực hiện tiến trình phân tích tĩnh APK, trên trang đích các điểm mức độ nghiêm trọng được hiển thị. Tiếp theo, hàm băm, tên tệp và kích thước của APK cũng được cung cấp. Trong cột thứ ba ở hàng đầu tiên là tên gói, hoạt động chính, phiên bản SDK tối thiểu và cả phiên bản ứng dụng. Mô tả của ứng dụng cũng được đưa ra.



Hình 2.2. Mô tả thông tin ứng dụng

- Trong các thẻ nhỏ, chúng ta thấy các thành phần ứng dụng khác nhau
- Tùy chọn để xem mã dịch ngược. Đây là mã được tạo bởi apktool, tệp tài nguyên cũng được giải mã. Và có thể dễ dàng phân tách và xem mã nguồn trong các lớp java riêng biệt.



Hình 2.3. Thành phần ứng dụng

Phân tích chứng chỉ người ký trong cột chứng chỉ, chúng ta có thể thấy chứng chỉ người ký, nơi người ta có thể tìm thấy thông tin quan trọng về nhà phát triển, quốc gia, tiểu bang, loại bí danh, kích thước bit, v.v.

SIGNER CERTIFICATE

```
Binary is signed
v1 signature: True
v2 signature: True
v3 signature: False
v4 signature: False
X.509 Subject: C=VN, ST=Hanoi, L=Hanoi, O=MiTC, OU=1, CN=mitc
Signature Algorithm: rsassa_pkcs1v15
Valid From: 2021-03-29 16:46:23+00:00
Valid To: 2046-03-23 16:46:23+00:00
Issuer: C=VN, ST=Hanoi, L=Hanoi, O=MiTC, OU=1, CN=mitc
Serial Number: 0x34b7d15
Hash Algorithm: sha256
md5: 57d887abc8d2340a416a49538cf76d12
sha1: d09b296b0868b439ff1f6d2eebc86ec3a1ee87a1
sha256: b512d36717fa82f6e61e377b0697ef749f0c5acdc29d76d67c051eaabcfe929b
sha512: a9134b5bdad4e445413a16d8bc5c60fa89cd87f1ffdfb33a51f49f109ffc6babdee8baf68a93e4ab558598365e20bc4e240204fc52c85560f10401127b2e95c4
PublicKey Algorithm: rsa
Bit Size: 2048
Fingerprint: 366c9f70143a3e4e8026a50788c7f336ec25fddd89b60b890b5ea9f151f66b21
Found 1 unique certificates
```

Hình 2.4. Chứng chỉ người ký

Quyền ứng dụng có nhiều quyền khác nhau được phân loại là nguy hiểm hoặc bình thường. Theo quan điểm của nhà phân tích bảo mật, điều quan trọng là phải hiểu quyền nào có thể dẫn đến thiệt hại.

Ví dụ: nếu một ứng dụng có quyền truy cập vào phương tiện bên ngoài và lưu trữ thông tin quan trọng trên phương tiện bên ngoài thì có thể bị xem là nguy hiểm vì các tệp được lưu trữ trên phương tiện bên ngoài có thể đọc và ghi được trên toàn cục.

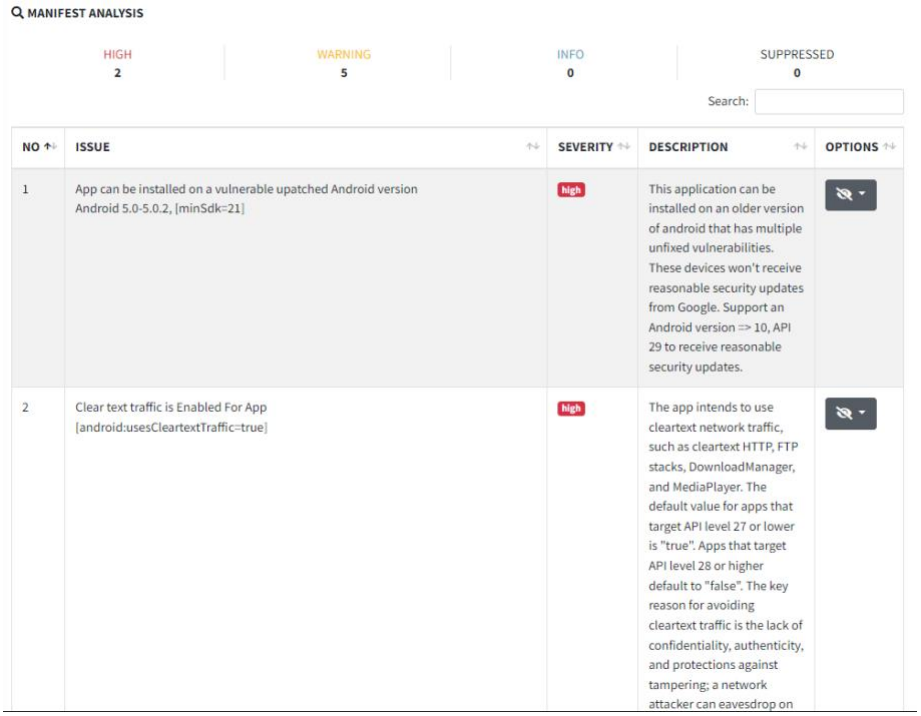
PERMISSION	STATUS	INFO	DESCRIPTION	CODE MAPPINGS
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.	Show Files
android.permission.ACCESS_WIFI_STATE	normal	view Wi-Fi status	Allows an application to view the information about the status of Wi-Fi.	Show Files
android.permission.CAMERA	dangerous	take pictures and videos	Allows application to take pictures and videos with the camera. This allows the application to collect images that the camera is seeing at any time.	Show Files
android.permission.DOWNLOAD_WITHOUT_NOTIFICATION	unknown	Unknown permission	Unknown permission from android reference	
android.permission.FOREGROUND_SERVICE	normal	enables regular apps to use Service.startForeground.	Allows a regular application to use Service.startForeground.	Show Files
android.permission.INTERNET	normal	full Internet access	Allows an application to create network sockets.	Show Files
android.permission.READ_CALENDAR	dangerous	read calendar events	Allows an application to read all of the calendar events stored on your phone. Malicious applications can use this	Show Files

Hình 2.5. Quyền ứng dụng

Hoạt động có thể duyệt và phân tích an ninh mạng: Trong phần bảo mật mạng, ta có thể tìm thấy một số chi tiết về các vấn đề an ninh mạng liên quan đến ứng dụng. Những vấn đề này đôi khi có thể dẫn đến các cuộc tấn công nghiêm trọng như MiTM.

