TẠP CHÍ KHOA HỌC
TRƯỜNG ĐẠI HỌC SƯ PHẠM TP HỒ CHÍ MINH
Tập 21, Số 7 (2024): 1252-1264
Website: https://journal.hcmue.edu.vn
ISSN: 2734-9918

HO CHI MINH CITY UNIVERSITY OF EDUCATION
JOURNAL OF SCIENCE
Vol. 21, No. 7 (2024): 1252-1264
https://doi.org/10.54607/hcmue.js.21.7.4032(2024)

Bài báo nghiên cứu

PHÁT TRIỂN CÔNG CỤ HỖ TRỢ LẬP TRÌNH AN TOÀN CÓ KHẢ NĂNG TUỲ CHỈNH CHO PHÁT TRIỂN ỨNG DỤNG TRÊN ANDROID

Lê Công Bình, Nguyễn Lê Bảo Thi, Trương Phước Lộc*, Trần Minh Triết, Trần Anh Duy

Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Thành phố Hồ Chí Minh, Việt Nam

**Tác giả liên hệ: Trương Phước Lộc – Email: tploc@fit.hcmus.edu.vn*

Ngày nhận bài: 27-11-2023.; ngày nhận bài sửa: 31-3-2024; ngày duyệt đăng: 01-4-2024

TÓM TẮT

Mặc dù Android là một hệ điều hành di động phổ biến, hệ sinh thái ứng dụng của Android vẫn còn tồn tại nhiều vấn đề bảo mật. Thiếu sự nhận thức và quan tâm đối với vấn đề bảo mật trong việc phát triển ứng dụng Android là một trong những nguyên nhân chính gây ra tình trạng này. Trong bối cảnh hiện nay, các nhà phát triển vẫn chưa có đủ kiến thức về bảo mật. Do đó, chúng tôi đã nghiên cứu và đề xuất một công cụ hỗ trợ lập trình an toàn. Dựa trên ý tưởng của DevSecOps, nhà phát triển được đặt ở trung tâm để tối ưu hóa giải pháp cho vấn đề này bằng cách tích hợp lập trình an toàn vào giai đoạn đầu trong quá trình phát triển phần mềm. Bài báo này trình bày hai đóng góp chính của nghiên cứu: biên soạn và phân loại vấn đề bảo mật trong phát triển ứng dụng Android và phát triển công cụ ArmorDroid, một plugin cho Android Studio hỗ trợ lập trình an toàn. Plugin này, có thể sử dụng cho các tệp Java, Kotlin và XML, có thể quét và phát hiện mã nguồn có lỗ hổng ngay lập tức và đề xuất các sửa nhanh cho lập trình viên trong giai đoạn phát triển. Plugin này giúp lập trình viên cải thiện mã nguồn an toàn của họ và huấn luyện họ viết mã nguồn an toàn bằng cách cung cấp các tiêu chuẩn lập trình an toàn trong ứng dụng Android. Hơn nữa, lập trình viên có thể tùy chỉnh tập luật để phù hợp với tình huống của họ và chia sẻ nó với các lập trình viên khác. Công trình của chúng tôi cũng trình bày kết quả của một nghiên cứu thử nghiệm về hiệu suất của plugin ArmorDroid.

Từ khóa: Android Studio plugin; DevSecOps; Secure Coding

1. Giới thiệu

Những năm gần đây, ứng dụng ngày càng được sử dụng phổ biến trên khắp thế giới bởi những lợi ích mà nó mang lại cho con người. Trong các hệ điều hành ứng dụng di động thì Android và iOS là hai hệ điều hành được sử dụng nhiều nhất. Theo thống kê từ Statista (Statista, 2023), trong quý ba của năm 2022 số ứng dụng được tải xuống từ Google Play là 35,3 tỉ và từ App Store là 7 tỉ. Qua số liệu trên, chúng ta có thể thấy ứng dụng chạy trên hệ

Cite this article as: Le Cong Binh, Nguyen Le Bao Thi, Truong Phuoc Loc, Tran Minh Triet, & Tran Anh Duy (2024). Development of a customizable secure coding plugin for android application development. *Ho Chi Minh City University of Education Journal of Science, 21*(7), 1252-1264.

điều hành Android được sử dụng phổ biến hơn ứng dụng trên hệ điều hành iOS. Đã có những so sánh về mức độ bảo mật của ứng dụng trên hai hệ điều hành trên và các kết quả đang nghiêng về phía hệ điều hành iOS với khả năng bảo mật tốt hơn. Trước tiên về bản chất, iOS là một hệ thống mã nguồn đóng nên việc phân phối mã cho các nhà phát triển và tuỳ chỉnh thiết bị là rất nghiêm ngặt. Trong khi đó, Android là mã nguồn mở, các phiên bản của Android có thể dễ dàng tuỳ chỉnh và nâng cấp cũng như tìm kiếm các vấn đề bảo mật dễ dàng hơn và khắc phục nhanh hơn. Ngoài yếu tố về bảo mật của hệ điều hành, chúng ta cần quan tâm đến vấn đề kiểm duyệt trên cửa hàng thị trường ứng dụng của Android là Google Play và của iOS là App Store. Các ứng dụng trên App Store được kiểm tra chặt chẽ và mất nhiều thời gian để xuất bản lên thị trường hơn Google Play nên số lượng ứng dụng chứa mã độc của nó được kiểm soát đáng kể. Ngược lại, cửa hàng ứng dụng mở Google Play của Android tuy có nhiều ứng dụng để lựa chọn hơn App Store nhưng sẽ lượng phần mềm độc hại hay có vấn đề bảo mật nhiều hơn và gây mất an toàn cho người dùng khi tải xuống. Nguyên nhân là vì khâu kiểm soát ứng dụng tải lên của thị trường này không được chặt chẽ, các bên có thể dễ dàng xuất bản ứng dụng lên thị trường. Bên cạnh đó, người dùng Android cũng có thể thay đổi cấu hình để cài đặt các ứng dụng bên ngoài cửa hàng Google Play.

