

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC TẬP
TỐT NGHIỆP ĐẠI HỌC

***Đề tài:* NGHIÊN CỨU PHÁT HIỆN CÁC LỖ**
HÔNG OWASP MASVS TRÊN ỨNG DỤNG
ANDROID

Giáo viên hướng dẫn : THẠC SĨ LÊ HÀ THANH
Sinh viên thực hiện : BÙI ĐỨC PHÚ
Mã số sinh viên : N20DCAT042
Lớp : D20CQAT01 - N
Khóa : 2020 - 2025
Hệ : ĐẠI HỌC CHÍNH QUY

TPHCM, tháng 8 năm 2024

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**



BÁO CÁO ĐỒ ÁN THỰC TẬP TỐT NGHIỆP

***Đề tài:* NGHIÊN CỨU PHÁT HIỆN CÁC LỖ
HỔNG OWASP MASVS TRÊN ỨNG DỤNG
ANDROID**

| | | |
|----------------------------|----------|----------------------------|
| Giáo viên hướng dẫn | : | THẠC SĨ LÊ HÀ THANH |
| Sinh viên thực hiện | : | BÙI ĐỨC PHÚ |
| Mã số sinh viên | : | N20DCAT042 |
| Lớp | : | D20CQAT01-N |
| Khóa | : | 2020 – 2025 |
| Hệ | : | ĐẠI HỌC CHÍNH QUY |

TPHCM, tháng 8 năm 2024

LỜI CẢM ƠN

Trong suốt thời gian 4 năm rưỡi học tập và rèn luyện tại Học viện Công nghệ Bưu chính Viễn thông Cơ sở tại TP. Hồ Chí Minh cho đến nay, em đã nhận được nhiều sự quan tâm, giúp đỡ của quý Thầy Cô và bạn bè. Với lòng biết ơn sâu sắc và chân thành nhất, em xin gửi lời cảm ơn đến quý Thầy Cô ở Khoa Công nghệ thông tin 2 – Học viện Công nghệ Bưu chính Viễn thông Cơ sở tại TP. Hồ Chí Minh đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho em trong suốt thời gian học tập tại trường.

Trong học kỳ này, Khoa đã tổ chức cho em được thực hiện đề tài thực tập tốt nghiệp:

“Nghiên cứu phát hiện các lỗ hổng OWASP MASVS trên ứng dụng Android”

Đối với em đây là đề tài rất hữu ích cho việc tổng hợp và vận dụng toàn bộ kiến thức đã lĩnh hội được trong suốt 4 năm rưỡi học tập tại nhà trường.

Em xin chân trọng gửi lời cảm ơn đặc biệt sâu sắc tới thầy Lê Hà Thanh – giáo viên hướng dẫn của em trong suốt thời gian làm đồ án thực tập tốt nghiệp – đã truyền lửa và tận tâm hướng dẫn chúng em qua từng buổi học trên lớp cũng như trong thời gian làm đồ án thực tập tốt nghiệp này.

Đối với em, được thầy hướng dẫn thực hiện đồ án này là tài sản vô giá mà em sẽ đem theo bên mình như hành trang trong suốt quãng đường phía trước của mình.

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

Em xin chân trọng cảm ơn!

MỤC LỤC

| | |
|---|------|
| LỜI CẢM ƠN | i |
| MỤC LỤC | ii |
| DANH MỤC CÁC HÌNH | vi |
| KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT | viii |
| LỜI MỞ ĐẦU | 1 |
| 1. Tổng quan tình hình nghiên cứu..... | 1 |
| 1. Lý do chọn đề tài | 1 |
| 2. Mục tiêu đề tài | 1 |
| 3. Phương pháp nghiên cứu..... | 1 |
| 4. Đối tượng nghiên cứu..... | 1 |
| CHƯƠNG 1 : CƠ SỞ LÝ THUYẾT | 3 |
| 1.1 GIỚI THIỆU VỀ HỆ THỐNG ANDROID | 3 |
| 1.1.1 Hệ điều hành Android | 3 |
| 1.1.2 Lịch sử | 3 |
| 1.1.3 Giao diện | 6 |
| 1.1.4 Ứng dụng | 7 |
| 1.1.5 Bộ nhớ | 8 |
| 1.1.6 Tùy chọn nhà phát triển | 8 |
| 1.1.7 Các tính năng bảo mật kỹ thuật | 8 |
| 1.1.8 Các tính năng bảo mật kỹ thuật | 9 |
| 1.2 CẤU TRÚC CỦA HỆ ĐIỀU HÀNH ANDROID | 10 |
| 1.2.1 Lớp Linux Kernel trong Android | 10 |
| 1.2.2 Libraries trong Android | 11 |
| 1.2.3 Android Libraries | 11 |
| 1.2.4 Android Runtime | 11 |
| 1.2.5 Application Framework | 11 |
| 1.2.6 Applications..... | 12 |
| 1.3 TIÊU CHUẨN OWASP SASVS (2022)..... | 12 |
| 1.3.1 ASVS có hai mục tiêu chính: | 12 |
| 1.3.2 Các mức Application Security Verification | 13 |
| 1.3.3 Sử dụng tiêu chuẩn ASVS như thế nào | 13 |
| 1.3.4 V1 - Architecture, Design and Threat Modeling Requirements..... | 14 |
| 1.3.5 V2 - Data Storage and Privacy Requirements | 15 |
| 1.3.6 V3 - Cryptography Requirements..... | 16 |

| | | |
|--|---|-----------|
| 1.3.7 | V4 - Authentication and Session Management Requirements | 17 |
| 1.3.8 | V5 - Network Communication Requirements | 17 |
| 1.3.9 | V6 - Platform Interaction Requirements | 18 |
| 1.3.10 | V7 - Code Quality and Build Setting Requirements | 19 |
| 1.3.11 | V8 - Resilience Requirements | 19 |
| CHƯƠNG 2 : NGHIÊN CỨU LỖ HỔNG ANDROID VỚI TIÊU CHUẨN OWASP SASVS 21 | | |
| 3.1 | SQL INJECTION..... | 21 |
| 2.1.1 | Giới thiệu..... | 21 |
| 2.1.2 | Mô tả lỗ hổng SQL Injection..... | 21 |
| 2.1.3 | Mức độ tác động..... | 21 |
| 2.1.4 | Khả năng khai thác | 21 |
| 2.1.5 | Hậu quả..... | 21 |
| 2.1.6 | Đánh giá lỗ hổng theo OWASP SASVS | 21 |
| 2.1.7 | Biện pháp phòng ngừa chi tiết..... | 22 |
| 2.1.8 | Tổng kết..... | 23 |
| 3.2 | Eavesdropping (Man-In-The-Middle Attacks)..... | 23 |
| 3.2.1 | Giới Thiệu..... | 23 |
| 3.2.2 | Mô Tả Lỗ Hổng Eavesdropping (Man-In-The-Middle Attacks)..... | 23 |
| 3.2.3 | Mức Độ Tác Động..... | 23 |
| 3.2.4 | Khả Năng Khai Thác | 23 |
| 3.2.5 | Hậu Quả..... | 24 |
| 3.2.6 | Đánh Giá Theo OWASP MASVS..... | 24 |
| 3.2.7 | Biện Pháp Phòng Ngừa Chi Tiết | 25 |
| 3.2.8 | Tổng Kết..... | 25 |
| 3.3 | Hard-Coded Keys/Tokens/Secrets/URL | 25 |
| 3.3.1 | Giới Thiệu..... | 25 |
| 3.3.2 | Mô Tả Lỗ Hổng Hard-Coded Keys/Tokens/Secrets | 25 |
| 3.3.3 | Mức Độ Tác Động..... | 26 |
| 3.3.4 | Khả Năng Khai Thác | 26 |
| 3.3.5 | Hậu Quả..... | 26 |
| 3.3.6 | Đánh Giá theo OWASP MASVS | 26 |
| 3.3.7 | Biện Pháp Phòng Ngừa Chi Tiết | 28 |
| 3.3.8 | Tổng Kết..... | 28 |
| 3.4 | File Integrity Checks..... | 28 |

| | | |
|---|---|-----------|
| 3.4.1 | Giới Thiệu..... | 28 |
| 3.4.2 | Mô Tả Lỗi Hồng Kiểm Tra Tính Toàn Vẹn của Tập | 28 |
| 3.4.3 | Mức Độ Tác Động | 29 |
| 3.4.4 | Khả Năng Khai Thác | 29 |
| 3.4.5 | Hậu Quả..... | 29 |
| 3.4.6 | Đánh Giá theo OWASP MASVS | 29 |
| 3.4.7 | Biện Pháp Khắc Phục Chi Tiết..... | 31 |
| 3.4.8 | Tổng Kết..... | 32 |
| CHƯƠNG 3 : XÂY DỰNG CÔNG CỤ VÀ PENTES ỨNG DỤNG | | 33 |
| 3.1 | giới thiệu về công cụ Pentest. | 33 |
| 3.1.1 | cơ chế hoạt động..... | 33 |
| 3.1.2 | Giới thiệu về tool..... | 33 |
| 3.1.3 | Cài đặt công cụ..... | 33 |
| 3.2 | Pentest và đưa ra kết quả | 34 |
| 3.2.1 | Kịch bản..... | 34 |
| 3.2.2 | Pentest SQL Injection..... | 34 |
| 3.2.3 | Pentest Eavesdropping (Man-In-The-Middle Attacks) | 39 |
| 3.2.4 | Pentest Hard-Coded Keys/Tokens/Secrets/URL | 45 |
| 3.2.5 | Pentest file Integrity Checks..... | 52 |
| KẾT LUẬN | | 59 |
| TÀI LIỆU THAM KHẢO..... | | 60 |

DANH MỤC BẢNG

| | |
|--|----|
| BẢNG 2.1 Đánh Giá SQL Injection..... | 21 |
| BẢNG 2.2 Đánh giá Eavesdropping (Man-In-The-Middle) | 24 |
| BẢNG 2.3 Đánh giá hard-coded keys/tokens/secrets/URL | 26 |
| BẢNG 2.4 File Integrity Checks | 29 |

DANH MỤC CÁC HÌNH

| | |
|--|----|
| Hình 1.1 Eric Schmidt, Andy Rubin và Hugo Barra tại cuộc họp báo năm 2012 thông báo về máy tính bảng Nexus 7 của Google..... | 5 |
| Hình 1.2 Cấu trúc của ứng dụng Android | 10 |
| Hình 1.3 Các mức độ 4.0 của Tiêu chuẩn xác minh bảo mật ứng dụng OWASP..... | 13 |
| Hình 2.1 Sử dụng Prepared Statements để ngăn chặn SQL Injection | 22 |
| Hình 2.2 Kiểm tra dữ liệu đầu vào bằng cách sử dụng Regex | 22 |
| Hình 3.1 Sơ đồ hoạt động của công cụ..... | 33 |
| Hình 3.2 Khởi chạy công cụ với --help | 34 |
| Hình 3.3 Thực hiện login..... | 35 |
| Hình 3.4 SQL nhận được..... | 35 |
| Hình 3.5 Chuẩn bị APK để Pentest | 36 |
| Hình 3.6 Đăng Pentest | 36 |
| Hình 3.7 Cảnh báo SQL Injection | 36 |
| Hình 3.8 Hàm gây ra lỗi hỏng | 37 |
| Hình 3.9 Khắc phục lỗi hỏng..... | 38 |
| Hình 3.10 Kết quả Pentest lần 2 | 38 |
| Hình 3.11 Set proxy cho thiết bị Android | 40 |
| Hình 3.12 Set proxy cho Burp Suite..... | 40 |
| Hình 3.13 Chuẩn bị đăng nhập | 41 |
| Hình 3.14 Đăng nhập..... | 41 |
| Hình 3.15 Burp Suite nghe lén thành công..... | 42 |
| Hình 3.16 Kết quả Pentest MitM..... | 42 |
| Hình 3.17 Certificates..... | 43 |
| Hình 3.18 Certificates in client..... | 43 |
| Hình 3.19 Kết quả Pentest lần 2 | 43 |
| Hình 3.20 Test đăng nhập..... | 44 |
| Hình 3.21 Decompiler.com | 45 |
| Hình 3.22 Dịch ngược thành công..... | 46 |
| Hình 3.23 File string.xml..... | 46 |
| Hình 3.24 Path file SignUp.java..... | 47 |
| Hình 3.25 Hàm signUp..... | 47 |
| Hình 3.26 Model User | 47 |
| Hình 3.27 Call api postman | 48 |
| Hình 3.28 Call api thành công | 48 |
| Hình 3.29 Kết quả Pentest Hand-coded..... | 49 |
| Hình 3.30 Thông tin quan trọng bị lộ | 49 |
| Hình 3.31 Khắc phục loại bỏ key word trong string.xml | 50 |

| | |
|--|----|
| Hình 3.32 Kết quả Pentest hand-coded lần 2..... | 50 |
| Hình 3.33 Khởi động MT Manager..... | 52 |
| Hình 3.34 Mở app cần crack | 53 |
| Hình 3.35 Mở 2 file classes.dex và classes2.dex..... | 54 |
| Hình 3.36 Sửa đổi URL..... | 54 |
| Hình 3.37 Ký lại ứng dụng | 55 |
| Hình 3.38 Chặn bị server với port 3002 | 55 |
| Hình 3.39 Test login | 56 |
| Hình 3.40 Kết quả khi login | 56 |
| Hình 3.41 Lệnh test tính toàn vẹn của apk | 56 |
| Hình 3.42 Kết quả Pentest tính toàn vẹn của apk..... | 57 |
| Hình 3.43 Pentest với 2 file giống nhau | 57 |

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

MASVS: Mobile Application Security Verification Standard

Fraunhofer AISEC: Fraunhofer Institute for Applied and Integrated Security

CWE: Common Weakness Enumeration

MITM : Man-In-The-Middle

LỜI MỞ ĐẦU

1. Tổng quan tình hình nghiên cứu

Trong những năm gần đây, việc sử dụng các ứng dụng di động ngày càng phổ biến, kéo theo đó là sự gia tăng các mối đe dọa bảo mật đối với các ứng dụng này. Đặc biệt, các lỗ hổng bảo mật trong ứng dụng di động có thể bị khai thác để tấn công, đánh cắp dữ liệu và gây thiệt hại lớn về tài chính cũng như uy tín cho các cá nhân và tổ chức.