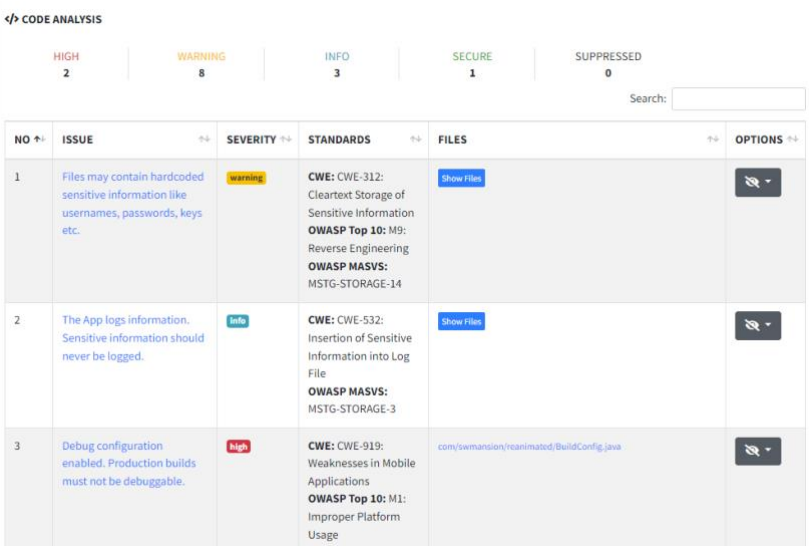
Hình 2.6. Hoạt động có thể duyệt

Phân tích Tệp kê khai: có thể tìm thấy nhiều thông tin từ tệp kê khai android như hoạt động nào được xuất, ứng dụng có thể gỡ lỗi hay không, lược đồ liệu, v.v. Để tham khảo, hãy xem ảnh chụp màn hình bên dưới.



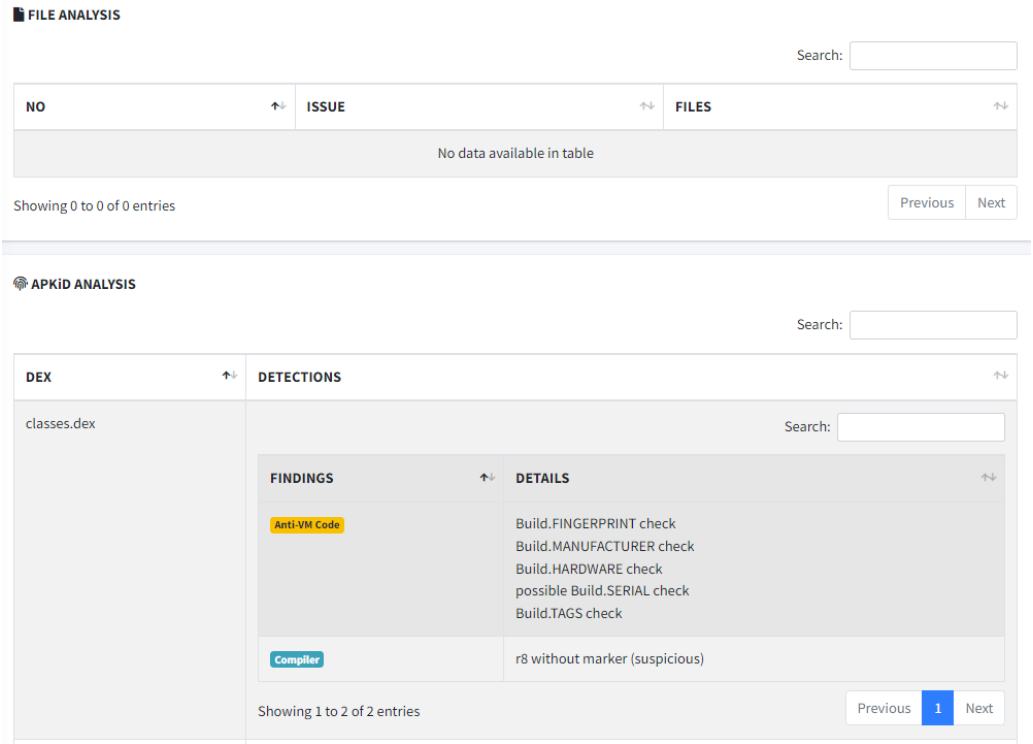
Hình 2.7. Phân tích tệp kê khai

Phân tích mã: có thể thấy rằng MobSF đã phân tích và so sánh một số hành vi của ứng dụng dựa trên các thông lệ tiêu chuẩn bảo mật của ngành như OWASP MSTG và lập bản đồ các lỗ hổng với OWASP Top 10. Các phân tích và so sánh này có thể giúp phân tích các tình huống khác nhau và tạo báo cáo dễ dàng hơn.



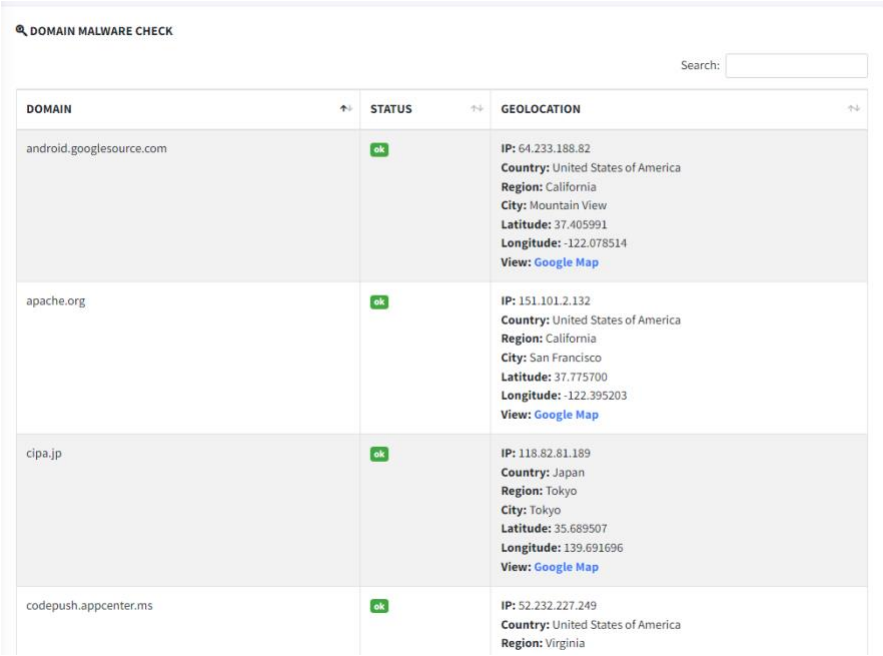
Hình 2.8. Phân tích mã

Phân tích phần mềm độc hại MobSF có tích hợp APKiD là một công cụ mã nguồn mở rất hữu ích để xác định các trình đóng gói, trình biên dịch, obfuscators khác nhau, v.v. trong các tệp android tương tự như PEiD trong APK.



Hình 2.9. Phân tích phần mềm độc hại

MobSF cũng trích xuất tất cả các URL / địa chỉ IP được mã hóa cứng hoặc đang được sử dụng trong ứng dụng và hiển thị trạng thái phần mềm độc hại cũng như sử dụng ip2location để cung cấp vị trí địa lý IP.