Kế thừa từ quy trình DevOps, DevSecOps là phương pháp tích hợp kiểm thử bảo mật vào mọi giai đoạn của quy trình phát triển phần mềm. Từ tư tưởng *"Shifting-left"*, việc tích hợp kiểm thử bảo mật ở giai đoạn sớm trong quy trình phát triển phần mềm giúp phát hiện vấn đề bảo mật sớm từ đó giảm chi phí và tiết kiệm thời gian trong việc khắc phục các lỗi bảo mật. Giải pháp được đưa ra là lập trình an toàn ngay ở giai đoạn phát triển để có thể phát hiện và sửa lỗi một cách nhanh nhất. Chính vì vậy, lập trình viên được đặt làm trung tâm để tối ưu hoá phương pháp giải quyết vấn đề này. Theo nghiên cứu của (Tran et al., 2021), những nguyên nhân chính dẫn đến vấn đề lập trình kém an toàn của các lập trình viên là họ chưa có nhận thức về vấn đề bảo mật trong lập trình ứng dụng, chưa được cung cấp đủ kiến thức và chưa được đào tạo thực tế về các vấn đề bảo mật có thể xảy ra do lỗ hổng bảo mật trong quá trình phát triển. Thực tế thì họ đã phải chịu áp lực về việc phát triển nhanh sản phẩm từ các bên liên quan cũng như chưa có nhiều quy chuẩn cho việc lập trình bảo mật dành cho họ. Ngoài ra, hiện nay tiêu chuẩn về lập trình bảo mật đang còn thiếu và chưa được phổ biến. Nhiều nghiên cứu và công cụ có thể quét và phát hiện được lỗ hổng bảo mật trong source code của ứng dụng Android bằng các kĩ thuật phân tích tĩnh (static analysis) (Senanayake, 2023). Tuy nhiên, các phương pháp này chỉ có thể áp dụng cho có sản phẩm hoàn chỉnh và vẫn không đặt lập trình vào trung tâm của vấn đề. Vì vậy, nhóm tác giả đã thực hiện đề tài nghiên cứu giải pháp giúp hỗ trợ lập trình viên trong giai đoạn lập trình một cách nhanh chóng và hiệu quả. Giải pháp này không loại bỏ đi sự cần thiết của các công cụ quét lỗi bảo mật khác mà chỉ là thêm một bước để chủ động phòng chống các lỗ hổng bảo mật theo chiến lược defense in depth. Các đóng góp chính của nghiên cứu bao gồm:

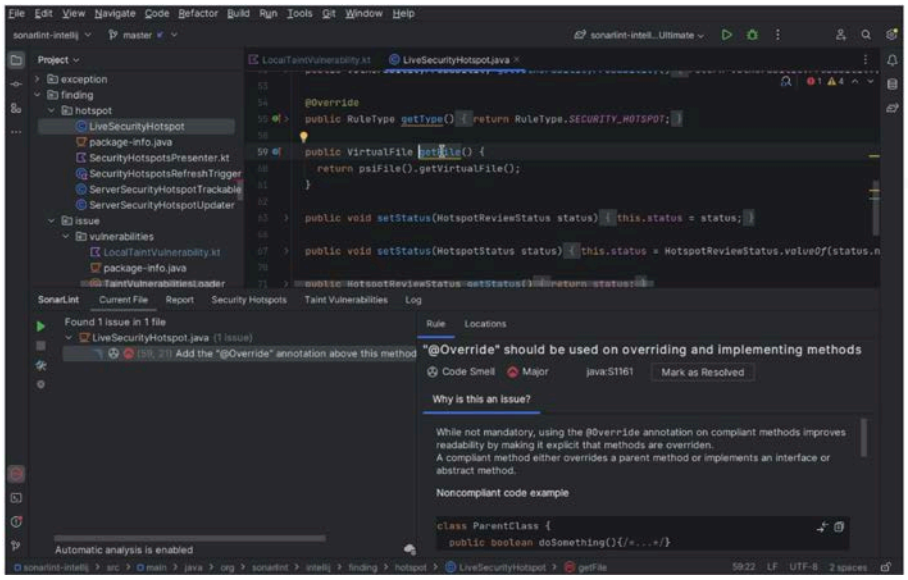
- Tổng hợp và phân loại các lỗi bảo mật trong lập trình ứng dụng Android;
- Xây dựng công cụ, ArmorDroid, hỗ trợ lập trình ứng dụng Android an toàn được tích hợp trong Android Studio;
- Công cụ hỗ trợ này và bộ quy tắc kèm theo sẽ giúp các lập trình viên không chỉ lập trình ra những đoạn mã an toàn trong giai đoạn phát triển mà còn hỗ trợ các lập trình viên trong việc học tập và thực hành thực tế;
- Chạy thực nghiệm công cụ để khảo sát kết quả ban đầu của công cụ.

2. Nội dung

2.1. Các công trình liên quan

Bằng cách thực hiện phân tích mã nguồn tĩnh, chúng ta có thể kiểm tra luồng điều khiển và luồng dữ liệu của chương trình mà không cần thực thi nó. EstiDroid là một công cụ mã nguồn mở được phát triển bởi (Fan et al., 2020). Nó phân tích các cuộc gọi API của ứng dụng Android bằng cách quét tệp APK. Phương pháp này nhanh hơn so với phân tích động nhưng ít chính xác hơn.