Tiêu chuẩn MASVS (Mobile Application Security Verification Standard) của OWASP (Open Web Application Security Project) đã trở thành một tiêu chuẩn phổ biến trong việc xác định và xử lý các lỗ hổng bảo mật trên ứng dụng di động. MASVS cung cấp một khung làm việc toàn diện để đánh giá và cải thiện bảo mật cho các ứng dụng di động, giúp các nhà phát triển và chuyên gia bảo mật phát hiện và khắc phục các lỗ hổng bảo mật một cách hiệu quả.

1. Lý do chọn đề tài

Từ thực tế trên, em chọn đề tài "Nghiên cứu phát hiện các lỗ hổng OWASP MASVS trên ứng dụng Android" để nghiên cứu. Đề tài này không chỉ có ý nghĩa khoa học mà còn mang tính thực tiễn cao, giúp nâng cao khả năng bảo mật cho các ứng dụng di động, từ đó bảo vệ thông tin cá nhân và dữ liệu quan trọng của người dùng.

2. Mục tiêu đề tài

Mục tiêu của đề tài là xây dựng một phương pháp hiệu quả để phát hiện các lỗ hổng bảo mật theo tiêu chuẩn OWASP MASVS trên các ứng dụng Android. Phương pháp này sẽ được thử nghiệm và đánh giá trên nhiều ứng dụng khác nhau, nhằm đảm bảo khả năng phát hiện và khắc phục các lỗ hổng bảo mật một cách toàn diện và hiệu quả.

3. Phương pháp nghiên cứu

Em sẽ sử dụng phương pháp nghiên cứu định lượng để xây dựng và đánh giá phương pháp phát hiện lỗ hổng bảo mật.

- **Nghiên cứu các kiến thức nền tảng:** Tìm hiểu về tiêu chuẩn OWASP MASVS, các lỗ hổng bảo mật phổ biến trên ứng dụng Android, các phương pháp kiểm tra và đánh giá bảo mật.
- **Thu thập và xử lý dữ liệu:** Sử dụng các công cụ và phương pháp để phân tích bảo mật các ứng dụng Android, thu thập thông tin về các lỗ hổng bảo mật.
- **Xây dựng phương pháp:** Thiết kế và xây dựng phương pháp phát hiện lỗ hổng bảo mật theo tiêu chuẩn OWASP MASVS, lựa chọn các công cụ và kỹ thuật phù hợp.
- **Thử nghiệm và đánh giá phương pháp:** Áp dụng phương pháp trên các ứng dụng Android, đánh giá hiệu suất của phương pháp dựa trên các tiêu chí như độ chính xác, tính toàn diện và khả năng phát hiện các lỗ hổng mới.

4. Đối tượng nghiên cứu

Đối tượng nghiên cứu chính của đề tài là các lỗ hổng bảo mật theo tiêu chuẩn OWASP MASVS trên các ứng dụng Android, bao gồm các lỗ hổng liên quan đến:

- **Permission:** Các lỗ hổng liên quan đến việc sử dụng quyền hạn không đúng cách hoặc quá mức, chẳng hạn như quyền truy cập vào các tài nguyên hệ thống nhạy cảm mà không có lý do chính đáng.

- **Data:** Các lỗ hổng liên quan đến việc lưu trữ và xử lý dữ liệu không an toàn, bao gồm việc lưu trữ dữ liệu nhạy cảm không được mã hóa hoặc truyền tải dữ liệu nhạy cảm mà không có biện pháp bảo vệ thích hợp.
- **Network:** Các lỗ hổng liên quan đến giao tiếp mạng không an toàn, chẳng hạn như truyền dữ liệu qua các kết nối không được mã hóa, sử dụng các giao thức không an toàn hoặc thiếu các biện pháp bảo vệ trước các cuộc tấn công mạng như Man-In-The-Middle (MITM).

Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế của sinh viên, nghiên cứu này không thể tránh được những thiếu sót. Em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy, các cô để em có điều kiện bổ sung, nâng cao và phục vụ tốt hơn trong công việc sau này.

Em xin chân thành cảm ơn.

CHƯƠNG 1 : CƠ SỞ LÝ THUYẾT

1.1 GIỚI THIỆU VỀ HỆ THỐNG ANDROID

1.1.1 Hệ điều hành Android

Android là một hệ điều hành dựa trên nền tảng Linux được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng. Ban đầu, Android được phát triển bởi Android Inc. với sự hỗ trợ tài chính từ Google và sau đó Google đã mua lại vào năm 2005.

Android ra mắt vào năm 2007 cùng với tuyên bố thành lập Liên minh thiết bị cầm tay mở: một hiệp hội gồm các công ty phần cứng, phần mềm, và viễn thông với mục tiêu đẩy mạnh các tiêu chuẩn mở cho các thiết bị di động. Chiếc điện thoại đầu tiên chạy Android được bán vào năm 2008.

Android có mã nguồn mở và Google phát hành mã nguồn theo Giấy phép Apache. Chính mã nguồn mở cùng với một giấy phép không có nhiều ràng buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên nhiệt huyết được điều chỉnh và phân phối Android một cách tự do. Ngoài ra, Android còn có một cộng đồng lập trình viên đông đảo chuyên viết các ứng dụng để mở rộng chức năng của thiết bị, bằng một loại ngôn ngữ lập trình Java có sửa đổi. Tháng 10 năm 2012, có khoảng 700.000 ứng dụng trên Android, và số lượt tải ứng dụng từ Google Play, cửa hàng ứng dụng chính của Android, ước tính khoảng 25 tỷ lượt.

Những yếu tố này đã giúp Android trở thành nền tảng điện thoại thông minh phổ biến nhất thế giới, vượt qua Symbian OS vào quý 4 năm 2010, và được các công ty công nghệ lựa chọn khi họ cần một hệ điều hành không nặng nề, có khả năng tinh chỉnh, và giá rẻ chạy trên các thiết bị công nghệ cao thay vì tạo dựng từ đầu. Kết quả là mặc dù được thiết kế để chạy trên điện thoại và máy tính bảng, Android đã xuất hiện trên TV, máy chơi game và các thiết bị điện tử khác. Bản chất mở của Android cũng khích lệ một đội ngũ đông đảo lập trình viên và những người đam mê sử dụng mã nguồn mở để tạo ra những dự án do cộng đồng quản lý. Những dự án này bổ sung các tính năng cao cấp cho những người dùng thích tìm tòi hoặc đưa Android vào các thiết bị ban đầu chạy hệ điều hành khác.

Android chiếm 87,7% thị phần điện thoại thông minh trên toàn thế giới vào thời điểm quý 2 năm 2017, với tổng cộng 2 tỷ thiết bị đã được kích hoạt và 1,3 triệu lượt kích hoạt mỗi ngày. Sự thành công của hệ điều hành cũng khiến nó trở thành mục tiêu trong các vụ kiện liên quan đến bằng phát minh, góp mặt trong cái gọi là "cuộc chiến điện thoại thông minh" giữa các công ty công nghệ.

Từ năm 2011, Android đã là hệ điều hành bán chạy nhất trên toàn cầu trên điện thoại thông minh và từ năm 2013 trên máy tính bảng. Tính đến tháng 5 năm 2021, nó có hơn ba tỷ người dùng hàng tháng, là hệ điều hành có cài đặt nhiều nhất trên thế giới, và tính đến tháng 1 năm 2021, Cửa hàng Google Play có hơn 3 triệu ứng dụng. Android 13, được phát hành vào ngày 15 tháng 8 năm 2022, là phiên bản mới nhất, và phiên bản Android 12.1/12L mới nhất bao gồm những cải tiến đặc biệt cho điện thoại gập, máy tính bảng, màn hình có kích thước lớn như máy tính để bàn 1080p và Chromebook [1].

1.1.2 Lịch sử

Android, Inc. được thành lập tại Palo Alto, California vào tháng 10 năm 2003 bởi Andy Rubin (đồng sáng lập công ty Danger), Rich Miner (đồng sáng lập Tổng công ty Viễn thông Wildfire),

Nick Sears (từng là Phó giám đốc T-Mobile), và Chris White (trưởng thiết kế và giao diện tại WebTV) để phát triển, theo lời của Rubin, "các thiết bị di động thông minh hơn có thể biết được vị trí và sở thích của người dùng". Dù những người thành lập và nhân viên đều là những người có tiếng tăm, Android, Inc. hoạt động một cách âm thầm, chỉ tiết lộ rằng họ đang làm phần mềm dành cho điện thoại di động. Trong năm đó, Rubin hết kinh phí. Steve Perlman, một người bạn thân của Rubin, mang cho ông 10.000 USD tiền mặt nhưng từ chối tham gia vào công ty.

Google mua lại Android, Inc. vào ngày 17 tháng 8 năm 2005, biến nó thành một bộ phận trực thuộc Google. Những nhân viên chủ chốt của Android, Inc., gồm Rubin, Miner và White, vẫn tiếp tục ở lại công ty làm việc sau thương vụ này. Vào thời điểm đó không có nhiều thông tin về công ty, nhưng nhiều người đồn đoán rằng Google dự tính tham gia thị trường điện thoại di động sau bước đi này. Tại Google, nhóm do Rubin đứng đầu đã phát triển một nền tảng thiết bị di động phát triển trên nền nhân Linux. Google quảng bá nền tảng này cho các nhà sản xuất điện thoại và các nhà mạng với lời hứa sẽ cung cấp một hệ thống uyển chuyển và có khả năng nâng cấp. Google đã liên hệ với hàng loạt hãng phần cứng cũng như đối tác phần mềm, bắn tin cho các nhà mạng rằng họ sẵn sàng hợp tác với các cấp độ khác nhau.

Ngày càng nhiều suy đoán rằng Google sẽ tham gia thị trường điện thoại di động kể từ tháng 12 năm 2006. Tin tức của BBC và Nhật báo phố Wall chú thích rằng Google muốn đưa công nghệ tìm kiếm và các ứng dụng của họ vào điện thoại di động và họ đang nỗ lực làm việc để thực hiện điều này. Các phương tiện truyền thông truyền thống lẫn online cũng viết về tin đồn rằng Google đang phát triển một thiết bị cầm tay mang thương hiệu Google. Một vài tờ báo còn nói rằng trong khi Google vẫn đang thực hiện những bản mô tả kỹ thuật chi tiết, họ đã trình diễn sản phẩm mẫu cho các nhà sản xuất điện thoại di động và nhà mạng. Tháng 9 năm 2007, InformationWeek đăng tải một nghiên cứu của Evaluateserve cho biết Google đã nộp một số đơn xin cấp bằng sáng chế trong lĩnh vực điện thoại di động.