Hình 2.10. URL/IP được trích xuất

Các email được mã hóa cứng trong MobSF cũng được hiển thị tại trang báo cáo. Tất cả điều này được thực hiện bằng cách sử dụng mã nguồn được dịch ngược. Thông thường, một pentester có thể tìm thấy các ID email quan trọng đang được sử dụng làm thông tin đăng nhập trên trang web của bên thứ ba để truy cập cơ sở dữ liệu. Như email thì các URL cũng thường được mã hóa cứng. Có thể tìm thấy các URL hấp dẫn đôi khi được sử dụng. Thông thường, các nhà phân tích nhận thấy các URL độc hại cũng được truy cập hoặc thậm chí là máy chủ C&C.

Các thành phần hoạt động hiện tại: Danh sách tất cả các hoạt động hiện có cũng có thể xem bằng MobSF. Điều này cung cấp thông tin chi tiết về khung của APK android. Ngoài ra, đôi khi jadx thay thế tên thật của lớp bằng một số chữ cái ngẫu nhiên nếu nhà phát triển đã áp dụng phương pháp xáo trộn, MobSF cũng có thể liên kết tên thật của nó[8].

Hình 2.11. Các thành phần hoạt động

2.3.2. Phương pháp kiểm tra mã nguồn của tập luật trong MobSF

Nguyên lý hoạt động của phương pháp kiểm tra mã nguồn của tập luật trong công cụ rà quét tự động thường bao gồm các bước chính sau:

1. Thu thập mã nguồn

2. Phân đoạn (parsing) và phân tích cú pháp

3. Xây dựng đồ thị ưu tiên

4. Kiểm tra lỗ hổng tĩnh

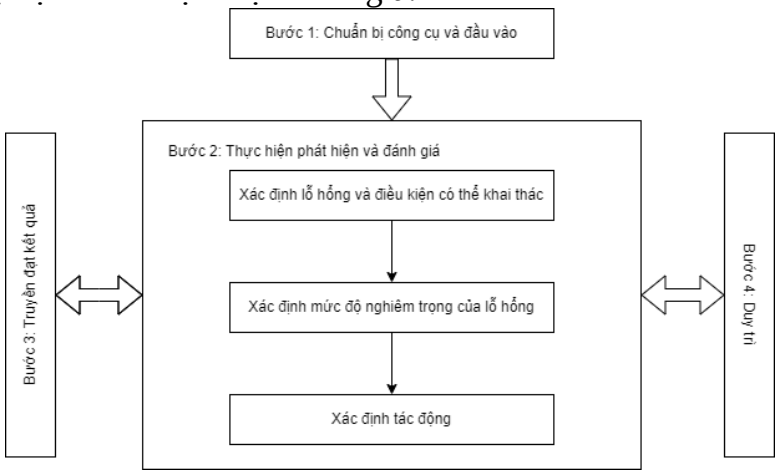
5. Phát hiện lỗ hổng bảo mật

6. Tạo báo cáo và thông báo

2.4. Quy trình phát hiện lỗ hổng mã nguồn dựa trên công cụ MobSF và tập luật

Thông qua phân tích hiện trạng của ứng dụng bao gồm mô tả thông tin ứng dụng, lỗ hổng bảo mật tồn tại, các biện pháp an toàn bảo mật đang dùng, từ đó xác định rủi ro và đánh giá ảnh hưởng của rủi ro tạo nên cho ứng dụng, đồng thời đưa ra giải pháp nhằm giảm thiểu rủi ro.

Quy trình phát hiện lỗ hổng mã nguồn bao gồm 4 bước, mỗi bước được chia thành một nhóm các nhiệm vụ. Đối với mỗi nhiệm vụ, hướng dẫn bổ sung cung cấp thông tin bổ sung cho thực hiện đánh giá rủi ro. Hình sau minh họa các bước cơ bản trong quy trình đánh giá rủi ro và các nhiệm vụ cụ thể để thực hiện đánh giá.



Hình 2.12. Các bước cơ bản trong quy trình đánh giá rủi ro

Bảng dưới đây là tổng hợp các nhiệm vụ đánh giá rủi ro:

Bảng 2.1. Bảng tổng hợp các nhiệm vụ phát hiện và đánh giá lỗ hổng

Nhiệm vụ	Mô tả nhiệm vụ
Bước 1: Chuẩn bị đánh giá	
Nhiệm vụ 1-1: Xác định mục đích	Xác định mục đích của phát hiện lỗ hổng theo thông tin mà đánh giá được dự định và đưa ra các quyết định.
Nhiệm vụ 1-2: Xác định phạm vi	Xác định phạm vi đánh giá rủi ro, khung thời gian được hỗ trợ và cân nhắc về công nghệ.
Nhiệm vụ 1-3: Xác định các đánh giá và hạn chế	Xác định các giả định và ràng buộc cụ thể theo đó đánh giá rủi ro được thực hiện.
Nhiệm vụ 1-4: Xác định nguồn thông tin	Xác định các nguồn thông tin mô tả, đe dọa, dễ bị tổn thương và tác động sẽ được sử dụng trong đánh giá rủi ro lỗ hổng có thể gây ra.
Bước 2: Thực hiện đánh giá	

Nhiệm vụ 2-1: Xác định nguồn gốc	Xác định và mô tả các nguồn đe dọa, bao gồm khả năng, ý định và đặc điểm mục tiêu cho các mối đe dọa đối nghịch và phạm vi ảnh hưởng đối với các mối đe dọa không đối nghịch.
Nhiệm vụ 2-2: Xác định sự kiện	Xác định các sự kiện đe dọa tiềm ẩn, mức độ liên quan của các sự kiện và các nguồn đe dọa có thể bắt đầu các sự kiện.
Nhiệm vụ 2-3: Xác định nhiệm vụ và điều kiện bảo đảm	Xác định các lỗ hổng và các điều kiện có xu hướng ảnh hưởng đến khả năng các mối quan tâm đe dọa dẫn đến các tác động bất lợi.
Nhiệm vụ 2-4: Xác định các tác động	Xác định các tác động bất lợi từ các sự kiện đe dọa đáng quan tâm, xem xét: (i) các đặc điểm của các nguồn đe dọa có thể bắt đầu các sự kiện; (ii) các lỗ hổng / điều kiện có khuynh hướng được xác định; và (iii) các biện pháp bảo vệ / biện pháp đối phó được lên kế hoạch hoặc thực hiện để cản trở các sự kiện đó.
Nhiệm vụ 2-5: Xác định rủi ro	Xác định rủi ro cho tổ chức từ các sự kiện đe dọa cần quan tâm: (i) tác động sẽ xảy ra từ các sự kiện; và (ii) khả năng xảy ra sự kiện.
Bước 3: Truyền đạt kết quả	
Nhiệm vụ 3-1: Truyền đạt kết quả	Truyền đạt kết quả cho những người ra quyết định tổ chức để hỗ trợ các phản ứng rủi ro.
Nhiệm vụ 3-2: Chia sẻ thông tin liên quan	Chia sẻ thông tin liên quan đến rủi ro được tạo ra trong quá trình đánh giá rủi ro với nhân viên tổ chức phù hợp.
Bước 4: Duy trì	
Nhiệm vụ 4-1: Giám sát các yếu tố rủi ro	Tiến hành giám sát liên tục các yếu tố rủi ro góp phần thay đổi rủi ro đối với hoạt động của ứng dụng.
Nhiệm vụ 4-2: Cập nhật đánh giá	Cập nhật đánh giá rủi ro hiện có.