Một số giải pháp đề xuất các công cụ dưới dạng plugin cho môi trường phát triển tích hợp (IDE), giúp nhà phát triển phát hiện mã không an toàn ngay tại thời điểm lập trình (Tebib, 2023). SonarLint (SonarLint, 2023) là một plugin kiểm tra mã an toàn hỗ trợ nhiều IDE như Eclipse, Visual Studio và Android Studio (Hình 1). Plugin này cung cấp phản hồi thời gian thực cho các ngôn ngữ lập trình như Python, JavaScript và Java. DroidPatrol của (Talukder et al., 2019) là một plugin cho Android Studio thực hiện phân tích an ninh dựa trên phân tích mã nguồn tĩnh. Nó giải mã tệp APK đầu vào và sau đó tạo và phân tích đồ thị cuộc gọi (giữa định nghĩa phương thức và vị trí gọi phương thức) để tìm kiếm các rò rỉ dữ liệu có thể. Nhược điểm là nhà phát triển chỉ có thể kiểm tra mã của họ sau khi biên dịch thành tệp APK, không phải trong quá trình lập trình.



Hình 1. *Plugin SonarLint cho Android*

FixDroid, một plugin cho Android Studio được phát triển bởi (Nguyen et al., 2017), giải quyết vấn đề thời gian thực này. Nó cảnh báo lập trình viên khi họ viết mã có lỗ hổng và cung cấp nhiều gợi ý bảo mật để sửa ngay lập tức. Tuy nhiên, số lượng vấn đề mà họ có thể phát hiện là hữu hạn, và nhà phát triển không thể mở rộng các quy tắc cho các vấn đề bảo mật mới. FixDroid cũng cần một máy chủ để xử lí vấn đề bảo mật được gửi từ máy khách, điều này làm đặt ra vấn đề về quyền riêng tư của mã nguồn. 9Fix của (Tran et al., 2021) và Sensei của (Cremer et al., 2020) là hai plugin giúp lập trình viên viết mã an toàn và tuân thủ hướng dẫn lập trình. Chúng cung cấp phản hồi và gợi ý thời gian thực để sửa lỗi bảo mật và cải thiện chất lượng mã nguồn. Một hạn chế hiện tại của hai plugin này là chúng không hỗ trợ Kotlin, một ngôn ngữ hiện đại và phổ biến cho phát triển ứng dụng Android.

2.2. Các loại lỗi bảo mật trong lập trình ứng dụng Android

Trong phần này, nhiều vấn đề bảo mật khác nhau ảnh hưởng đến ứng dụng Android và cách chúng xảy ra sẽ được trình bày. Điều này giúp chúng ta học từ những sai lầm phổ biến của nhà phát triển Android và tìm cách khắc phục chúng.

2.2.1. Các vấn đề bảo mật liên quan đến mật mã học

Mật mã là yếu tố quan trọng để bảo vệ dữ liệu trong quá trình trao đổi thông tin và lưu trữ dữ liệu nhằm đảm bảo tính bí mật, tính toàn vẹn và tính sẵn sàng của dữ liệu – ba tính này còn gọi là tam giác C-I-A (confidentiality, integrity, availability). Khi sử dụng không đúng cách các công cụ và kĩ thuật trong mật mã thì sẽ dẫn đến rò rỉ dữ liệu, suy yếu tính bảo mật của hệ thống… - đó là những vấn đề bảo mật trong việc sử dụng mật mã.

Một số vấn đề phổ biến liên quan đến mật mã trong các ứng dụng Android là vi phạm dữ liệu (data breaches). Vấn đề này xảy ra khi dữ liệu nhạy cảm của cá nhân bị truy cập hoặc tiết lộ trái phép bởi những người không được ủy quyền. Thuật toán yếu, sử dụng khóa yếu như độ dài khóa, nguồn entropy không đủ vì độ ngẫu nhiên vì entropy rất quan trọng để tạo ra các khóa mật mã mạnh, quản lí khóa không đúng cách hoặc lưu trữ khóa không an toàn, hoặc các lỗ hổng khác liên quan đến mật mã. Đây là những nguyên nhân đến từ người lập trình làm cho việc tin tặc đánh cắp dữ liệu nhạy cảm và xâm phạm quyền riêng tư của người dùng một cách dễ dàng hơn.

2.2.2. Các vấn đề bảo mật trong giao tiếp mạng

Vấn đề bảo mật trong giao tiếp mạng trong ứng dụng Android đề cập đến các lỗ hổng hoặc yếu điểm trong cách ứng dụng tương tác với các thiết bị hoặc máy chủ khác qua mạng. Những vấn đề này có thể tiếp xúc ứng dụng và dữ liệu mà nó chứa đến các cuộc tấn công, tăng nguy cơ xảy ra việc xâm nhập dữ liệu, đe dọa tính toàn vẹn hệ thống hoặc những sự cố bảo mật khác tương tự như các vấn đề trong triển khai mật mã vi phạm ba tính CIA.

Thực tế có rất nhiều vấn đề bảo mật ảnh hưởng đến giao tiếp mạng đang xảy ra và mang lại tác hại nghiêm trọng cho trải nghiệm của người dùng như Eavesdropping Attacks, Man-in-the-Middle (MitM) Attacks, Denial of Service (DoS) Attacks, Distributed Denial of Service (DDoS) Attacks, Packet Sniffing hay giả mạo IP (IP Spoofing), giả mạo dữ liệu (Data Tampering), giả mạo DNS (DNS Spoofing)… Man-in-the-Middle (MITM) xảy ra khi người tấn công chặn, đọc và thay đổi thông tin giao tiếp đã được mã hóa ở giữa người gửi và người nhận mà không ai hay biết nếu hệ thống mã hóa yếu hoặc triển khai kém. Trên môi trường mạng, những kẻ tấn công có thể sử dụng các công cụ chuyên dụng để theo dõi và chặn lưu lượng mạng để thu thập các gói tin trên đường truyền không an toàn và trích xuất, phân tích dữ liệu nhạy cảm từ chúng như thông tin cá nhân, thẻ tín dụng, mật khẩu.