Ngày 5 tháng 11 năm 2007, Liên minh thiết bị cầm tay mở (Open Handset Alliance), một hiệp hội bao gồm nhiều công ty trong đó có Texas Instruments, Tập đoàn Broadcom, Google, HTC, Intel, LG, Tập đoàn Marvell Technology, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel và T-Mobile được thành lập với mục đích phát triển các tiêu chuẩn mở cho thiết bị di động. Cùng ngày, Android cũng được ra mắt với vai trò là sản phẩm đầu tiên của Liên minh, một nền tảng thiết bị di động được xây dựng trên nhân Linux phiên bản 2.6. Chiếc điện thoại chạy Android đầu tiên được bán ra là HTC Dream, phát hành ngày 22 tháng 10 năm 2008. Biểu trưng của hệ điều hành Android mới là một con robot màu xanh lá cây do hãng thiết kế Irina Blok tại California vẽ.

Từ năm 2008, Android đã trải qua nhiều lần cập nhật để dần dần cải tiến hệ điều hành, bổ sung các tính năng mới và sửa các lỗi trong những lần phát hành trước. Mỗi bản nâng cấp được đặt tên lần lượt theo thứ tự bảng chữ cái, theo tên của một món ăn tráng miệng; ví dụ như phiên bản 1.5 Cupcake (bánh bông lan nhỏ có kem) tiếp nối bằng phiên bản 1.6. Phiên bản mới nhất hiện nay là Android 13, ra mắt vào tháng 8 năm 2022. Vào năm 2010, Google ra mắt loạt thiết bị Nexus—một dòng sản phẩm bao gồm điện thoại thông minh và máy tính bảng chạy hệ điều hành Android, do các đối tác phần cứng sản xuất. HTC đã hợp tác với Google trong chiếc điện thoại thông minh Nexus đầu tiên, Nexus One. Kể từ đó nhiều thiết bị mới hơn đã gia nhập vào dòng sản phẩm này, như điện thoại Nexus 4 và máy tính bảng Nexus 10, lần lượt do LG và

Samsung sản xuất. Google xem điện thoại và máy tính bảng Nexus là những thiết bị Android chủ lực của mình, với những tính năng phần cứng và phần mềm mới nhất của Android.

Năm 2010, Google ra mắt dòng sản phẩm Nexus, một loạt thiết bị mà Google hợp tác với các nhà sản xuất thiết bị khác nhau để sản xuất các thiết bị mới và giới thiệu các phiên bản Android mới. Dòng sản phẩm này được mô tả là đã "đóng vai trò then chốt trong lịch sử của Android bằng việc giới thiệu các phiên bản phần mềm và tiêu chuẩn phần cứng mới trên toàn bộ hệ thống" và trở nên nổi tiếng với phần mềm "không quá nhiều chức năng" và "cập nhật kịp thời". Tại hội nghị phát triển Google I/O vào tháng 5 năm 2013, Google công bố một phiên bản đặc biệt của Samsung Galaxy S4, trong đó thay vì sử dụng phiên bản tùy chỉnh Android của Samsung, điện thoại chạy "stock Android" và hứa hẹn sẽ nhận được các cập nhật hệ thống mới nhanh chóng. Thiết bị này trở thành khởi đầu của chương trình Google Play edition, và được tiếp tục bởi các thiết bị khác, bao gồm phiên bản Google Play edition của HTC One, và Moto G Google Play edition. Năm 2015, Ars Technica viết rằng "Đầu tuần này, các điện thoại Android phiên bản Google Play edition cuối cùng trong cửa hàng trực tuyến của Google đã được liệt kê là "không còn bán" và "Bây giờ chúng đã biến mất, và có vẻ như chương trình đã kết thúc hoàn toàn."

Từ năm 2008 đến 2013, Hugo Barra đã đảm nhận vai trò người phát ngôn sản phẩm, đại diện cho Android tại các cuộc họp báo và Google I/O, hội nghị hàng năm dành cho nhà phát triển của Google. Ông rời khỏi Google vào tháng 8 năm 2013 để gia nhập hãng điện thoại Trung Quốc Xiaomi. Chưa đầy sáu tháng trước đó, Larry Page, khi đó là CEO của Google, thông báo trong một bài đăng trên blog rằng Andy Rubin đã chuyển từ bộ phận Android để đảm nhận các dự án mới tại Google, và Sundar Pichai sẽ trở thành người đứng đầu Android mới. Pichai sau đó cũng thay đổi vị trí, trở thành CEO mới của Google vào tháng 8 năm 2015 sau khi công ty tái cơ cấu thành tập đoàn Alphabet, khiến Hiroshi Lockheimer trở thành người đứng đầu Android mới. Trên Android 4.4 Kit Kat, quyền truy cập viết chung vào thẻ nhớ MicroSD đã bị khóa đối với các ứng dụng cài đặt bởi người dùng, chỉ có thể ghi vào các thư mục chuyên dụng với tên gói tương ứng, nằm trong đường dẫn Android/data/. Quyền truy cập viết đã được khôi phục trong Android 5 Lollipop thông qua Google Storage Access Framework interface không tương thích ngược [1].



Hình 1.1 Eric Schmidt, Andy Rubin và Hugo Barra tại cuộc họp báo năm 2012 thông báo về máy tính bảng Nexus 7 của Google

Vào tháng 6 năm 2014, Google công bố Android One, một bộ "mô hình tham chiếu phần cứng" giúp "cho phép [nhà sản xuất thiết bị] dễ dàng tạo ra điện thoại chất lượng cao với giá thấp", được thiết kế dành cho người tiêu dùng ở các nước đang phát triển. Vào tháng 9, Google công

bộ điện thoại Android One đầu tiên sẽ ra mắt tại Ấn Độ. Tuy nhiên, Recode đưa tin vào tháng 6 năm 2015 rằng dự án này "thất bại", trích dẫn "người tiêu dùng và các đối tác sản xuất không chịu tham gia" và "những lần lỡ tay của công ty tìm kiếm chưa từng thành công với phần cứng". Kế hoạch tái khởi động Android One nổi lên vào tháng 8 năm 2015, và một tuần sau đó, châu Phi được công bố là địa điểm tiếp theo cho chương trình này. Một báo cáo từ The Information vào tháng 1 năm 2017 cho biết Google đang mở rộng chương trình Android One giá rẻ tới Hoa Kỳ, mặc dù The Verge cho biết rằng công ty có thể sẽ không sản xuất các thiết bị thực tế. Google giới thiệu các điện thoại thông minh Pixel và Pixel XL vào tháng 10 năm 2016, được tiếp thị là những chiếc điện thoại đầu tiên do Google sản xuất, và được trang bị các tính năng phần mềm đặc biệt, chẳng hạn như Trợ lý Google, trước khi được phổ biến rộng rãi. Các điện thoại Pixel đã thay thế dòng Nexus, và thế hệ mới của điện thoại Pixel được ra mắt vào tháng 10 năm 2017.

Vào tháng 5 năm 2019, hệ điều hành Android đã vướng vào cuộc chiến thương mại giữa Trung Quốc và Hoa Kỳ liên quan đến Huawei, một công ty công nghệ khác, đã trở nên phụ thuộc vào việc truy cập vào nền tảng Android. Vào mùa hè năm 2019, Huawei thông báo rằng họ sẽ tạo ra một hệ điều hành thay thế cho Android được biết đến với tên gọi Harmony OS, và đã đăng ký quyền sở hữu trí tuệ tại các thị trường toàn cầu chính. Dưới các lệnh trừng phạt như vậy, Huawei có kế hoạch dài hạn để thay thế Android vào năm 2022 bằng hệ điều hành mới này, vì Harmony OS ban đầu được thiết kế cho các thiết bị mạng lưới các vật liệu kết nối (internet of things), chứ không phải cho điện thoại thông minh và máy tính bảng.

Vào ngày 22 tháng 8, 2019, thông báo rằng Android "Q" sẽ chính thức được đặt tên là Android 10, chấm dứt việc đặt tên các phiên bản chính sau các loại món tráng miệng. Google cho biết rằng những tên này không "bao hàm" cho người dùng quốc tế (do những món ăn trên không được biết đến quốc tế, hoặc khó để phát âm trong một số ngôn ngữ). Cùng ngày đó, Android Police đưa tin rằng Google đã thuê chế tác một tượng điêu khắc số "10" khổng lồ để đặt trong sảnh của văn phòng mới của các nhà phát triển. Android 10 được phát hành vào ngày 3 tháng 9, 2019, trước tiên dành cho điện thoại Google Pixel.

Vào cuối năm 2021, một số người dùng báo cáo rằng họ không thể gọi các dịch vụ khẩn cấp. Vấn đề này là do sự kết hợp của các lỗi trên Android và ứng dụng Microsoft Teams; cả hai công ty đã phát hành các bản cập nhật để khắc phục vấn đề [1].

1.1.3 Giao diện

Giao diện người dùng của Android dựa trên nguyên tắc tác động trực tiếp, sử dụng cảm ứng chạm tương tự như những động tác ngoài đời thực như vuốt, chạm, kéo giãn và thu lại để xử lý các đối tượng trên màn hình. Sự phản ứng với tác động của người dùng diễn ra gần như ngay lập tức, nhằm tạo ra giao diện cảm ứng mượt mà, thường dùng tính năng rung của thiết bị để tạo phản hồi rung cho người dùng. Những thiết bị phản cứng bên trong như gia tốc kế, con quay hồi chuyển và cảm biến khoảng cách được một số ứng dụng sử dụng để phản hồi một số hành động khác của người dùng, ví dụ như điều chỉnh màn hình từ chế độ hiển thị dọc sang chế độ hiển thị ngang tùy theo vị trí của thiết bị, hoặc cho phép người dùng lái xe đua bằng xoay thiết bị, giống như đang điều khiển vô-lăng.

Các thiết bị Android sau khi khởi động sẽ hiển thị màn hình chính, điểm khởi đầu với các thông tin chính trên thiết bị, tương tự như khái niệm desktop (bàn làm việc) trên máy tính để bàn. Màn hình chính Android thường gồm nhiều biểu tượng (icon) và tiện ích (widget); biểu tượng ứng

dụng sẽ mở ứng dụng tương ứng, còn tiện ích hiển thị những nội dung sống động, cập nhật tự động như dự báo thời tiết, hộp thư của người dùng, hoặc những mẫu tin thời sự ngay trên màn hình chính. Màn hình chính có thể gồm nhiều trang xem được bằng cách vuốt ra trước hoặc sau, mặc dù giao diện màn hình chính của Android có thể tùy chỉnh ở mức cao, cho phép người dùng tự do sắp đặt hình dáng cũng như hành vi của thiết bị theo sở thích. Những ứng dụng do các hãng thứ ba có trên Google Play và các kho ứng dụng khác còn cho phép người dùng thay đổi "chủ đề" của màn hình chính, thậm chí bắt chước hình dáng của hệ điều hành khác như Windows Phone chẳng hạn. Phần lớn những nhà sản xuất, và một số nhà mạng, thực hiện thay đổi hình dáng và hành vi của các thiết bị Android của họ để phân biệt với các hãng cạnh tranh. Ở phía trên cùng màn hình là thanh trạng thái, hiển thị thông tin về thiết bị và tình trạng kết nối. Thanh trạng thái này có thể "kéo" xuống để xem màn hình thông báo gồm thông tin quan trọng hoặc cập nhật của các ứng dụng, như email hay tin nhắn SMS mới nhận, mà không làm gián đoạn hoặc khiến người dùng cảm thấy bất tiện. Trong các phiên bản đầu tiên, người dùng có thể nhấn vào thông báo để mở ra ứng dụng tương ứng, về sau này các thông tin cập nhật được bổ sung thêm tính năng, như có khả năng lập tức gọi ngược lại khi có cuộc gọi nhỡ mà không cần phải mở ứng dụng gọi điện ra. Thông báo sẽ luôn nằm đó cho đến khi người dùng đã đọc hoặc xóa nó đi.

1.1.4 Ứng dụng

Android có lượng ứng dụng của bên thứ ba ngày càng nhiều, được chọn lọc và đặt trên một cửa hàng ứng dụng như Google Play hay Amazon Appstore để người dùng lấy về, hoặc bằng cách tải xuống rồi cài đặt tập tin "APK" từ trang web khác. Các ứng dụng trên Play Store cho phép người dùng duyệt, tải về và cập nhật các ứng dụng do Google và các nhà phát triển thứ ba phát hành. Play Store được cài đặt sẵn trên các thiết bị thỏa mãn điều kiện tương thích của Google. Ứng dụng sẽ tự động lọc ra một danh sách các ứng dụng tương thích với thiết bị của người dùng, và nhà phát triển có thể giới hạn ứng dụng của họ chỉ dành cho những nhà mạng cố định hoặc những quốc gia cố định vì lý do kinh doanh. Nếu người dùng mua một ứng dụng mà họ cảm thấy không thích, họ được hoàn trả tiền sau 15 phút kể từ lúc tải về, và một vài nhà mạng còn có khả năng mua giúp các ứng dụng trên Google Play, sau đó tính tiền vào trong hóa đơn sử dụng hàng tháng của người dùng. Đến tháng 9 năm 2012, có hơn 675.000 ứng dụng dành cho Android, và số lượng ứng dụng tải về từ Play Store ước tính đạt 25 tỷ.