Quy trình phát hiện lỗ hổng mã nguồn:

Bước 1: Thu thập thông tin

Thu thập tất cả thông tin liên quan đến mục tiêu cần đánh giá: Source code của ứng dụng. Như xác định ngôn ngữ lập trình, Framework, … được sử dụng.

Bước 2: Xác định các điểm yếu/lỗ hổng bảo mật

Bước 3: Tổng hợp kết quả và khuyến nghị khắc phục

Tổng hợp lại tất các các thông tin thu thập được, nội dung của quá trình làm việc và kết quả thu được để đưa ra báo cáo kết quả, đề xuất các phương án, giải pháp để khắc phục các lỗ hổng, điểm yếu bảo mật.

2.4.1. Thành phần trong tập luật của MobSF

Để tích hợp quy trình phát hiện lỗ hổng mã nguồn, cách tiếp cận theo tập luật từ công cụ Mobsf nhằm giải quyết nguy cơ về ATTT. Dưới đây là một số các luật liên quan đến lỗ hổng mã nguồn và phân loại theo các quy tắc thường được áp dụng để phát hiện lỗ hổng mã nguồn :

1. Quy tắc về bảo mật:

Write app directory là ứng dụng có quyền ghi vào thư mục của chính nó và trong đó có thông báo rằng thông tin nhạy cảm nên được mã hóa.

SQL cipher là ứng dụng sử dụng SQL Cipher (một thư viện mã hóa cho cơ sở dữ liệu SQLite), và đồng thời cảnh báo về việc không nên lưu trữ các thông tin bí mật trực tiếp trong mã nguồn của ứng dụng.

Aar jar debug enabled cảnh báo về việc cấu hình gỡ lỗi đã được kích hoạt trong ứng dụng. Nó nêu rõ rằng trong bản build dành cho môi trường sản xuất, không nên cho phép khả năng gỡ lỗi.

Debugger detect phát hiện mã nguồn trong ứng dụng đã được bảo vệ bằng DexGuard, và mã nguồn này chứa mã để phát hiện sự tồn tại của bộ gỡ lỗi trong quá trình thực thi ứng dụng.

Emulator detect phát hiện mã nguồn trong ứng dụng đã được bảo vệ bằng DexGuard, và mã nguồn này chứa các đoạn mã được thiết kế để phát hiện sự chạy trên máy ảo.

Debug sign khi phát triển và kiểm thử ứng dụng Android, thường sử dụng một chìa khóa gỡ lỗi để ký ứng dụng, giúp việc gỡ lỗi và theo dõi thông tin trên thiết bị Android[6].

Dexguard root detection cho biết mã nguồn liên quan đến việc phát hiện thiết bị đã được root trong ứng dụng Android được bảo vệ bằng DexGuard đã được phát hiện. Trong ngữ cảnh này:

2. Quy tắc về Hiệu suất:

Tamper detect mã liên quan đến việc phát hiện sự thay đổi trong ứng dụng được bảo vệ bằng DexGuard đã được xác định hoặc phát hiện ra. Trong ngữ cảnh này:

3. Quy tắc về Độ tin cậy:

WebView thực thi mã do người dùng kiểm soát trong WebView là một lỗ hổng bảo mật nghiêm trọng. Cảnh báo về một vấn đề bảo mật trong cách ứng dụng xử lý và sử dụng WebView. Dưới đây là một số điểm quan trọng:

4. Quy tắc về Kiểm soát biên:

Logging là cảnh báo về việc ứng dụng đang ghi lại thông tin trong quá trình chạy, nhưng thông tin nhạy cảm không nên xuất hiện trong nhật ký. Điều này là quan trọng để bảo vệ thông tin cá nhân và giữ cho ứng dụng tuân thủ các chuẩn bảo mật và quy định về quyền riêng tư.

5. Quy tắc về Tiêu Chuẩn Mã:

Insecure random là ứng dụng sử dụng một công cụ tạo số ngẫu nhiên không an toàn. Một công cụ tạo số ngẫu nhiên không an toàn có thể tạo ra các số ngẫu nhiên không đều, dự đoán được, hoặc có thể bị dự đoán bởi người khác. Trong lập trình và bảo mật thông tin, việc sử dụng một hàm tạo số ngẫu nhiên không an toàn có thể tạo ra nhiều vấn đề bảo mật.

Một số vấn đề có thể xuất hiện khi ứng dụng sử dụng một công cụ tạo số ngẫu nhiên không an toàn bao gồm:

6. Quy tắc về Ngôn Ngữ và API:

Safetynet cho biết ứng dụng này sử dụng SafetyNet API. Dưới đây là một số giải thích liên quan:

7. Quy tắc về Giao Diện Người Dùng (UI):

Hidden UI là các phần tử ẩn trong giao diện người dùng có thể được sử dụng để che giấu thông tin khỏi người dùng, nhưng rủi ro là thông tin này có thể bị rò rỉ hoặc bị tiết lộ ra

bên ngoài. Trong lập trình web hoặc phát triển ứng dụng, các phần tử ẩn thường được sử dụng để lưu trữ dữ liệu hoặc giữ các thành phần không mong muốn nằm ngoài tầm nhìn của người dùng. Tuy nhiên, nếu không có các biện pháp bảo mật phù hợp, thông tin này có thể trở nên khả dụng cho kẻ tấn công hoặc người dùng cuối có kiến thức kỹ thuật cao.

2.4.2. Phương pháp kiểm tra mã nguồn của tập luật trong MobSF

Nguyên lý hoạt động của phương pháp kiểm tra mã nguồn của tập luật trong công cụ rà quét tự động thường bao gồm các bước chính sau:

1. Thu thập mã nguồn: Công cụ sẽ thu thập mã nguồn của ứng dụng hoặc dự án từ nguồn cung cấp (ví dụ: tệp mã nguồn, repository Git). Đối với ứng dụng di động, có thể là tệp tin APK (Android) hoặc IPA (iOS).

2. Phân đoạn (parsing) và phân tích cú pháp: Công cụ sẽ phân tích cú pháp của mã nguồn để hiểu cấu trúc và ngữ pháp của mã. Điều này thường bao gồm việc sử dụng lexer và parser để chuyển đổi mã nguồn thành cây cú pháp.

3. Xây dựng đồ thị ưu tiên (Abstract Syntax Tree - AST): Công cụ tạo ra một đồ thị ưu tiên của mã nguồn, thường được biết đến là Abstract Syntax Tree (AST). AST biểu diễn cấu trúc cú pháp của mã nguồn và là cơ sở cho việc phân tích tiếp theo.