2.2.3. Các vấn đề bảo mật trong lưu trữ dữ liệu

Phân loại này đề cập đến các vấn đề bảo mật về cách mà một ứng dụng Android lưu trữ và quản lí dữ liệu người dùng. Hệ thống lưu trữ dữ liệu có thể đối mặt với nhiều vấn đề bảo mật khác nhau đe dọa đến tính bảo mật, tính toàn vẹn và khả dụng của dữ liệu được lưu trữ. Chúng ta có ba cách lưu dữ liệu trên thiết bị di động: bộ nhớ trong, bộ nhớ ngoài và nhà cung cấp nội dung (content provider). Dữ liệu lưu trữ bên ngoài là một cách lưu trữ phổ biến cho các ứng dụng di động. Những vấn đề bảo mật dễ xảy ra và xuất phát từ việc lưu trữ dữ liệu nhạy cảm trong khi không tích hợp sẵn quyền kiểm soát truy cập.

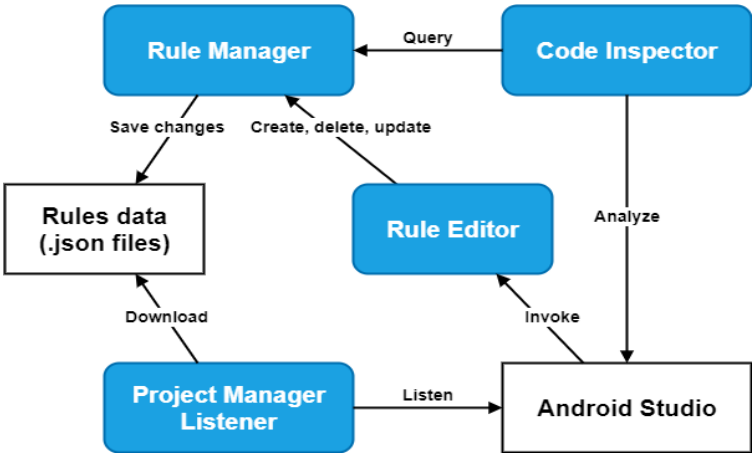
2.2.4. Các vấn đề bảo mật liên quan đến giao tiếp liên tiến trình (Interprocess communication - IPC)

Trong Android, cơ chế Giao tiếp liên tiến trình (IPC) được sử dụng để các tiến trình khác nhau trong hệ thống Android trao đổi dữ liệu và tương tác với nhau. Android cung cấp một số cơ chế IPC như Intents, Binders và Content Providers. Để xác minh danh tính của ứng dụng đang kết nối với IPC và đặt chính sách bảo mật cho từng cơ chế IPC, Android sử dụng mô hình quyền hạn và kiểm soát quyền truy cập của ứng dụng. Các sự cố liên lạc giữa các tiến trình có thể dẫn đến truy cập trái phép vào dữ liệu và thực thi mã. Điều quan trọng là cần tuân thủ những giải pháp được đưa ra để đảm bảo an toàn trong quá trình IPC. Đầu tiên, sử dụng các giao thức liên lạc an toàn để giao tiếp mạng giữa các quá trình như HTTPS, SSL/TLS. Thứ hai, đảm bảo rằng các tiến trình và ứng dụng IPC xác thực và chứng thực lẫn nhau trước khi truy cập vào dữ liệu nhạy cảm. Tiếp theo, kiểm tra đầu vào của người dùng để chắc chắn không có mã độc nào xâm nhập vào hệ thống, đồng thời giới hạn quyền truy cập vào dữ liệu và chức năng nhạy cảm.

2.3. Plugin hỗ trợ lập trình an toàn cho Android Studio

Android Studio cho phép mở rộng tính năng IDE theo nhiều cách khác nhau, trong đó có hai kiểu mở rộng phù hợp với công cụ mà nhóm nghiên cứu đã lên kế hoạch phát triển. Đầu tiên, ta có thể mở rộng thêm giao diện người dùng của IDE như MenuBar và ToolBar, cung cấp cho lập trình viên một phương tiện để tương tác với plugin. Thứ hai, các nhà phát triển bên thứ ba có thể mở rộng tính năng phân tích mã nguồn tự động của Android Studio

để có thể truy soát và sửa chữa các đoạn mã nguồn lỗi nằm ngoài bộ quy tắc mặc định của IDE. Android Studio có thể làm điều này nhờ vào *Program Structure Interface (PSI)*. PSI cung cấp rất nhiều chức năng, từ điều hướng nhanh đến tệp, kí hiệu đến sửa chữa nhanh và tái cấu trúc mã nguồn. Sử dụng các tính năng tiện ích mở rộng mà IDE cung cấp, nhóm đã triển khai một plugin lập trình an toàn cho Android Studio, có tên là **ArmorDroid**. Kiến trúc của plugin của nhóm được mô tả trong Hình 2.



Hình 2. *Kiến trúc của ArmorDroid*

Khi người dùng mở một project ứng dụng Android thì mô-đun *Project Manager Listener* sẽ bắt sự kiện này, sau đó hiển thị một hộp thoại hỏi liệu người dùng có muốn ArmorDroid quét lỗi trên project này hay không. Nếu người dùng đồng ý, plugin sẽ tự động tạo một thư mục có tên là *ArmorDroidRuleData* và sẽ tải các tệp lưu trữ bộ quy tắc mặc định về lập trình an toàn vào thư mục. Tương ứng với mỗi loại ngôn ngữ lập trình mà ArmorDroid hỗ trợ, sẽ có một file định dạng JSON để lưu trữ bộ quy tắc riêng dành cho ngôn ngữ đó, bao gồm các tệp tin như sau: java.json, kotlin.json và xml.json. Những tệp này chứa thông tin dùng để nhận diện đoạn mã nguồn xấu và các cách sửa nó. Những tệp này và mô-đun *Code Inspector* sẽ được lưu trữ trên máy người dùng và hoạt động cục bộ, không có dữ liệu nào được gửi ra bên ngoài. Điều giúp đảm bảo tính bí mật cho mã nguồn dự án.