Các ứng dụng cho Android được phát triển bằng ngôn ngữ Java sử dụng Bộ phát triển phần mềm Android (SDK). SDK bao gồm một bộ đầy đủ các công cụ dùng để phát triển, gồm có công cụ gỡ lỗi, thư viện phần mềm, bộ giả lập điện thoại dựa trên QEMU, tài liệu hướng dẫn, mã nguồn mẫu, và hướng dẫn từng bước. Môi trường phát triển tích hợp (IDE) được hỗ trợ chính thức là Eclipse sử dụng phần bổ sung Android Development Tools (ADT). Các công cụ phát triển khác cũng có sẵn, gồm có Bộ phát triển gốc dành cho các ứng dụng hoặc phần mở rộng viết bằng C hoặc C++, Google App Inventor, một môi trường đồ họa cho những nhà lập trình mới bắt đầu, và nhiều nền tảng ứng dụng web di động đa nền tảng phong phú.

Để vượt qua những hạn chế khi tiếp cận các dịch vụ của Google do sự Kiểm duyệt Internet tại Cộng hòa Nhân dân Trung Hoa, các thiết bị Android bán tại Trung Quốc lục địa thường được điều chỉnh chỉ được sử dụng dịch vụ đã được duyệt.

1.1.5 Bộ nhớ

Bộ nhớ của các thiết bị Android có thể được mở rộng bằng các thiết bị phụ như thẻ SD. Android nhận dạng hai loại bộ nhớ phụ: bộ nhớ di động (được sử dụng theo mặc định) và bộ nhớ có thể áp dụng. Bộ nhớ di động được coi là một thiết bị lưu trữ bên ngoài. Bộ nhớ có thể áp dụng, được giới thiệu trên Android 6.0, cho phép bộ nhớ trong của thiết bị được mở rộng bằng thẻ SD, coi nó như một phần mở rộng của bộ nhớ trong. Điều này có nhược điểm là ngăn không cho thẻ nhớ được sử dụng với thiết bị khác trừ khi nó được định dạng lại.

Android 4.4 đã giới thiệu Storage Access Framework (SAF), một bộ API để truy cập các tệp trên hệ thống tệp của thiết bị. Kể từ Android 11, Android yêu cầu các ứng dụng phải tuân theo chính sách bảo mật dữ liệu được gọi là scoped storage, theo đó các ứng dụng chỉ có thể tự động truy cập vào một số thư mục nhất định (chẳng hạn như thư mục dành cho hình ảnh, nhạc và video) và các thư mục dành riêng cho ứng dụng mà chúng đã tạo. Các ứng dụng được yêu cầu sử dụng SAF để truy cập vào bất kỳ phần nào khác của hệ thống tệp.

1.1.6 Tùy chọn nhà phát triển

Một số cài đặt dành cho các nhà phát triển để gỡ lỗi và người dùng nâng cao được đặt trong một trình đơn phụ "Tùy chọn nhà phát triển", chẳng hạn như khả năng làm nổi bật các phần cập nhật của màn hình, hiển thị lớp phủ với trạng thái hiện tại của màn hình cảm ứng, hiển thị các điểm chạm có thể sử dụng trong truyền hình màn hình, thông báo cho người dùng về các quy trình nền không phản hồi với tùy chọn kết thúc chúng ("Hiển thị tất cả ANR", nghĩa là "Ứng dụng không phản hồi"), ngăn cản ứng dụng khách âm thanh Bluetooth điều khiển âm lượng hệ thống ("Tắt âm lượng tuyệt đối") và điều chỉnh thời lượng hoạt ảnh chuyển tiếp hoặc tắt hoàn toàn chúng để tăng tốc độ điều hướng.

Tùy chọn nhà phát triển ban đầu bị ẩn từ Android 4.2 "Jelly Bean" trở đi, nhưng có thể được bật bằng cách nhấn vào số hiệu bản dựng của hệ điều hành trong thông tin thiết bị bảy lần. Để ẩn lại tùy chọn nhà phát triển, bạn cần xóa dữ liệu người dùng cho ứng dụng "Cài đặt", có thể đặt lại một số tùy chọn khác.

1.1.7 Các tính năng bảo mật kỹ thuật

Năm 2018, công ty bảo mật Na Uy Promon đã phát hiện ra một lỗ hổng bảo mật nghiêm trọng trên Android có thể bị khai thác để đánh cắp thông tin đăng nhập, truy cập tin nhắn và theo dõi vị trí, xuất hiện trong tất cả các phiên bản của Android, bao gồm Android 10. Lỗ hổng này xuất phát từ việc khai thác lỗi trong hệ thống đa nhiệm, cho phép ứng dụng độc hại phủ lên các ứng dụng hợp pháp bằng màn hình đăng nhập giả mà người dùng không biết khi nhập thông tin đăng nhập bảo mật. Người dùng cũng có thể bị lừa cấp quyền bổ sung cho các ứng dụng độc hại, sau đó cho phép chúng thực hiện nhiều hoạt động độc hại khác nhau, bao gồm chặn tin nhắn hoặc cuộc gọi và đánh cắp thông tin đăng nhập ngân hàng. Avast Threat Labs cũng phát hiện ra rằng nhiều ứng dụng được cài đặt sẵn trên hàng trăm thiết bị Android mới chứa phần mềm độc hại và phần mềm quảng cáo nguy hiểm. Một số phần mềm độc hại được cài đặt sẵn có thể thực hiện gian lận quảng cáo hoặc thậm chí chiếm đoạt thiết bị chủ của nó.

Vào ngày 5 tháng 8 năm 2020, Twitter đã đăng một bài viết kêu gọi người dùng cập nhật ứng dụng của họ lên phiên bản mới nhất liên quan đến một vấn đề bảo mật cho phép người khác truy cập tin nhắn trực tiếp. Kẻ tấn công có thể dễ dàng sử dụng "quyền hệ thống Android" để lấy thông tin đăng nhập tài khoản để thực hiện việc này. Vấn đề bảo mật chỉ xảy ra với Android

8 (Android Oreo) và Android 9 (Android Pie). Twitter xác nhận rằng việc cập nhật ứng dụng sẽ hạn chế các hành vi đó. [1]

1.1.8 Các tính năng bảo mật kỹ thuật

Các ứng dụng Android chạy trong một sandbox, một khu vực bị cô lập của hệ thống không có quyền truy cập vào phần còn lại của tài nguyên hệ thống, trừ khi người dùng cấp quyền truy cập rõ ràng khi cài đặt ứng dụng. Tuy nhiên, điều này có thể không thể đối với các ứng dụng được cài đặt sẵn. Ví dụ, không thể tắt quyền truy cập micro của ứng dụng camera được cài đặt sẵn mà không tắt hoàn toàn camera. Điều này cũng đúng trong Android phiên bản 7 và 8.

Từ tháng 2 năm 2012, Google đã sử dụng trình quét phần mềm độc hại Google Bouncer để theo dõi và quét các ứng dụng có sẵn trên Google Play. Tính năng "Xác minh ứng dụng" đã được giới thiệu vào tháng 11 năm 2012, như một phần của phiên bản hệ điều hành Android 4.2 "Jelly Bean", để quét tất cả các ứng dụng, cả từ Google Play và từ các nguồn của bên thứ ba, để tìm các hành vi độc hại. Ban đầu, tính năng này chỉ quét các ứng dụng khi cài đặt, nhưng đã được cập nhật vào năm 2014 để quét các ứng dụng "liên tục" và vào năm 2017, tính năng này đã được hiển thị cho người dùng thông qua menu trong Cài đặt.

Trước khi cài đặt ứng dụng, Google Play sẽ hiển thị danh sách các yêu cầu mà ứng dụng cần để hoạt động. Sau khi xem xét các quyền này, người dùng có thể chọn chấp nhận hoặc từ chối chúng, chỉ cài đặt ứng dụng nếu họ chấp nhận. Trong Android 6.0 "Marshmallow", hệ thống quyền đã được thay đổi; các ứng dụng không còn được tự động cấp tất cả các quyền đã chỉ định của chúng tại thời điểm cài đặt. Thay vào đó, một hệ thống chọn tham gia được sử dụng, trong đó người dùng được nhắc cấp hoặc từ chối các quyền riêng lẻ cho ứng dụng khi chúng cần thiết lần đầu tiên. Các ứng dụng ghi nhớ các cấp quyền, có thể được người dùng thu hồi bất kỳ lúc nào. Tuy nhiên, các ứng dụng được cài đặt sẵn không phải lúc nào cũng nằm trong phương pháp tiếp cận này. Trong một số trường hợp, có thể không thể từ chối một số quyền nhất định đối với các ứng dụng được cài đặt sẵn hoặc tắt chúng. Ứng dụng Google Play Services không thể gỡ cài đặt hoặc tắt. Bất kỳ nỗ lực dùng cưỡng bức nào cũng sẽ khiến ứng dụng khởi động lại chính nó. Mô hình quyền mới chỉ được sử dụng bởi các ứng dụng được phát triển cho Marshmallow bằng bộ phát triển phần mềm (SDK) của nó và các ứng dụng cũ hơn sẽ tiếp tục sử dụng phương pháp tiếp cận tất cả hoặc không có gì trước đây. Quyền vẫn có thể bị thu hồi đối với các ứng dụng đó, mặc dù điều này có thể ngăn chúng hoạt động bình thường và một cảnh báo sẽ được hiển thị về hiệu ứng đó.

Vào tháng 9 năm 2014, Jason Nova của Android Authority đã báo cáo về một nghiên cứu của công ty bảo mật Đức Fraunhofer AISEC về phần mềm chống vi-rút và các mối đe dọa phần mềm độc hại trên Android. Nghiên cứu của Fraunhofer AISEC, kiểm tra phần mềm chống vi-rút từ Avast, AVG, Bitdefender, ESET, F-Secure, Kaspersky, Lookout, McAfee (trước đây là Intel Security), Norton, Sophos, và Trend Micro, tiết lộ rằng "các ứng dụng chống vi-rút được thử nghiệm không cung cấp bảo vệ chống lại phần mềm độc hại được tùy chỉnh hoặc các cuộc tấn công có chủ đích" và rằng "các ứng dụng chống vi-rút được thử nghiệm cũng không thể phát hiện ra phần mềm độc hại hoàn toàn chưa biết đến ngày nay nhưng không cố gắng che giấu tính độc hại của nó".

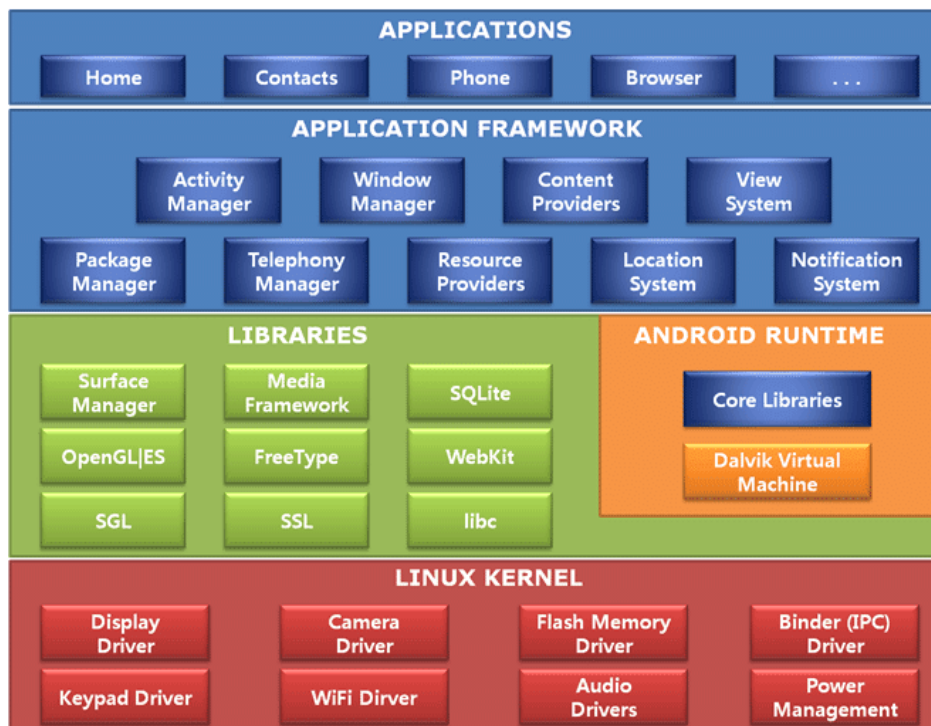
Tháng 8 năm 2013, Google công bố Android Device Manager (đổi tên thành Find My Device vào tháng 5 năm 2017), một dịch vụ cho phép người dùng theo dõi, định vị và xóa thiết bị Android của mình từ xa, với ứng dụng Android cho dịch vụ được phát hành vào tháng 12.