4. Kiểm tra lỗ hổng tĩnh (Static Analysis): Công cụ sử dụng các kỹ thuật phân tích tĩnh để kiểm tra mã nguồn mà không cần thực hiện thực thi chương trình. Các kỹ thuật này bao gồm:

- Kiểm tra quy tắc lập trình (Code Rule Checking): Kiểm tra xem mã nguồn có tuân thủ các quy tắc lập trình an toàn hay không.
- Phân tích luồng dữ liệu (Data Flow Analysis): Theo dõi cách dữ liệu được truyền đi trong mã nguồn để xác định các lỗ hổng liên quan đến dữ liệu nhạy cảm.
- Phân tích luồng điều khiển (Control Flow Analysis): Phân tích cách chương trình chuyển đổi giữa các câu lệnh để xác định các vấn đề an toàn điều khiển.

5. Phát hiện lỗ hổng bảo mật: Công cụ sử dụng cơ sở dữ liệu chứa các luật và quy tắc lỗ hổng bảo mật để so sánh với mã nguồn và phát hiện các mô hình hoặc cấu trúc mã nguồn có thể liên quan đến các lỗ hổng bảo mật.

6. Tạo báo cáo và thông báo: Sau khi hoàn tất quá trình phân tích, công cụ tạo ra báo cáo chi tiết về các lỗ hổng bảo mật phát hiện được. Báo cáo này thường bao gồm mô tả vấn đề, địa chỉ của nó trong mã nguồn, và đôi khi cung cấp giải pháp sửa chữa.

2.5. Kết luận chương 2

Phát hiện lỗ hổng mã nguồn là một bước quan trọng để ngăn chặn, giảm thiểu các rủi ro gây hại cho ứng dụng và khả năng bị lộ lọt thông tin hay mất dữ liệu. Cách phát hiện lỗ hổng là: sử dụng công cụ và phân tích báo cáo rà quét. Chương tiếp theo sẽ trình bày chi tiết quá trình phát hiện và đánh giá lỗ hổng mã nguồn cho một ứng dụng bao gồm các bước cài đặt, cấu hình công cụ, tạo kịch bản thử nghiệm, phân tích báo cáo và đánh giá kết quả.

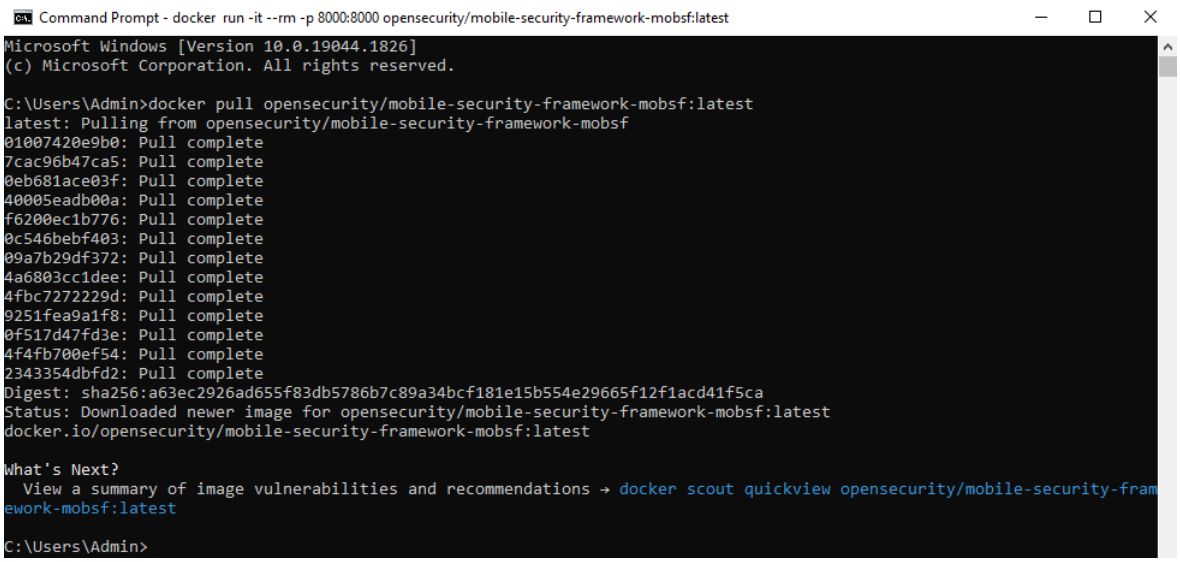
CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

3.1. Cài đặt và cấu hình công cụ

Để cài đặt công cụ MobSF, cần phải cài đặt docker để chạy được MobSF và làm theo lệnh:

docker pull opensecurity/mobile-security-framework-mobsf:latest

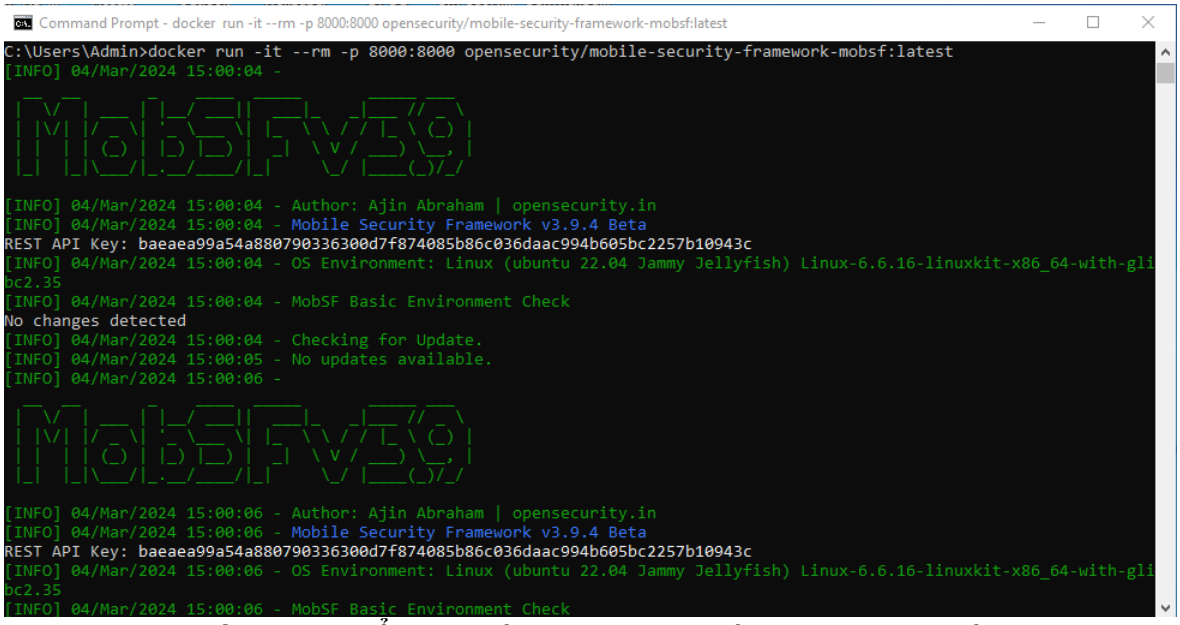
Cần tải và cài đặt phiên bản cuối của MobSF để có phiên bản, tính năng, bảo mật và sửa lỗi cập nhật nhất