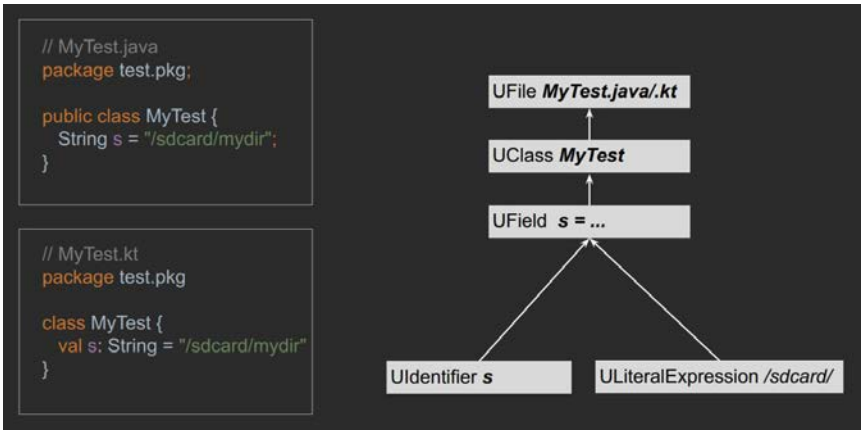
Mô-đun *Rule Editor* có tác dụng cung cấp giao diện trực quan để người dùng thực hiện thao tác chỉnh sửa cũng như thêm và xóa các quy tắc. Để sửa đổi các quy tắc, *Rule Editor* gọi các API được cung cấp bởi mô-đun *Rule Manager*. Rule Manager chịu trách nhiệm đọc các quy tắc từ các tệp tin, tạo, cập nhật và xóa các quy tắc từ các tệp tin này và cung cấp bộ quy tắc phù hợp cho ngôn ngữ lập trình mà Code Inspector yêu cầu. Ngoài ra, mỗi khi có thay đổi được thực hiện trên các tệp quy tắc, Rule Editor sẽ thông báo tới Code Inspector để cập nhật lại.

2.3.1. Code Inspector

Để kiểm tra các quy tắc, công cụ của nhóm sử dụng một API được gọi là Program Structure Interface (PSI). PSI là một API được cung cấp bởi IntelliJ Platform Plugin SDK, có chức năng phân tích tập tin mã nguồn và tạo ra mô hình mã ngữ nghĩa và cú pháp. Khi

lập trình viên viết một đoạn code mới, IDE sẽ xây dựng lại Cây Cú pháp Trừu tượng hay Abstract Syntax Tree (AST) và tính toán các thay đổi so với phiên bản trước đó. Nói một cách đơn giản, AST phân chia đoạn mã nguồn thành các thành phần nhỏ hơn để có thể thao tác với từng thành phần. Những thành phần của AST này tương ứng với các thành phần thông thường được tìm thấy trong nhiều ngôn ngữ lập trình hiện nay. Ví dụ như PsiWhiteSpace sẽ tương ứng với một khoảng trắng, PsiClass tương ứng với một lớp, PsiMethod tương ứng một phương thức trong khai báo của một lớp. Trong đó, lớp PsiElement là lớp cơ sở chung cho các lớp kiểu thành phần.

Mỗi ngôn ngữ có thể có cách triển khai PSI riêng của mình. Java và Kotlin có phiên bản PSI riêng của mình được gọi là Unified Abstract Syntax Tree (UAST), được mô tả trong Hình 3. Mặc dù, mỗi ngôn ngữ chạy trên máy ảo Java (JVM) đều có PSI riêng của mình, nhưng nhiều tính năng như kiểm tra, đánh dấu, tham chiếu và nhiều hơn nữa, hoạt động theo một cách giống nhau trên tất cả các ngôn ngữ JVM này. Việc áp dụng UAST cho phép cung cấp chức năng sẽ hoạt động với một cài đặt duy nhất trên tất cả các ngôn ngữ JVM được hỗ trợ.



Hình 3. *Một ví dụ về UAST cho ngôn ngữ lập trình Java và Koltin*

Đối với XML, nhóm sử dụng API Document Object Model hay XML DOM được cung cấp bởi nền tảng. Có nhiều loại phần tử được định nghĩa bởi XML DOM, nhưng plugin chủ yếu tương tác với XmlTag và XmlAttribute vì cả hai đã cung cấp đủ thông tin để plugin kiểm tra lỗ hổng trong các tệp AndroidManifest.xml. Để duyệt qua tất cả các phần tử trong một tệp nguồn, một đối tượng Visitor sẽ được thêm vào hàm accept() của phần tử ở đầu tiên trong một tệp (thường là một phần tử PsiFile). Visitor là một giao diện chứa các hàm gọi lại, nhận một yếu tố PSI làm đối số. Bên trong các hàm gọi lại này là nơi quá trình kiểm tra mã diễn ra.

2.3.2. Rule Manager

ArmorDroid có một thành phần gọi là Rule Manager chịu trách nhiệm quản lí việc đọc ghi dữ liệu của bộ quy tắc. Các quy tắc được lưu trữ dưới định dạng JavaScript Object Notation (JSON), và mỗi tệp .json được dành riêng cho một ngôn ngữ được hỗ trợ (Java, Kotlin, XML). Rule Manager cung cấp các API để mã hóa và giải mã JSON thành đối tượng thuộc lớp Rule. Các mô-đun khác có thể gọi các API này khi cần để đọc, cập nhật hoặc xóa các đối tượng quy tắc. Kiến trúc của RuleManager dựa trên mẫu thiết kế Singleton. Cùng

một thời điểm chỉ có một đối tượng của RuleManager và đồng thời đối tượng này sẽ lưu các danh sách đối tượng Rule cho mỗi ngôn ngữ. Khi RuleManager đọc các quy tắc từ các tệp JSON lần đầu tiên kể từ khi dự án được mở, các quy tắc được lưu vào bộ nhớ cache trong các danh sách này. Khi các quy tắc được truy cập trong lần tiếp theo, các Rule được lưu trong bộ nhớ cache được truyền thay vì đọc lại từ dưới ổ đĩa. Điều này giúp tiết kiệm thời gian và giảm thiểu các thao tác truy/xuất trên ổ cứng không cần thiết, nhất là khi dữ liệu ngày một lớn do người dùng thêm nhiều quy tắc hơn theo thời gian.