Tháng 12 năm 2016, Google giới thiệu ứng dụng Trusted Contacts, cho phép người dùng yêu cầu theo dõi vị trí của người thân trong trường hợp khẩn cấp. Năm 2020, Trusted Contacts đã bị đóng cửa và tính năng chia sẻ vị trí được tích hợp vào Google Maps.

Vào ngày 8 tháng 10 năm 2018, Google đã công bố các yêu cầu mới cho Google Play Store để chống lại việc chia sẻ quá mức thông tin có khả năng nhạy cảm, bao gồm nhật ký cuộc gọi và tin nhắn văn bản. Vấn đề bắt nguồn từ thực tế là nhiều ứng dụng yêu cầu quyền truy cập vào thông tin cá nhân của người dùng (ngay cả khi thông tin này không cần thiết để ứng dụng hoạt động) và một số người dùng vô điều kiện cấp các quyền này. Ngoài ra, một quyền có thể được liệt kê trong tệp kê khai ứng dụng là bắt buộc (chứ không phải tùy chọn) và ứng dụng sẽ không cài đặt trừ khi người dùng cấp quyền; người dùng có thể thu hồi bất kỳ quyền nào, thậm chí là quyền bắt buộc, từ bất kỳ ứng dụng nào trong cài đặt thiết bị sau khi cài đặt ứng dụng, nhưng ít người dùng làm điều này. Google đã hứa sẽ làm việc với các nhà phát triển và tạo ra các ngoại lệ nếu ứng dụng của họ yêu cầu quyền Điện thoại hoặc SMS cho "chức năng cốt lõi của ứng dụng". Việc thực thi các chính sách mới bắt đầu vào ngày 6 tháng 1 năm 2019, 90 ngày sau khi công bố chính sách vào ngày 8 tháng 10 năm 2018 [1].

1.2 CẤU TRÚC CỦA HỆ ĐIỀU HÀNH ANDROID

Hệ điều hành Android là một ngăn xếp của các thành phần phần mềm mà có thể đại khái phân chia thành 5 khu vực và 4 lớp chính. Hình dưới đây minh họa sơ đồ cấu trúc Android.



Hình 1.2 Cấu trúc của ứng dụng Android

1.2.1 Lớp Linux Kernel trong Android

Linux Kernel là lớp thấp nhất. Nó cung cấp các chức năng cơ bản như quản lý tiến trình, quản lý bộ nhớ, quản lý thiết bị như: Camera, bàn phím, màn hình, ... Ngoài ra, nó còn quản lý mạng, driver của các thiết bị, điều này gỡ bỏ sự khó khăn về giao tiếp với các thiết bị ngoại vi.

1.2.2 Libraries trong Android

Phía trên Linux Kernel là tập hợp các bộ thư viện mã nguồn mở WebKit, bộ thư viện nổi tiếng libc, cơ sở dữ liệu SQLite hữu ích cho việc lưu trữ và chia sẻ dữ liệu, bộ thư viện thể phát, ghi âm về âm thanh, hoặc video. Thư viện SSL chịu trách nhiệm cho bảo mật Internet [2].

1.2.3 Android Libraries

Phần này gồm các thư viện dựa trên Java. Nó bao gồm các Framework Library giúp xây dựng, vẽ đồ họa và truy cập cơ sở dữ liệu trở nên dễ dàng hơn. Dưới đây là một số Android Library cốt lõi có sẵn cho lập trình viên Android:

- **Android.app:** Cung cấp truy cập tới mô hình ứng dụng và nó là nền móng cho tất cả ứng dụng Android.
- **Android.content:** việc truy cập nội dung, các thông điệp giữa các ứng dụng và các thành phần ứng dụng trở nên dễ dàng hơn.
- **Android.database:** được sử dụng để truy cập dữ liệu được công bố bởi Provider và bao gồm các lớp quản lý cơ sở dữ liệu SQLite.
- **Android.opengl:** một Java Interface cho OpenGL ES 3D thông qua API.
- **Android.os:** cung cấp cho các ứng dụng sự truy cập tới các dịch vụ chuẩn của hệ điều hành như thông báo, dịch vụ hệ thống và giao tiếp nội tiến trình.
- **Android.text:** được sử dụng để phục hồi và thao tác text trên một thiết bị hiển thị.
- **Android.view:** các khối kiến trúc nền tảng của ứng dụng UI.
- **Android.widget:** một tập hợp các UI được xây dựng trước như button, label, list view, layout manager, radio button, ...
- **Android.webkit:** một tập hợp các lớp cho phép khả năng để trình duyệt trên web được xây dựng bên trong các ứng dụng.
- Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

1.2.4 Android Runtime

Đây là thành phần thứ 3 trong cấu trúc, thuộc về lớp 2 tính từ dưới lên. Phần này cung cấp một thành phần quan trọng gọi là **Dalvik Virtual Machine** là một máy ảo Java đặc biệt, được thiết kế tối ưu cho Android.

Máy ảo Dalvik VM sử dụng các tính năng cốt lõi của Linux như quản lý bộ nhớ, đa luồng, mà thực chất là bên trong ngôn ngữ Java. Máy ảo Dalvik cho phép tất cả các ứng dụng Android chạy trong tiến trình riêng của nó

Android Runtime cũng cung cấp bộ thư viện cốt lõi, cho phép các lập trình viên Android sử dụng để viết các ứng dụng Android [2].

1.2.5 Application Framework

Lớp Application Framework cung cấp nhiều dịch vụ cấp cao hơn cho các ứng dụng trong các lớp Java. Các lập trình viên cũng được phép sử dụng các dịch vụ này trong các ứng dụng của họ.

Application Framework bao gồm các dịch vụ chính sau:

- **Activity Manager:** điều khiển các khía cạnh của vòng đời ứng dụng và Activity Stack.

- Content Providers: cho phép các ứng dụng công bố và chia sẻ dữ liệu với các ứng dụng khác.
- Resource Manager: cung cấp sự truy cập tới các resource được nhúng (không phải code) như chuỗi, thiết lập màu, UI layout.
- Notifications Manager: cho phép các ứng dụng hiển thị thông báo tới người dùng.
- View System: một tập hợp các view được sử dụng để tạo UI cho ứng dụng.

1.2.6 Applications

Trong hệ điều hành Android, lớp Ứng dụng là lớp trên cùng, nơi người dùng tương tác trực tiếp với các ứng dụng. Tại đây, bạn sẽ tìm thấy tất cả các ứng dụng Android, bao gồm cả các ứng dụng hệ thống được cài sẵn như Danh bạ, Trình duyệt, và Tin nhắn, cũng như các ứng dụng của bên thứ ba mà người dùng có thể tải về từ Google Play Store hoặc các kho ứng dụng khác.

Các ứng dụng trong lớp này có thể bao gồm các ứng dụng giải trí như trò chơi, ứng dụng mạng xã hội, và các ứng dụng văn phòng như xử lý văn bản hoặc bảng tính. Người dùng có thể cài đặt, cập nhật và gỡ bỏ các ứng dụng từ Cài đặt thiết bị hoặc qua các kho ứng dụng trực tuyến.

Khi cài đặt một ứng dụng, người dùng được yêu cầu xem xét và chấp nhận các quyền mà ứng dụng yêu cầu, chẳng hạn như quyền truy cập vào camera, danh bạ, hoặc vị trí. Các ứng dụng có thể yêu cầu quyền này ngay khi cài đặt hoặc trong quá trình sử dụng, và người dùng có quyền từ chối hoặc cấp quyền cho các chức năng cụ thể của ứng dụng.

Ứng dụng cũng tương tác trực tiếp với người dùng thông qua giao diện người dùng, nơi người dùng thực hiện các hành động như nhấp vào nút, điền vào biểu mẫu, và xem thông tin. Giao diện người dùng của ứng dụng có thể được thiết kế để phù hợp với các xu hướng thiết kế hiện hành hoặc theo sở thích của nhà phát triển.

Ngoài ra, các ứng dụng cần phải tuân theo các chính sách bảo mật của Android và đảm bảo quyền riêng tư của người dùng. Các ứng dụng phải bảo vệ thông tin cá nhân và yêu cầu quyền truy cập một cách minh bạch. Để duy trì bảo mật, Google Play Store thường xuyên cập nhật và quét các ứng dụng để phát hiện phần mềm độc hại và các vấn đề bảo mật.

Ứng dụng cũng phải tuân theo các quy định của Android về quyền riêng tư và bảo mật. Từ Android 6.0 (Marshmallow), một hệ thống quyền mới cho phép người dùng cấp hoặc từ chối các quyền riêng lẻ của ứng dụng, giúp người dùng kiểm soát tốt hơn những gì ứng dụng có thể truy cập.

Cuối cùng, các ứng dụng tương tác với các lớp dưới của hệ điều hành, bao gồm Application Framework để sử dụng các API hệ thống và Libraries để truy cập các thư viện phần mềm hỗ trợ chức năng của ứng dụng. Đây là lớp cuối cùng trong chuỗi tương tác của người dùng với hệ điều hành Android, đóng vai trò quan trọng trong việc cung cấp các dịch vụ và chức năng mà người dùng cần cho các hoạt động hàng ngày của mình [2].

1.3 TIÊU CHUẨN OWASP SASVS (2022)

1.3.1 ASVS có hai mục tiêu chính:

- giúp các tổ chức phát triển và duy trì các ứng dụng an toàn.
- cho phép các nhà cung cấp dịch vụ bảo mật, nhà cung cấp công cụ bảo mật và người tiêu dùng điều chỉnh các yêu cầu và dịch vụ của họ.

1.3.2 Các mức Application Security Verification

Tiêu chuẩn xác minh bảo mật ứng dụng xác định ba cấp độ xác minh bảo mật, với mỗi cấp độ tăng dần theo chiều sâu.

- **ASVS Cấp độ 1** dành cho mức độ đảm bảo thấp và hoàn toàn có thể kiểm tra khả năng thâm nhập
- **ASVS Cấp độ 2** dành cho các ứng dụng chứa dữ liệu nhạy cảm, yêu cầu bảo vệ và là cấp độ được khuyến nghị cho hầu hết các ứng dụng
- **ASVS Cấp độ 3** dành cho các ứng dụng quan trọng nhất - các ứng dụng thực hiện các giao dịch có giá trị cao, chứa dữ liệu y tế nhạy cảm hoặc bất kỳ ứng dụng nào yêu cầu mức độ tin cậy cao nhất

Mỗi cấp độ ASVS chứa một danh sách các yêu cầu bảo mật. Mỗi yêu cầu này cũng có thể được ánh xạ tới các tính năng và khả năng dành riêng cho bảo mật mà các nhà phát triển phải tích hợp vào phần mềm [3].

| | Applicability | Building | | | Building, Configuration, Deployment Assurance and Verification | | | Assurance and Verification | |
|---------|----------------|-----------------------------------|---------------|--------------------------|--|-----------|----------------------------|----------------------------|------|
| Level 1 | All apps | | Secure Coding | Standards and checklists | Secure & Peer Code Review | DevSecOps | Unit and Integration Tests | Penetration Testing | DAST |
| Level 2 | All apps | Security Architecture and Reviews | Secure Coding | Standards and checklists | Secure & Peer Code Review | DevSecOps | Unit and Integration Tests | Hybrid Reviews | SAST |
| Level 3 | High Assurance | Security Architecture and Reviews | Secure Coding | Standards and checklists | Secure & Peer Code Review | DevSecOps | Unit and Integration Tests | Hybrid Reviews | SAST |
| Legend | | Acceptable | Suitable | | | | | | |

Hình 1.3 Các mức độ 4.0 của Tiêu chuẩn xác minh bảo mật ứng dụng OWASP

1.3.3 Sử dụng tiêu chuẩn ASVS như thế nào

Level 1 - First steps, automated, or whole of portfolio view

Một ứng dụng đạt được **Cấp độ 1 của ASVS** nếu nó có thể bảo vệ đầy đủ chống lại các lỗ hổng bảo mật phổ biến, như những lỗ hổng được liệt kê trong **OWASP Top 10** và các danh sách kiểm tra tương tự khác. Cấp độ 1 là mức tối thiểu mà tất cả các ứng dụng nên đạt được. Nó đặc biệt hữu ích như một bước khởi đầu trong nỗ lực cải thiện bảo mật nhiều giai đoạn hoặc khi các ứng dụng không lưu trữ hay xử lý dữ liệu nhạy cảm, do đó không yêu cầu các biện pháp bảo vệ nghiêm ngặt hơn của Cấp 2 hoặc 3. Các kiểm tra ở Cấp độ 1 có thể được thực hiện tự động bằng các công cụ hoặc thủ công mà không cần truy cập mã nguồn. Các mối đe dọa đối với ứng dụng ở cấp độ này thường đến từ những kẻ tấn công sử dụng các kỹ thuật đơn giản, dễ thực hiện để tìm và khai thác lỗ hổng. Điều này khác biệt so với một kẻ tấn công quyết tâm, người sẽ dành nhiều năng lượng và tập trung để nhắm mục tiêu cụ thể vào ứng dụng. Nếu ứng dụng của bạn xử lý dữ liệu có giá trị cao, bạn sẽ hiếm khi muốn chỉ dừng lại ở đánh giá Cấp độ 1 [3].