Hình 3.1. Tải và cài đặt công cụ MobSF

Tiến hành triển khai công cụ MobSF với hỗ trợ phân tích tĩnh sử dụng câu lệnh sau:

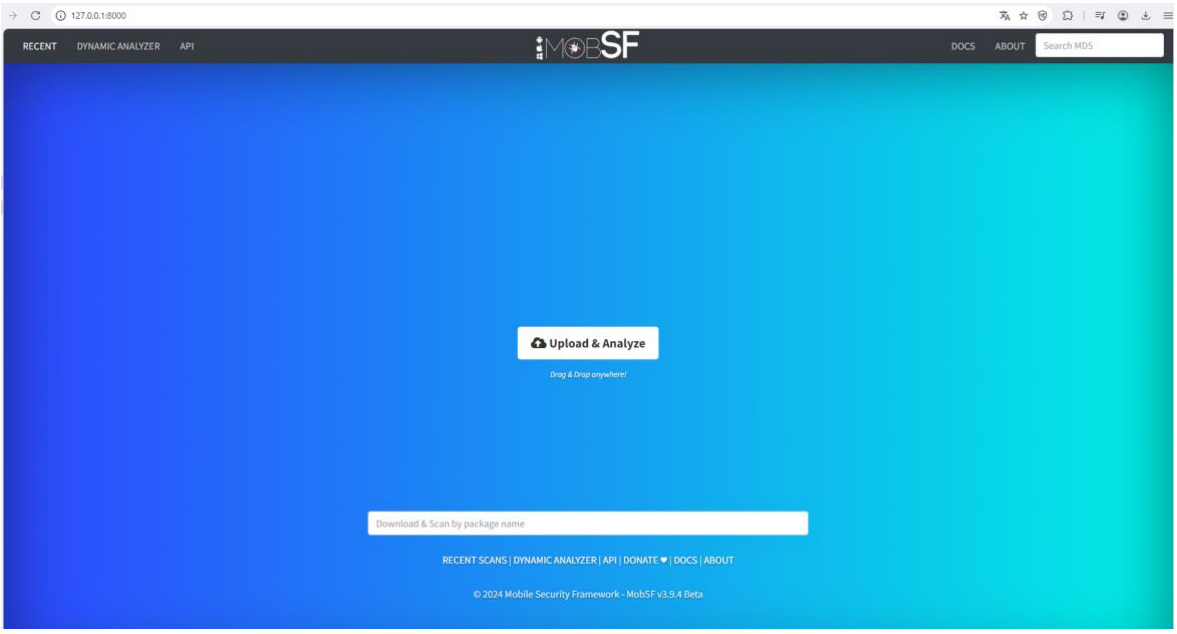
docker run -it --rm -p 8000:8000 opensecurity/mobile-security-framework-mobsf:latest



Hình 3.2. Triển khai công cụ MobSF trên máy chủ cục bộ

Bây giờ, để chạy MobSF thực hiện truy cập MobSF http://127.0.0.1:8000 trong trình duyệt web và MobSF sẽ chạy trên một máy chủ cục bộ trên cổng 8000.

Mở liên kết trong trình duyệt và xem MobSF đã được cài đặt



Hình 3.3. Hoàn thành cài đặt công cụ MobSF

3.2. Kịch bản thử nghiệm

Từ quy trình phát hiện lỗ hổng được trình bày ở trên, kịch bản thử nghiệm cho việc rà quét, đánh giá lỗ hổng bảo mật cho một phần mềm, ứng dụng cụ thể được lập và hoàn thành như sau:

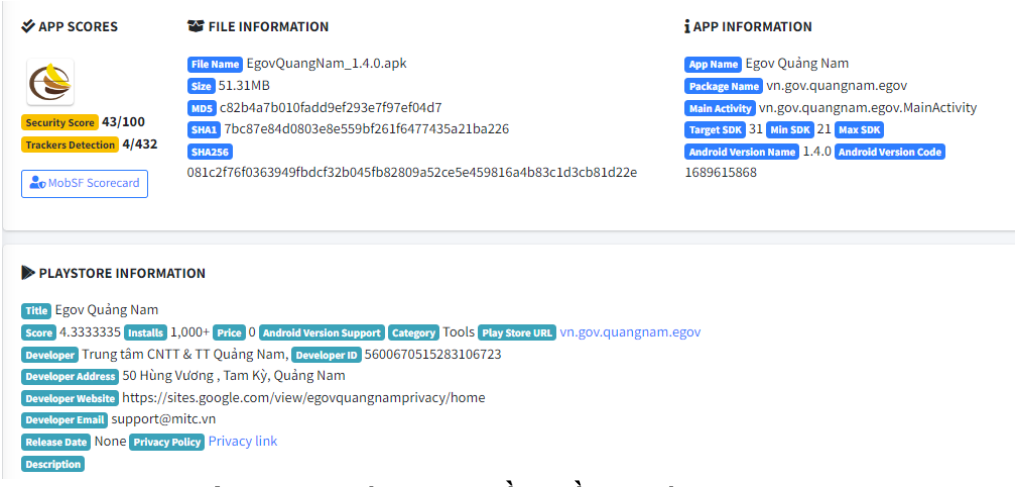
Bảng 3.1. Kịch bản thử nghiệm phát hiện lỗ hổng mã nguồn

Tên nhiệm vụ	Mô tả nhiệm vụ	Kết quả mong đợi
Bước 1: Chuẩn bị đánh giá		
Nhiệm vụ 1-1: Xác định mục đích	Tìm kiếm các lỗ hổng mã nguồn từ đó khắc phục các lỗ hổng và cải thiện độ tin cậy của phần mềm Egov Quảng Nam.	Đơn vị phát triển cung cấp tệp cài đặt đóng gói của phần mềm Egov Quảng Nam phiên bản Android.
Nhiệm vụ 1-2: Xác định phạm vi	Phát hiện lỗ hổng mã nguồn của phần mềm Egov Quảng Nam phiên bản Android, trong quá trình rà quét sẽ sử dụng công cụ MobSF.	
Nhiệm vụ 1-3: Xác định các đánh giá và hạn chế	Phát hiện các lỗ hổng trong mã nguồn của phần mềm Egov Quảng Nam phiên bản Android và yêu cầu bản vá đã cập nhật khắc phục lỗ hổng có mức độ nguy hại cao.	
Nhiệm vụ 1-4: Xác định nguồn thông tin	Phân tích chi tiết trong Báo cáo rà quét lỗ hổng bảo mật tại mục 3.3.	
Bước 2: Thực hiện đánh giá		
Nhiệm vụ 2-1: Xác định nguồn gốc	Phân tích chi tiết trong Báo cáo rà quét lỗ hổng bảo mật tại mục 3.3.	Rà quét, phân tích và xuất báo cáo để đơn vị phát triển khắc phục.
Nhiệm vụ 2-2: Xác định sự kiện	Phân tích chi tiết trong Báo cáo rà quét lỗ hổng bảo mật tại mục 3.3.	