2.3.3. Rule Editor

Một tính năng rất quan trọng của ArmorDroid là khả năng tùy chỉnh các quy tắc. Khả năng tùy chỉnh quy tắc này được kết hợp với Rule Manager giúp thay đổi bộ quy tắc cho từng ngôn ngữ được hỗ trợ (Java, Kotlin, XML) trong tệp dữ liệu lưu trữ bộ quy tắc đó. Nếu chỉnh sửa trực tiếp trên tệp dữ liệu lưu trữ bộ quy tắc sẽ dẫn đến các vấn đề như khó thao tác với người dùng, việc chỉnh sửa có thể gặp các lỗi trong quá trình thực hiện và gây khó khăn cho người dùng. Nền tảng IntelliJ cho phép các plugin thêm các tính năng giao diện người dùng mới vào IDE. Trong ArmorDroid, nhóm thêm một tùy chọn menu mới có tên "ArmorDroid Secure Coding" trong tab Công cụ trên thanh công cụ. Tùy chọn menu này bao gồm 5 chức năng tương ứng với các lựa chọn như sau:

- Customize Rules: Tùy chỉnh bộ quy tắc;
- Export Rules: Xuất bộ quy tắc ra một tệp;
- Import Rules: Nhập bộ quy tắc từ tệp tin được xuất;
- Reset Rules: Đặt lại bộ quy tắc thành mặc định ban đầu;
- Inspect Project: Kiểm tra toàn bộ hoặc một số tệp tin mã nguồn.

2.4. Thực nghiệm sơ bộ và đánh giá

2.4.1. Khảo sát người dùng

Để thực hiện thực nghiệm, nhóm đã mời 43 lập trình viên Android có kinh nghiệm lập trình ứng dụng Android từ vài tháng đến vài năm để giải quyết một bài tập lập trình khi có và khi không có sự hỗ trợ của ArmorDroid để có thể đánh giá giá trị mà ArmorDroid mang lại. Thực nghiệm có 43 người tham gia. Đối tượng tham gia là những sinh viên Khoa Công nghệ Thông tin đến từ Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM và các lập trình viên đến từ các chuyên ngành khác nhau. Bảng 1 cho biết số lượng người tham gia cho từng vị trí.

Bảng 1. *Khảo sát về thông tin lí lịch người tham gia*

Nghề nghiệp	Số lượng
Sinh viên	20
Lập trình viên Android	9
Lập trình viên bảo mật	4
Lập trình viên khác	10

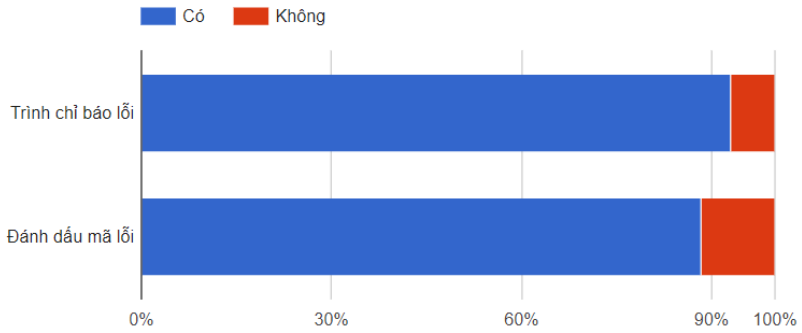
Những sinh viên tham gia khảo sát đều đã từng học qua môn lập trình thiết bị di động Android. Bên cạnh đó, hầu hết những lập trình viên tham gia đều đã có kinh nghiệm lập trình ứng dụng Android từ 3 tháng trở lên.

Bảng 2. *Khảo sát về kinh nghiệm lập trình ứng dụng Android của người tham gia*

Số năm kinh nghiệm lập trình ứng dụng Android	**Số lượng**
< 1 năm	24
1 – 2 năm	14
> 2 năm	5

Theo kết quả khảo sát, hơn 3/4 người tham gia (79,1%) quan tâm đến vấn đề bảo mật trong lập trình. Tuy nhiên, hơn một nửa người tham gia (65,1%) chưa có kinh nghiệm trong lập trình bảo mật. Qua chia sẻ từ người tham gia, hầu hết các bạn sinh viên chỉ bắt đầu học các môn về bảo mật sau khi chọn chuyên ngành từ năm ba và thông qua lí thuyết là phần lớn. Những người lập trình viên tham gia hầu như chưa được đào tạo kĩ về lập trình an toàn trong phát triển ứng dụng Android nói riêng và lập trình nói chung. Qua đó, phản ánh thực trạng chung của việc áp dụng lập trình an toàn vào việc lập trình ứng dụng Android nói riêng và lập trình nói chung.