Level 2 - Most applications

Một ứng dụng đạt được **Cấp độ 2 của ASVS** nếu nó có thể bảo vệ đầy đủ trước hầu hết các rủi ro liên quan đến phần mềm hiện nay. **Cấp độ 2** đảm bảo rằng các biện pháp kiểm soát bảo mật được áp dụng, có hiệu lực và được sử dụng trong ứng dụng. Cấp độ này thường thích hợp cho các ứng dụng xử lý các giao dịch quan trọng giữa doanh nghiệp với doanh nghiệp, bao gồm các

ứng dụng xử lý thông tin chăm sóc sức khỏe, thực hiện các chức năng quan trọng hoặc nhạy cảm, hoặc xử lý các tài sản nhạy cảm khác. Trong các ngành mà tính liên chính là khía cạnh quan trọng để bảo vệ doanh nghiệp, chẳng hạn như ngành công nghiệp trò chơi để ngăn chặn gian lận và hack trò chơi, Cấp độ 2 là cần thiết. Các mối đe dọa đối với các ứng dụng ở Cấp độ 2 thường đến từ những kẻ tấn công có kỹ năng và có động cơ, tập trung vào các mục tiêu cụ thể bằng cách sử dụng các công cụ và kỹ thuật cao cấp, hiệu quả trong việc phát hiện và khai thác các điểm yếu trong ứng dụng.

Level 3 - High value, high assurance, or high safety

ASVS Cấp độ 3 là cấp độ xác minh cao nhất trong ASVS, thường được dành cho các ứng dụng yêu cầu mức độ xác minh bảo mật đáng kể, chẳng hạn như trong các lĩnh vực quân sự, y tế, an toàn, cơ sở hạ tầng quan trọng, v.v. Các tổ chức có thể yêu cầu ASVS Cấp độ 3 cho các ứng dụng thực hiện các chức năng quan trọng, trong đó lỗi có thể ảnh hưởng đáng kể đến hoạt động của tổ chức và thậm chí là khả năng tồn tại của tổ chức.

Một ứng dụng đạt được **ASVS Cấp độ 3 (hoặc Nâng cao)** nếu nó bảo vệ đầy đủ chống lại các lỗ hổng bảo mật ứng dụng nâng cao và cũng thể hiện các nguyên tắc của thiết kế bảo mật tốt. Ứng dụng ở cấp độ này yêu cầu phân tích sâu hơn về kiến trúc, mã hóa và thử nghiệm so với tất cả các cấp độ khác. Ứng dụng an toàn cần được mô-đun hóa một cách có ý nghĩa để tạo điều kiện cho khả năng phục hồi, khả năng mở rộng và các lớp bảo mật. Mỗi mô-đun, được phân tách bằng kết nối mạng và/hoặc phiên bản vật lý, đảm nhận trách nhiệm bảo mật riêng (phòng thủ trong độ sâu), và cần phải được lập thành tài liệu thích hợp [3].

Các trách nhiệm này bao gồm các biện pháp kiểm soát để đảm bảo:

- Tính bảo mật (ví dụ: mã hóa)
- Tính toàn vẹn (ví dụ: giao dịch, xác thực đầu vào)
- Tính khả dụng (ví dụ: xử lý tải một cách duyên dáng)
- Xác thực (bao gồm giữa các hệ thống)
- Không từ chối
- Ủy quyền và kiểm tra (ghi nhật ký)

1.3.4 V1 - Architecture, Design and Threat Modeling Requirements

Kiến trúc an ninh đã trở thành một lĩnh vực bị lãng quên trong nhiều tổ chức trong thời đại của DevSecOps. Lĩnh vực bảo mật ứng dụng phải bắt kịp và áp dụng các nguyên tắc bảo mật nhanh trong khi giới thiệu lại các nguyên tắc kiến trúc bảo mật hàng đầu cho những người thực hành phần mềm. Kiến trúc không phải là một cách triển khai, mà là một cách suy nghĩ về một vấn đề có khả năng có nhiều câu trả lời khác nhau và không có một câu trả lời "đúng" nào. Thông thường, bảo mật được coi là không linh hoạt và đòi hỏi các nhà phát triển phải sửa mã theo một cách cụ thể, trong khi các nhà phát triển có thể biết cách tốt hơn để giải quyết vấn đề. Không có giải pháp đơn giản nào cho kiến trúc.

Việc triển khai cụ thể của một ứng dụng web có thể được sửa đổi liên tục trong suốt thời gian tồn tại của nó, nhưng kiến trúc tổng thể sẽ hiếm khi thay đổi và phát triển chậm hơn. Kiến trúc bảo mật cũng tương tự - cần xác thực ngày hôm nay, sẽ yêu cầu xác thực vào ngày mai và trong tương lai. Nếu đưa ra quyết định đúng đắn ngày hôm nay, có thể tiết kiệm rất nhiều công sức, thời gian và tiền bạc nếu lựa chọn và sử dụng lại các giải pháp phù hợp với kiến trúc. Ví dụ, một thập kỷ trước, xác thực đa yếu tố hiếm khi được triển khai.

Nếu các nhà phát triển đã đầu tư vào một mô hình nhà cung cấp danh tính bảo mật, duy nhất, chẳng hạn như danh tính liên kết SAML, thì nhà cung cấp danh tính có thể được cập nhật để kết hợp các yêu cầu mới như tuân thủ NIST 800-63, trong khi không thay đổi giao diện của ứng dụng gốc. Nếu nhiều ứng dụng chia sẻ cùng một kiến trúc bảo mật và cùng một thành phần, tất cả chúng đều được hưởng lợi từ bản nâng cấp này cùng một lúc. Tuy nhiên, SAML sẽ không phải lúc nào cũng là giải pháp xác thực tốt nhất hoặc phù hợp nhất - có thể cần hoán đổi cho các giải pháp khác khi các yêu cầu thay đổi. Những thay đổi như thế này có thể phức tạp và tốn kém đến mức yêu cầu viết lại hoàn toàn hoặc không thể thực hiện được nếu không có kiến trúc bảo mật.

ASVS đề cập đến các khía cạnh chính của bất kỳ kiến trúc bảo mật âm thanh nào: tính khả dụng, tính bảo mật, tính toàn vẹn của quá trình xử lý, tính không từ chối và quyền riêng tư. Mỗi nguyên tắc bảo mật này phải được tích hợp sẵn và là bản chất bẩm sinh cho tất cả các ứng dụng. Điều quan trọng là phải "chuyển sang trái", bắt đầu với sự hỗ trợ của nhà phát triển với danh sách kiểm tra mã hóa an toàn, cố vấn và đào tạo, mã hóa và thử nghiệm, xây dựng, triển khai, cấu hình và hoạt động, và hoàn thiện bằng thử nghiệm độc lập tiếp theo để đảm bảo rằng tất cả các biện pháp kiểm soát bảo mật hiện tại và chức năng.

Bước cuối cùng từng là mọi thứ được làm trong ngành công nghiệp, nhưng điều đó không còn đủ khi các nhà phát triển đẩy mã vào sản xuất hàng chục hoặc hàng trăm lần mỗi ngày. Các chuyên gia bảo mật ứng dụng phải theo kịp các kỹ thuật, có nghĩa là áp dụng các công cụ dành cho nhà phát triển, học cách viết mã và làm việc với các nhà phát triển [3].

Security Verification Requirements:

- **V1.1** - All app components are identified and known to be needed.
- **V1.2** - Security controls are never enforced only on the client side, but on the respective remote endpoints.
- **V1.3** - A high-level architecture for the mobile app and all connected remote services has been defined and security has been addressed in that architecture.
- **V1.4** - Data considered sensitive in the context of the mobile app is clearly identified.
- **V1.5** - All app components are defined in terms of the business functions and/or security functions they provide.
- **V1.6** - A threat model for the mobile app and the associated remote services has been produced that identifies potential threats and countermeasures.
- **V1.7** - All security controls have a centralized implementation.
- **V1.8** - There is an explicit policy for how cryptographic keys (if any) are managed, and the lifecycle of cryptographic keys is enforced. Ideally, follow a key management standard such as NIST SP 800-57.
- **V1.9** - A mechanism for enforcing updates of the mobile app exists.
- **V1.10** - Security is addressed within all parts of the software development lifecycle.
- **V1.11** - A responsible disclosure policy is in place and effectively applied.
- **V1.12** - The app should comply with privacy laws and regulations.

1.3.5 V2 - Data Storage and Privacy Requirements

Xác thực là hành động thiết lập hoặc xác nhận ai đó (hoặc cái gì đó) là xác thực và tuyên bố của một người hoặc về thiết bị là đúng, chống lại việc mạo danh và ngăn chặn việc khôi phục

hoặc đánh chặn mật khẩu. Khi ASVS lần đầu tiên được phát hành, tên người dùng và mật khẩu là hình thức xác thực phổ biến nhất bên ngoài các hệ thống bảo mật cao. Xác thực đa yếu tố (MFA) thường được chấp nhận trong giới bảo mật nhưng hiếm khi được yêu cầu ở những nơi khác.

Khi số lượng vi phạm mật khẩu tăng lên, ý tưởng rằng tên người dùng bằng cách nào đó được bảo mật và mật khẩu không xác định đã khiến nhiều biện pháp kiểm soát bảo mật trở nên không thực tế. Ví dụ, NIST 800-63 coi tên người dùng và xác thực dựa trên kiến thức (KBA) là thông tin công khai, thông báo qua SMS và email là loại trình xác thực "restricted", và mật khẩu như đã vi phạm trước. Thực tế này khiến các trình xác thực dựa trên kiến thức, khôi phục qua SMS và email, lịch sử mật khẩu, độ phức tạp và điều khiển xoay vòng trở nên vô dụng. Các biện pháp kiểm soát này luôn ít hữu ích, thường buộc người dùng phải đưa ra mật khẩu yếu vài tháng một lần, nhưng với việc phát hành hơn 5 tỷ tên người dùng và mật khẩu vi phạm, đã đến lúc phải tiến lên.

Trong tất cả các phần trong ASVS, chương về xác thực và quản lý phiên đã thay đổi nhiều nhất. Việc áp dụng phương pháp hàng đầu dựa trên bằng chứng hiệu quả sẽ là thách thức đối với nhiều người và điều đó hoàn toàn bình thường. Cần bắt đầu chuyển đổi sang một tương lai không dùng mật khẩu ngay bây giờ [3].

Security Verification Requirements:

- **V2.1** - Sensitive data is stored using system credential storage facilities.
- **V2.2** - Sensitive data is not stored outside the app container or system credential storage.
- **V2.3** - Sensitive data is not written to application logs.
- **V2.4** - Sensitive data is not shared with third parties unless necessary.
- **V2.5** - Keyboard cache is disabled for sensitive data input fields.
- **V2.6** - Sensitive data is not exposed via inter-process communication (IPC).
- **V2.7** - Sensitive data, such as passwords or PINs, is not exposed through the user interface.
- **V2.8** - Sensitive data is not included in mobile operating system backups.
- **V2.9** - Sensitive data is removed from views when the app is in the background.
- **V2.10** - Sensitive data is not held in memory longer than necessary and is cleared explicitly.
- **V2.11** - The app enforces a minimum device access security policy (e.g., device passcode).
- **V2.12** - The app educates users about PII and security best practices.
- **V2.13** - Sensitive data is not stored locally unless necessary; retrieve from a remote endpoint when needed.
- **V2.14** - If local storage is required, sensitive data is encrypted with a hardware-backed key requiring authentication.
- **V2.15** - Local storage is wiped after excessive failed authentication attempts.