Nhiệm vụ 2-3: Xác định nhiệm vụ và điều kiện bảo đảm	Các lỗ hổng có cảnh báo mức độ nguy hại cao gây ảnh hưởng đến an toàn bảo mật sẽ được yêu cầu khắc phục. Các cảnh báo còn lại sẽ khắc phục sau trong quá trình vận hành thực tế.	
Nhiệm vụ 2-4: Xác định các tác động	Phân tích chi tiết trong Báo cáo rà quét lỗ hổng bảo mật tại mục 3.3.	
Nhiệm vụ 2-5: Xác định rủi ro	Phân tích chi tiết trong Báo cáo rà quét lỗ hổng bảo mật tại mục 3.3.	
Bước 3: Truyền đạt kết quả		
Nhiệm vụ 3-1: Truyền đạt kết quả	Gửi báo cáo rà quét và bản phân tích để đơn vị phát triển khắc phục.	Đơn vị phát triển tiếp nhận báo cáo và khắc phục lỗ hổng trong phần mềm.
Nhiệm vụ 3-2: Chia sẻ thông tin liên quan	Đơn vị phát triển khắc phục và xử lý các lỗ hổng có cảnh báo mức độ nguy hại cao.	
Bước 4: Duy trì		
Nhiệm vụ 4-1: Giám sát các yếu tố rủi ro	Đơn vị phát triển gửi lại bản cập nhật để tiến hành rà quét và đánh giá lại phần mềm.	Rà quét, đánh giá và cập nhật lại báo cáo kết quả.
Nhiệm vụ 4-2: Cập nhật đánh giá	Cập nhật đánh giá rủi ro hiện có.	

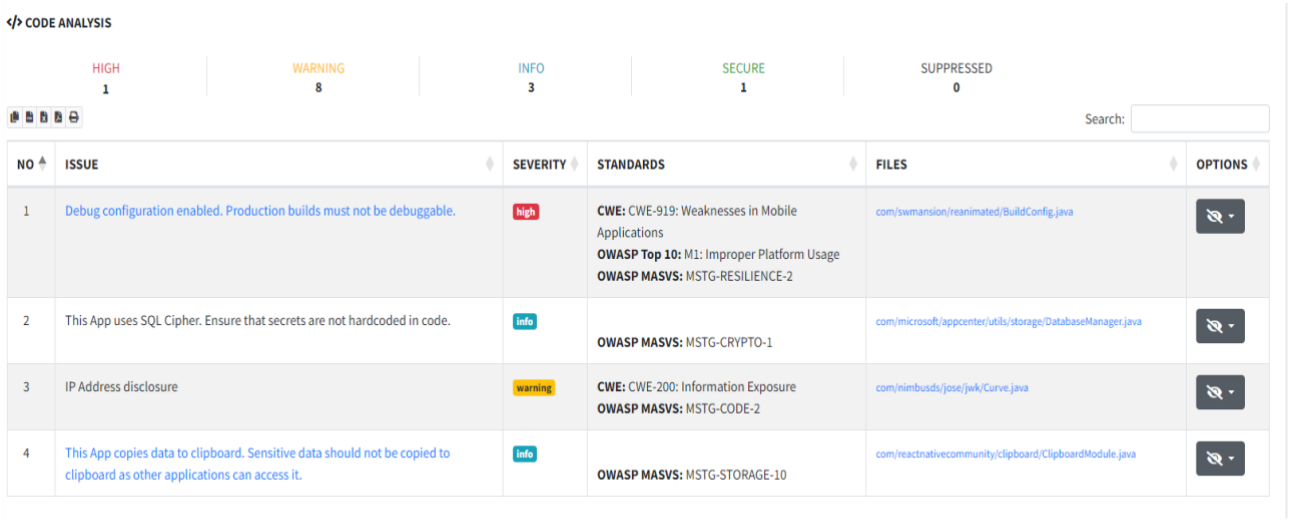
3.3. Thực nghiệm và đánh giá

Dựa trên kịch bản thử nghiệm để tiến hành rà quét phát hiện lỗ hổng mã nguồn của phần mềm Egov Quảng Nam phiên bản Android. Đầu tiên, tệp apk của phần mềm sẽ được rà quét bằng công cụ MobSF.



Hình 3.4. Thông tin phần mềm phiên bản 1.4.0

Sau khi rà quét, báo cáo phân tích mã nguồn chi tiết được xuất ra như sau:

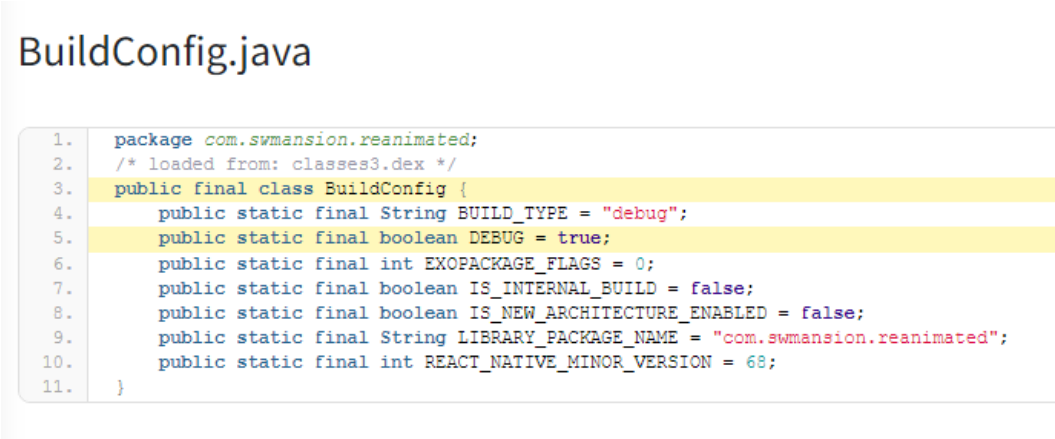


Hình 3.5. Báo cáo phân tích mã nguồn của phần mềm phiên bản 1.4.0

Theo kết quả trên, các lỗ hổng mã nguồn có mức độ nguy hại cao phát hiện được trong phiên bản 1.4.0 của phần mềm Egov Quảng Nam có 1 lỗ hổng:

1. android_aar_jar_debug_enabled
   - Mức độ cảnh báo: Cao
   - Tiêu chuẩn:
     - CWE: CWE-919: Weaknesses in Mobile Applications
     - OWASP Top 10: M1: Improper Platform Usage
     - OWASP MASVS: MSTG-RESILIENCE-2
   - Đường dẫn: com/swmansion/reanimated/BuildConfig.java
   - Mô tả: Chế độ gỡ lỗi thường cung cấp thông tin chi tiết về mã nguồn và hoạt động của ứng dụng, điều này có thể tạo ra lỗ hổng bảo mật và có thể được lợi dụng bởi những người có ý định xấu.
   - Tác động: Việc đặt cờ android:debuggable thành true sẽ cho phép kẻ tấn công gỡ lỗi ứng dụng, giúp chúng dễ dàng truy cập vào các phần của ứng dụng cần được bảo mật.
   - Giải pháp đề xuất giảm thiểu rủi ro: Đặt cờ android:debuggable thành false.