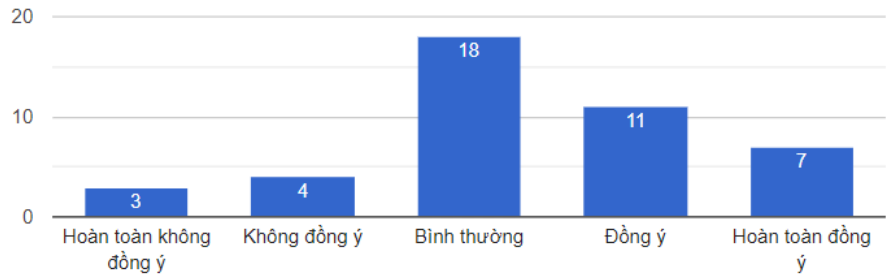
Lỗi bảo mật khi được ArmorDroid phát hiện sẽ được hiển thị cảnh báo theo hai cách: Đoạn mã lỗi sẽ được đánh dấu và hiển thị lỗi lên trình chỉ báo lỗi.



Hình 4. *Khảo sát đánh giá tính năng Kiểm tra mã nguồn và hiển thị cảnh báo theo 2 cách*

Qua các thông tin trên, Hình 4 mô tả hiệu quả của tính năng kiểm tra mã nguồn của ArmorDroid thông qua thực nghiệm bởi những người tham gia. Biểu đồ trong hình này cho thấy tỉ lệ phát hiện lỗi bảo mật trong mã nguồn trong quá trình viết mã nguồn của plugin tương đối cao. Plugin đã hiển thị mã nguồn có lỗ hổng trong trình chỉ báo lỗi và cũng làm nổi bật trên dòng code bị lỗi. Việc này giúp người dùng dễ dàng dễ dàng tìm thấy vị trí mã nguồn gây lỗi bảo mật.

Khi phát hiện mã nguồn có lỗ hổng, plugin sẽ hiển thị tên của lỗi đó và một mô tả ngắn gọn để người dùng có thể hiểu được lỗi này. Tuy nhiên, theo kết quả Hình 5, việc hiểu lỗi thông qua mô tả ngắn của bộ quy tắc không đạt được kết quả như mong đợi. Người tham gia đã đề xuất rằng mô tả ngắn gọn nên bổ sung thêm một liên kết đến tài liệu liên quan với lỗi này để giúp người dùng hiểu rõ hơn về lỗi.



Hình 5. *Khảo sát đánh giá mức độ hiểu của người tham gia về lỗi bảo mật thông qua mô tả ngắn về quy tắc*

Tính năng sửa nhanh (Quick fix) là một lựa chọn phổ biến giữa những người tham gia. Theo kết quả khảo sát, hơn hai phần ba (81,4%) số người tham gia cho biết họ đã sử dụng tính năng này trong suốt thời kì thử nghiệm. Kết quả cho thấy đa số người tham gia đã sử dụng sửa nhanh và đạt được kết quả tích cực.

Khoảng 90% số người tham gia báo cáo rằng Quản lí bộ quy tắc rất dễ sử dụng. Tuy nhiên, một phần nhỏ số người tham gia vẫn không hiểu rõ về các quy tắc và đề xuất cải thiện giao diện người dùng của Quản lí bộ quy tắc. Ngoài ra, nhóm tác giả hỏi người tham gia liệu việc tùy chỉnh bộ quy tắc có dễ hay không. Hơn một nửa số người tham gia (58,1%) nghĩ rằng việc tùy chỉnh bộ quy tắc khá khó khăn. Điều này cho thấy rằng cần phải cải tiến giao diện người dùng của Quản lí bộ quy tắc cho phát triển trong tương lai.

2.4.2. So sánh hiệu suất với Sonarlint

Nhóm đã tiến hành so sánh hiệu suất giữa ArmorDroid và Sonarlint trên một máy tính xách tay chạy hệ điều hành Windows 11, với 16 GB RAM và bộ vi xử lí Intel Core i5-8300H. Thử nghiệm đầu tiên tập trung vào tốc độ phát hiện, trong khi thử nghiệm thứ hai tập trung vào việc sử dụng bộ nhớ.

Đối với thử nghiệm về tốc độ, nhóm chạy kiểm tra trên 30 mẫu mã nguồn lỗi và đo thời gian mà mỗi plugin mất để phát hiện chúng. Kết quả cho thấy rằng ArmorDroid có tốc độ tương đương với Sonarlint, trong khoảng từ 120 đến 400 ms so với 150 đến 420 ms. Kết quả này chứng minh rằng ArmorDroid đáp ứng kì vọng của nhóm về việc phát hiện mã nguồn lỗi trong thời gian thực.

Đối với thử nghiệm về việc sử dụng bộ nhớ, nhóm sử dụng một kịch bản bằng Python để theo dõi việc sử dụng bộ nhớ của Android Studio mỗi 2 giây trong 2 phút khi thực hiện một nhiệm vụ lập trình tương tự như trong cuộc khảo sát ở phần khảo sát người dùng. Nhóm đã lặp lại thử nghiệm này 10 lần và tính toán trung bình việc sử dụng bộ nhớ cho mỗi điều kiện. Kết quả cho thấy rằng Android Studio mà không có bất kì plugin nào sử dụng 1689.25 MB bộ nhớ, trong khi với ArmorDroid nó sử dụng 1824.24 MB và với Sonarlint nó sử dụng 1892.61 MB. Điều này có nghĩa là ArmorDroid tăng việc sử dụng bộ nhớ khoảng 135.01 MB, trong khi Sonarlint tăng khoảng 203.36 MB. Ban đầu, ArmorDroid sử dụng ít bộ nhớ hơn Sonarlint, nhưng điều này có thể do bộ quy tắc nhỏ hơn và ít tính năng và ngôn ngữ hơn. Sonarlint có nhiều tính năng hơn ArmorDroid, điều này có thể giải thích việc tiêu thụ bộ nhớ cao hơn.