1.3.6 V3 - Cryptography Requirements

Một trong những thành phần cốt lõi của bất kỳ ứng dụng dựa trên web hoặc API trạng thái nào là cơ chế kiểm soát và duy trì trạng thái cho người dùng hoặc thiết bị tương tác với nó. Quản lý phiên thay đổi giao thức không trạng thái thành trạng thái, điều này rất quan trọng để phân biệt người dùng hoặc thiết bị khác nhau.

Đảm bảo rằng một ứng dụng đã được xác minh đáp ứng các yêu cầu quản lý phiên cấp cao sau:

- Phiên là duy nhất cho mỗi cá nhân và không thể đoán hoặc chia sẻ.
- Phiên bị vô hiệu khi không còn được yêu cầu và hết thời gian chờ trong khoảng thời gian không hoạt động.

Như đã lưu ý trước đây, các yêu cầu này đã được điều chỉnh để trở thành một tập hợp con tuân thủ các điều khiển NIST 800-63b được chọn, tập trung vào các mối đe dọa phổ biến và các điểm yếu xác thực thường bị khai thác. Các yêu cầu xác minh trước đây đã được gỡ bỏ, loại bỏ hoặc trong hầu hết các trường hợp được điều chỉnh để phù hợp nhất với mục đích bắt buộc mà NIST 800-63b yêu cầu [3].

Security Verification Requirements:

- **V3.1** - The app does not solely rely on symmetric cryptography with hardcoded keys for encryption.
- **V3.2** - Proven implementations of cryptographic primitives are used.
- **V3.3** - Cryptographic primitives are appropriate for the use case and configured according to industry best practices.
- **V3.4** - Deprecated cryptographic protocols or algorithms are not used.
- **V3.5** - Cryptographic keys are not reused for multiple purposes.
- **V3.6** - A sufficiently secure random number generator is used for all random value generation.

1.3.7 V4 - Authentication and Session Management Requirements

Ủy quyền là khái niệm chỉ cho phép truy cập vào tài nguyên đối với những người được phép sử dụng chúng. Đảm bảo rằng một ứng dụng đã được xác minh đáp ứng các yêu cầu cấp cao sau:

- Những người truy cập tài nguyên có thông tin xác thực hợp lệ.
- Người dùng được liên kết với một nhóm các vai trò và đặc quyền được xác định rõ ràng.
- Siêu dữ liệu về vai trò và quyền được bảo vệ khỏi phát lại hoặc giả mạo.

Security Verification Requirements:

- **V4.1** - Remote service access requires authentication (e.g., username/password).
- **V4.2** - Stateful sessions use randomly generated identifiers for authentication.
- **V4.3** - Stateless authentication uses securely signed tokens.
- **V4.4** - Logout terminates existing sessions at the remote endpoint.
- **V4.5** - A password policy is enforced at the remote endpoint.
- **V4.6** - The remote endpoint protects against excessive credential submission attempts.
- **V4.7** - Sessions and access tokens expire after inactivity.
- **V4.8** - Biometric authentication unlocks the keychain/keystore (not event-bound).
- **V4.9** - Two-factor authentication (2FA) is consistently enforced at the remote endpoint.
- **V4.10** - Sensitive transactions require step-up authentication.
- **V4.11** - Users are informed of sensitive account activity and can manage devices.
- **V4.12** - Authorization models are defined and enforced at the remote endpoint.

1.3.8 V5 - Network Communication Requirements

Điểm yếu bảo mật phổ biến nhất của ứng dụng web là không thể xác thực đúng đầu vào từ máy khách hoặc môi trường trước khi sử dụng trực tiếp mà không có bất kỳ mã hóa đầu ra nào. Điều

này dẫn đến hầu như tất cả các lỗ hổng quan trọng trong các ứng dụng web như Cross-Site Scripting (XSS), SQL Injection, chèn thông dịch viên, tấn công locale/Unicode, tấn công hệ thống tệp và tràn bộ đệm.

Đảm bảo rằng một ứng dụng đã được xác minh đáp ứng các yêu cầu cấp cao sau:

- Xác thực đầu vào và kiến trúc mã hóa đầu ra có một đường dẫn đã được đồng ý để ngăn chặn các cuộc tấn công tiêm.
- Dữ liệu đầu vào được nhập mạnh mẽ, xác thực, kiểm tra phạm vi hoặc độ dài, hoặc tệ nhất là được làm sạch hoặc lọc.
- Dữ liệu đầu ra được mã hóa hoặc thoát theo ngữ cảnh của dữ liệu càng gần với trình thông dịch càng tốt.

Với kiến trúc ứng dụng web hiện đại, mã hóa đầu ra quan trọng hơn bao giờ hết. Rất khó để cung cấp xác thực đầu vào mạnh mẽ trong một số trường hợp nhất định, vì vậy việc sử dụng API an toàn như truy vấn được tham số hóa, khuôn khổ tạo mẫu tự động thoát hoặc mã hóa đầu ra được lựa chọn cẩn thận là rất quan trọng đối với bảo mật của ứng dụng [3].

Security Verification Requirements:

- **V5.1** - Data transmitted over the network is encrypted using TLS, consistently throughout the app.
- **V5.2** - TLS settings align with current best practices or the closest possible configuration supported by the mobile OS.
- **V5.3** - The app verifies the remote endpoint's X.509 certificate and accepts only certificates signed by trusted Certificate Authorities (CAs).
- **V5.4** - The app uses its own certificate store or implements certificate/public key pinning to prevent connections with endpoints offering different certificates, even if signed by a trusted CA.
- **V5.5** - The app does not rely solely on insecure communication channels (email or SMS) for critical operations like enrollment and account recovery.
- **V5.6** - The app uses only up-to-date connectivity and security libraries.

1.3.9 V6 - Platform Interaction Requirements

Đảm bảo rằng một ứng dụng đã được xác minh đáp ứng các yêu cầu cấp cao sau:

- Tất cả các mô-đun mật mã không thành công theo cách an toàn và các lỗi được xử lý đúng cách.
- Một bộ tạo số ngẫu nhiên thích hợp được sử dụng.
- Quyền truy cập vào các khóa được quản lý an toàn.

Security Verification Requirements:

- **V6.1** - The app requests only the minimum necessary permissions.
- **V6.2** - All inputs from external sources and users are validated and sanitized.
- **V6.3** - Sensitive functionality is not exported via custom URL schemes unless properly protected.
- **V6.4** - Sensitive functionality is not exported through IPC mechanisms unless properly protected.
- **V6.5** - JavaScript is disabled in WebViews unless explicitly required.

- **V6.6** - WebViews allow only the minimum necessary protocol handlers (ideally, only HTTPS).
- **V6.7** - If native methods are exposed to a WebView, it renders only JavaScript within the app package.
- **V6.8** - Object deserialization, if used, is implemented with safe serialization APIs.
- **V6.9** - The app is protected against screen overlay attacks (Android only).
- **V6.10** - WebView cache, storage, and resources are cleared before destruction.
- **V6.11** - The app prevents custom third-party keyboards when entering sensitive data (iOS only).

1.3.10 V7 - Code Quality and Build Setting Requirements

Mục tiêu chính của việc ghi nhật ký và xử lý lỗi là cung cấp thông tin hữu ích cho người dùng, quản trị viên và nhóm ứng phó sự cố. Mục tiêu không phải là tạo ra một lượng lớn nhật ký, mà là nhật ký chất lượng cao, với nhiều tín hiệu hơn là tiếng ồn bị loại bỏ. Nhật ký chất lượng cao thường chứa dữ liệu nhạy cảm và phải được bảo vệ theo luật hoặc chỉ thị về quyền riêng tư dữ liệu địa phương. Điều này nên bao gồm:

- Không thu thập hoặc ghi lại thông tin nhạy cảm trừ khi được yêu cầu cụ thể.
- Đảm bảo tất cả thông tin đã ghi được xử lý an toàn và được bảo vệ theo phân loại dữ liệu của nó.
- Đảm bảo rằng nhật ký không được lưu trữ mãi mãi, nhưng có thời gian tồn tại tuyệt đối càng ngắn càng tốt.

Nếu nhật ký chứa dữ liệu riêng tư hoặc nhạy cảm, định nghĩa về dữ liệu đó khác nhau giữa các quốc gia, thì nhật ký sẽ trở thành một số thông tin nhạy cảm nhất được ứng dụng nắm giữ và do đó rất hấp dẫn đối với những kẻ tấn công. Điều quan trọng nữa là đảm bảo rằng ứng dụng bị lỗi một cách an toàn và các lỗi không tiết lộ thông tin không cần thiết [3].

Security Verification Requirements:

- **V7.1** - The app is signed and provisioned with a valid certificate, and the private key is securely protected.
- **V7.2** - The app is built in release mode with appropriate settings (e.g., non-debuggable).
- **V7.3** - Debugging symbols are removed from native binaries.
- **V7.4** - Debugging and developer assistance code (e.g., test code, backdoors) are removed, and verbose error logging is disabled.
- **V7.5** - Third-party components are identified and checked for known vulnerabilities.
- **V7.6** - The app catches and handles potential exceptions.
- **V7.7** - Error handling in security controls denies access by default.
- **V7.8** - Memory in unmanaged code is allocated, freed, and used securely.
- **V7.9** - Free security features offered by the toolchain (e.g., byte-code minification, stack protection) are activated.

1.3.11 V8 - Resilience Requirements

Có ba yếu tố chính để bảo vệ dữ liệu: Tính bảo mật, Tính toàn vẹn và Tính khả dụng (CIA). Tiêu chuẩn này giả định rằng việc bảo vệ dữ liệu được thực thi trên một hệ thống đáng tin cậy, chẳng hạn như một máy chủ, hệ thống này đã được tăng cường và có đủ các biện pháp bảo vệ.

Các ứng dụng phải giả định rằng tất cả các thiết bị của người dùng đều bị xâm phạm theo một cách nào đó. Trong trường hợp ứng dụng truyền hoặc lưu trữ thông tin nhạy cảm trên các thiết bị không an toàn, chẳng hạn như máy tính, điện thoại và máy tính bảng dùng chung, ứng dụng có trách nhiệm đảm bảo dữ liệu được lưu trữ trên các thiết bị này được mã hóa và không thể dễ dàng lấy, thay đổi hoặc tiết lộ một cách bất hợp pháp [3].

Đảm bảo rằng một ứng dụng đã được xác minh đáp ứng các yêu cầu bảo vệ dữ liệu cấp cao sau đây:

- Tính bảo mật: Dữ liệu cần được bảo vệ khỏi sự quan sát hoặc tiết lộ trái phép cả khi vận chuyển và khi được lưu trữ.
- Tính toàn vẹn: Dữ liệu phải được bảo vệ khỏi bị những kẻ tấn công trái phép tạo, thay đổi hoặc xóa một cách ác ý.
- Tính khả dụng: Dữ liệu phải có sẵn cho người dùng được ủy quyền theo yêu cầu.

Security Verification Requirements:

- **V8.1** - Detects and responds to rooted/jailbroken devices (alert or termination).
- **V8.2** - Prevents debugging and/or detects and responds to attached debuggers.
- **V8.3** - Detects and responds to tampering with executable files and critical data within the app's sandbox.
- **V8.4** - Detects and responds to the presence of common reverse engineering tools.
- **V8.5** - Detects and responds to being run in an emulator environment.
- **V8.6** - Detects and responds to tampering with code and data in its memory space.
- **V8.7** - Implements multiple mechanisms in each defense category (V8.1-V8.6).
- **V8.8** - Detection mechanisms trigger diverse responses, including delayed and stealthy ones.
- **V8.9** - Obfuscation is applied to programmatic defenses to impede dynamic analysis.

CHƯƠNG 2 : NGHIÊN CỨU LỖ HỒNG ANDROID VỚI TIÊU CHUẨN OWASP SASVS

3.1 SQL INJECTION

2.1.1 Giới thiệu

SQL Injection (CWE-89) là một trong những lỗ hổng bảo mật phổ biến và nghiêm trọng nhất, ảnh hưởng đến các ứng dụng web và di động. Lỗ hổng này xảy ra khi dữ liệu đầu vào của người dùng không được kiểm tra hoặc làm sạch trước khi sử dụng trong các câu lệnh SQL, cho phép kẻ tấn công chen và thực thi các đoạn mã SQL độc hại.