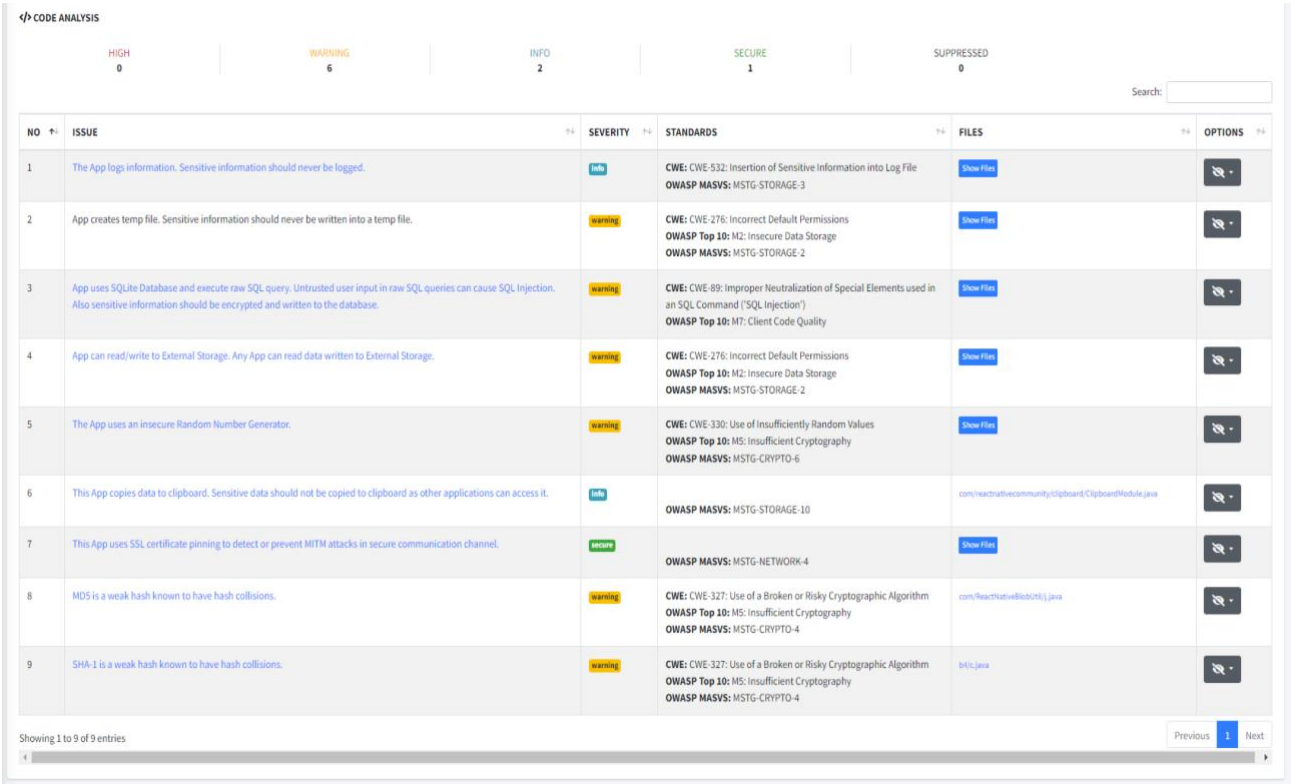
Sau khi hoàn thành rà quét và phân tích để đánh giá lỗ hổng mã nguồn trong ứng dụng, kết quả và báo cáo chuyển đến đơn vị phát triển để khắc phục sự cố.



Hình 3.6. Tệp mã nguồn phát hiện lỗ hổng hệ thống

Hình 3.7. Thông tin phần mềm phiên bản 1.5.1

Phần mềm sau khi được cập nhật sau khi phát hiện lỗ hổng mã nguồn đã không còn cảnh báo mức cao.



Hình 3.8. Báo cáo phân tích mã nguồn của phần mềm phiên bản 1.5.1

3.4. Kết luận chương 3

Như vậy qua kết quả thực nghiệm, có thể thấy thấy phương pháp phát hiện lỗ hổng mã nguồn dựa trên tập luật bằng công cụ MobSF đã phát hiện được lỗ hổng mã nguồn đồng thời công cụ cũng cho đánh giá về mức độ cảnh báo nguy hại của lỗ hổng. Qua việc rà quét, phân tích dựa trên báo cáo xuất ra từ MobSF các lỗ hổng có cảnh báo mức cao đã được phát hiện. Từ đó đánh giá được mức rủi ro của lỗ hổng và đề xuất phương án, giải pháp khắc phục lỗ hổng cho đơn vị phát triển để cập nhật khắc phục tại phiên bản sau của phần mềm. Sau khi tiếp thu báo cáo phân tích và đề xuất, đơn vị phát triển đã cập nhật khắc phục lỗ hổng mức cảnh báo cao trong phiên bản sau của phần mềm.

KẾT LUẬN

Luận văn đã trình bày một hướng tiếp cận về ứng dụng công cụ trong phát hiện lỗ hổng mã nguồn trên phần mềm nền tảng Android. Qua đó có thể đánh giá về việc sử dụng công cụ trong rà quét, phân tích, đánh giá, phát hiện các nguy cơ mã độc là một giải pháp khả thi, có hướng phát triển trong tương lai.

Đồng thời, luận văn đã cho thấy hiệu quả của việc sử dụng công cụ để phát hiện lỗ hổng mã nguồn từ khả năng tự động hóa, phát hiện sớm trước khi rủi ro bị khai thác, tối ưu được việc sử dụng nhân lực trong khi vẫn đảm bảo được chất lượng của phần mềm.

Các kết quả luận văn đạt được:

- Trình bày tổng quan về phát hiện lỗ hổng mã nguồn dựa trên tập luật.
- Trình bày về phương pháp phát hiện lỗ hổng mã nguồn dựa trên tập luật.
- Tiến hành thực nghiệm và đánh giá kết quả.

Hướng phát triển, nghiên cứu tiếp theo:

Mở rộng với hướng nghiên cứu phương pháp phát hiện lỗ hổng mã nguồn dựa trên tập luật với nền tảng IOS. Từ đó, tăng khả năng sử dụng công cụ để phát hiện, phân tích, đánh giá đa dạng và chính xác hơn.