3. Kết luận

Bài báo này giới thiệu ArmorDroid, một plugin cho Android Studio giúp nhà phát triển tránh các vấn đề bảo mật phổ biến trong lập trình ứng dụng trên Android. ArmorDroid phát hiện các mẫu đoạn mã không an toàn trong thời gian thực và đề xuất cách sửa. Nó cũng cho phép người dùng tùy chỉnh và chia sẻ các quy tắc kiểm tra với đồng nghiệp. Nhóm tác giả đã đánh giá ArmorDroid với các nhà phát triển ứng dụng Android và phát hiện rằng họ đánh giá cao khả năng của công cụ trong việc xác định và sửa lỗi đoạn mã có lỗ hổng một cách nhanh chóng và dễ dàng. Họ cũng thấy rằng tính năng chỉnh sửa quy tắc rất hữu ích.

Khảo sát tiết lộ một số điểm cần cải thiện trong ArmorDroid hiện tại. Đầu tiên, giao diện người dùng của trình sown thảo quy tắc cần được cải thiện. Người dùng cho rằng cần phải làm rõ cách sử dụng quy tác hoặc các mẫu regex. Một vấn đề khác là khả năng của trình kiểm tra mã không thể kiểm tra các đối số hàm biến do ArmorDroid có sự hiểu biết hạn chế về ngữ cảnh của đoạn mã.

❖ ***Tuyên bố về quyền lợi:*** *Các tác giả xác nhận hoàn toàn không có xung đột về quyền lợi.*

❖ ***Lời cảm ơn:*** *This research is funded by the University of Science, VNU-HCM, Vietnam under grant number CNTT 2023-05.*

TÀI LIỆU THAM KHẢO

Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., & Mcdaniel, P. (2014). FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*.

De Cremer, P., Desmet, N., Madou, M., & De Sutter, B. (2020). Sensei: Enforcing secure coding guidelines in the integrated development environment. *Software: Practice and Experience, 50*(9), 1682-1718. Wiley Online Library.

Egele, M., Brumley, D., Fratantonio, Y., & Kruegel, C. (2013). An empirical study of cryptographic misuse in android applications. *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp.73-84).

Fan, W., Zhang, D., Chen, Y.-g., Wu, F., & Liu, Y. (2020). EstiDroid: Estimate API Calls of Android Applications Using Static Analysis Technology. *IEEE Access, 8,* 105384-105398.

Moore, R., & Lopes, J. (1999). Paper templates. *In TEMPLATE'06, 1st International Conference on Template Production*. Scitepress.

Nguyen, D. C., Wermke, D., Acar, Y., Backes, M., Weir, C., & Fahl, S. (2017). A stitch in time: Supporting Android developers in writing secure code. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp.1065-1077).

Pan, L., Cui, B., Yan, J., Ma, X., Yan, J., & Zhang, J. (2019). Androlic: an extensible flow, context, object, field, and path-sensitive static analysis framework for Android. *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp.394-397).

Senanayake, J., Kalutarage, H., Al-Kadri, M. O., Petrovski, A., & Piras, L. (2023). Android source code vulnerability detection: a systematic literature review. *ACM Computing Surveys*, *55*(9), 1-37.

Smith, J. (1998). *The Book* (2nd ed.). The Publishing Company.

Statista. (2023). *Quarterly number of mobile app downloads worldwide from 1st quarter 2016 to 4th quarter 2022*. https://www.statista.com/statistics/695094/quarterly-number-of-mobile-app-downloads-store/

Talukder, M. A. I., Shahriar, H., Qian, K., Rahman, M., Ahamed, S., Wu, F., & Agu, E. (2019). DroidPatrol: a static analysis plugin for secure mobile software development. *2019 IEEE 43rd annual computer software and applications conference (COMPSAC), 1*, 565-569. IEEE.

Tebib, M. E. A., Graa, M., & Andre, P. (2023). A survey on secure android apps development life-cycle: Vulnerabilities and tools. *International Journal On Advances in Security*, *16*(1 & 2), 54-71.

Tran, A.-D., Nguyen, M.-Q., Phan, G.-H., & Tran, M.-T. (2021). Security Issues in Android Application Development and Plug-in for Android Studio to Support Secure Programming. In *Future Data and Security Engineering. Big Data, Security and Privacy, Smart City and Industry 4.0 Applications: 8th International Conference, FDSE 2021, Virtual Event, November 24--26, 2021, Proceedings 8* (pp. 105-122). Springer.

UAST - Unified Abstract Syntax Tree. (2023). *UAST - Unified Abstract Syntax Tree. https://plugins.jetbrains.com/docs/intellij/uast.html*

DEVELOPMENT OF A CUSTOMIZABLE SECURE CODING PLUGIN FOR ANDROID APPLICATION DEVELOPMENT

Le Cong Binh, Nguyen Le Bao Thi, Truong Phuoc Loc*, Tran Minh Triet, Tran Anh Duy

University of Science, Vietnam National University Ho Chi Minh City, Vietnam

**Corresponding author: Truong Phuoc Loc – Email: tploc@fit.hcmus.edu.vn*

Received: November 27, 2023; Revised: March 31, 2024; Accepted: April 01, 2024

ABSTRACT

Even though Android is a popular mobile operating system, its application ecosystem is still facing many security issues. Lack of awareness and proper attention to security issues during Android application development is considered one of the main causes leading to this situation. This research was conducted to propose a tool to support secure programming. Aligned with the philosophy of DevSecOps, we prioritize placing the developer at the core of the process, aiming to optimize the solution by integrating secure programming practices from the earliest stages of software development.

This article presents two main contributions of the research: synthesizing and classifying security issues in Android application development, along with developing the ArmorDroid tool - a plugin for Android Studio that supports secure programming. This plugin can detect vulnerabilities in source code instantly and suggest modifications during development. It works with Java, Kotlin, and XML files. This plugin provides secure programming standards for Android app development and also educates developers on writing secure code. Developers can also customize rules to fit specific needs and share them with the other programmer community. Our work also presents results from a preliminary study on the effectiveness of the ArmorDroid plugin.

Keywords: Android Studio plugin; DevSecOps; Secure Coding