2.1.2 Mô tả lỗ hổng SQL Injection

- **Mã lỗ hổng:** CWE-89
- **Mô tả:** SQL Injection xảy ra khi ứng dụng không kiểm tra hoặc làm sạch các tham số đầu vào, cho phép kẻ tấn công chen các đoạn mã SQL độc hại vào các câu lệnh SQL, dẫn đến việc thực thi các câu lệnh không mong muốn trên cơ sở dữ liệu.
- **Hình thái hoạt động:** Kẻ tấn công chen các đoạn mã SQL vào các tham số của ứng dụng, chẳng hạn như các trường nhập liệu, để thực thi các câu lệnh SQL không mong muốn.
- **Môi trường:** Lỗ hổng này ảnh hưởng đến các ứng dụng di động sử dụng cơ sở dữ liệu hoặc các kết nối dữ liệu với máy chủ SQL.
- **Đối tượng tác động:** Bất kỳ ứng dụng di động nào có chức năng tương tác với cơ sở dữ liệu và không kiểm tra hoặc bảo vệ chống lại SQL Injection đều có thể bị ảnh hưởng.

2.1.3 Mức độ tác động

Cao: SQL Injection cho phép kẻ tấn công truy cập, sửa đổi hoặc xóa dữ liệu trong cơ sở dữ liệu, gây ra mất dữ liệu, rò rỉ thông tin nhạy cảm và kiểm soát hệ thống.

2.1.4 Khả năng khai thác

Cao: SQL Injection dễ khai thác nếu không có biện pháp phòng ngừa thích hợp. Kẻ tấn công có thể sử dụng các công cụ tự động hoặc mã đơn giản để thử nghiệm và khai thác lỗ hổng.

2.1.5 Hậu quả

Nghiêm Trọng: Hậu quả của việc khai thác lỗ hổng SQL Injection bao gồm mất mát dữ liệu, rò rỉ thông tin nhạy cảm, kiểm soát hệ thống, mất uy tín và các vấn đề pháp lý nghiêm trọng.

2.1.6 Đánh giá lỗ hổng theo OWASP SASVS

BẢNG 2.1 Đánh Giá SQL Injection

| Tiêu chí | Mô tả |
|----------------------|---|
| Phân loại theo MASVS | V6 - Platform Interaction Requirements |
| Mức độ nghiêm trọng | Critical (Rất cao) |
| CWEs | CWE-89: SQL Injection |
| Mô tả chi tiết | Lỗ hổng xảy ra khi dữ liệu đầu vào của người dùng không được kiểm tra hoặc làm sạch đúng cách trước khi được sử dụng trong câu lệnh SQL. Kẻ tấn công có thể chen các đoạn |

| | |
|------------------------------|--|
| | mã SQL độc hại vào đầu vào, cho phép chúng thực thi các lệnh SQL tùy ý trên cơ sở dữ liệu. |
| Khả năng khai thác | High (Cao). SQL Injection là một lỗ hổng phổ biến và có nhiều công cụ tự động giúp kẻ tấn công khai thác dễ dàng. |
| Tác động | Critical (Rất cao). Kẻ tấn công có thể đọc, sửa đổi, xóa dữ liệu nhạy cảm, thậm chí chiếm quyền kiểm soát hoàn toàn hệ thống. |
| Khuyến nghị khắc phục | Sử dụng Prepared Statements (câu lệnh chuẩn bị sẵn) hoặc ORM (Object-Relational Mapping) để tách biệt dữ liệu khỏi câu lệnh SQL. |

2.1.7 Biện pháp phòng ngừa chi tiết

Sử dụng Prepared Statements: Sử dụng các câu lệnh chuẩn bị sẵn để đảm bảo rằng dữ liệu được kiểm tra và làm sạch trước khi sử dụng trong câu lệnh SQL. Prepared Statements tách biệt dữ liệu và câu lệnh SQL, ngăn chặn việc chèn mã SQL độc hại.

```
// Ví dụ trong Java
String query = "SELECT * FROM users WHERE username = ? AND password = ?";
PreparedStatement preparedStatement = connection.prepareStatement(query);
preparedStatement.setString(1, username);
preparedStatement.setString(2, password);
ResultSet resultSet = preparedStatement.executeQuery();
```

Hình 2.1 Sử dụng Prepared Statements để ngăn chặn SQL Injection

Sử dụng ORM (Object-Relational Mapping): Sử dụng các framework ORM như Hibernate, Sequelize, hoặc Entity Framework để quản lý các câu lệnh SQL một cách an toàn. ORM tự động làm sạch và kiểm tra dữ liệu đầu vào.

Kiểm tra và làm sạch dữ liệu đầu vào: Áp dụng các biện pháp kiểm tra và làm sạch dữ liệu đầu vào để loại bỏ các ký tự đặc biệt và mã độc hại.

```
// Ví dụ trong JavaScript
function sanitizeInput(input) {
    return input.replace(/^[a-zA-Z0-9]/g, "");
}
let sanitizedUsername = sanitizeInput(username);
```

Hình 2.2 Kiểm tra dữ liệu đầu vào bằng cách sử dụng Regex

Sử dụng các thư viện bảo mật: Sử dụng các thư viện và công cụ bảo mật để phát hiện và ngăn chặn các lỗ hồng SQL Injection.

Đào tạo và nâng cao nhận thức: Đào tạo đội ngũ phát triển về các lỗ hồng bảo mật và biện pháp phòng ngừa, nâng cao nhận thức về tầm quan trọng của bảo mật ứng dụng.

2.1.8 Tổng kết

SQL Injection là một lỗ hồng bảo mật nghiêm trọng với mức độ tác động và khả năng khai thác rất cao. Việc phòng ngừa lỗ hồng này cần được đặt lên hàng đầu trong quá trình phát triển ứng dụng. Sử dụng các biện pháp như Prepared Statements, ORM, và kiểm tra dữ liệu đầu vào là rất quan trọng để đảm bảo an toàn cho cơ sở dữ liệu và hệ thống.

3.2 Eavesdropping (Man-In-The-Middle Attacks)

3.2.1 Giới Thiệu

Eavesdropping hay còn gọi là Man-In-The-Middle (MitM) attacks là một trong những lỗ hồng bảo mật phổ biến và nguy hiểm trong các ứng dụng web và di động. Lỗ hồng này cho phép kẻ tấn công chặn và đọc các dữ liệu truyền giữa hai bên mà không bị phát hiện. Trong bối cảnh bảo mật ứng dụng, việc hiểu và phòng ngừa các cuộc tấn công MitM là rất quan trọng để bảo vệ thông tin người dùng và hệ thống.

3.2.2 Mô Tả Lỗ Hồng Eavesdropping (Man-In-The-Middle Attacks)

Mã lỗ hồng: CWE-300, CWE-294

Mô tả: MitM là lỗ hồng bảo mật xảy ra khi kẻ tấn công chen mình vào giữa quá trình truyền thông giữa hai bên, cho phép chúng chặn, đọc, thay đổi hoặc chen các dữ liệu độc hại vào trong luồng thông tin.

Hình thái hoạt động: Kẻ tấn công sẽ tạo ra một điểm trung gian để chặn các gói dữ liệu giữa hai bên (ví dụ: giữa ứng dụng và máy chủ). Chúng có thể sử dụng các kỹ thuật như giả mạo DNS, ARP spoofing, hoặc thiết lập một điểm truy cập Wi-Fi giả mạo để thực hiện cuộc tấn công.

Môi trường: Các cuộc tấn công MitM có thể xảy ra trên các mạng công cộng hoặc không bảo mật, nơi mà kẻ tấn công có thể dễ dàng tiếp cận và chặn các luồng dữ liệu.

Đối tượng tác động: Bất kỳ ứng dụng nào truyền dữ liệu qua mạng không bảo mật hoặc không sử dụng các biện pháp mã hóa thích hợp đều có thể bị ảnh hưởng [2] [3] [4] [1].

3.2.3 Mức Độ Tác Động

Cao: Lỗ hồng MitM có thể gây ra các hậu quả nghiêm trọng như:

- **Rò rỉ dữ liệu nhạy cảm:** Thông tin cá nhân, thông tin tài chính, và các dữ liệu nhạy cảm khác có thể bị đánh cắp.
- **Thay đổi dữ liệu:** Kẻ tấn công có thể thay đổi nội dung dữ liệu truyền giữa hai bên mà không bị phát hiện.
- **Lừa đảo:** MitM có thể dẫn đến việc kẻ tấn công giả mạo một bên để lừa đảo người dùng, chiếm đoạt tài khoản hoặc thực hiện các giao dịch trái phép.

3.2.4 Khả Năng Khai Thác

Cao: MitM dễ bị khai thác nếu không có các biện pháp bảo vệ phù hợp, đặc biệt là trên các mạng công cộng hoặc không bảo mật.

- **Dễ thực hiện:** Kẻ tấn công có thể sử dụng các công cụ như Wireshark, ettercap, hoặc Cain & Abel để chặn và phân tích lưu lượng mạng.
- **Tự động hóa:** Các công cụ và kịch bản tự động có thể dễ dàng thiết lập và thực hiện các cuộc tấn công MitM.

3.2.5 Hậu Quả

Nghiêm Trọng: Hậu quả của việc bị khai thác lỗ hổng MitM có thể rất nghiêm trọng, bao gồm:

- **Mất dữ liệu:** Rò rỉ thông tin nhạy cảm như thông tin cá nhân, thông tin tài chính.
- **Thiệt hại tài chính:** Kẻ tấn công có thể thực hiện các giao dịch trái phép, chiếm đoạt tài khoản ngân hàng hoặc thông tin thẻ tín dụng.
- **Mất uy tín:** Sự cố bảo mật có thể dẫn đến mất uy tín nghiêm trọng cho công ty hoặc tổ chức, làm giảm lòng tin của khách hàng và đối tác.

3.2.6 Đánh Giá Theo OWASP MASVS

BẢNG 2.2 Đánh giá Eavesdropping (Man-In-The-Middle)

| Tiêu chí | Mô tả |
|-----------------------|--|
| Phân loại theo MASVS | V5 – Network Communication requirement |
| Mức độ nghiêm trọng | High(Cao) |
| CWEs | CWE-319: Cleartext Transmission of Sensitive Information, CWE-326: Inadequate Encryption Strength |
| Mô tả chi tiết | Lỗ hổng Eavesdropping (Man-In-The-Middle - MitM) xảy ra khi kẻ tấn công chặn và đọc dữ liệu nhạy cảm được truyền giữa ứng dụng di động và máy chủ. Điều này có thể xảy ra khi ứng dụng không sử dụng mã hóa mạnh hoặc sử dụng các giao thức không an toàn. |
| Khả năng khai thác | Medium đến High - Tùy thuộc vào môi trường mạng và các biện pháp bảo mật hiện có. Trên các mạng Wi-Fi công cộng không được bảo mật, việc khai thác có thể dễ dàng hơn. |
| Tác động | High - Kẻ tấn công có thể đánh cắp dữ liệu nhạy cảm như thông tin đăng nhập, thông tin tài chính, thông tin cá nhân,... Điều này có thể dẫn đến mất tài khoản, gian lận tài chính và các hậu quả nghiêm trọng khác. |
| Khuyến nghị khắc phục | Sử dụng HTTPS, Certificate Pinning, Mã hóa đầu cuối (End-to-end encryption) |

3.2.7 Biện Pháp Phòng Ngừa Chi Tiết

- **Sử dụng HTTPS:** Đảm bảo rằng tất cả các giao tiếp giữa ứng dụng và máy chủ được mã hóa bằng HTTPS với chứng chỉ hợp lệ.
- **Certificate Pinning (Ghim chứng chỉ):** Cảnh nhắc sử dụng kỹ thuật ghim chứng chỉ để ngăn chặn các cuộc tấn công MitM sử dụng chứng chỉ giả mạo.
- **Mã hóa đầu cuối (End-to-end encryption):** Nếu có thể, hãy mã hóa dữ liệu nhạy cảm ở phía client trước khi gửi đến máy chủ để đảm bảo rằng chỉ người nhận dự định mới có thể giải mã được.
- **Kiểm tra tính toàn vẹn của dữ liệu:** Sử dụng các cơ chế như mã xác thực thông báo (HMAC) để phát hiện xem dữ liệu có bị giả mạo hay không trong quá trình truyền.
- **Thường xuyên cập nhật các thư viện mã hóa:** Đảm bảo rằng ứng dụng sử dụng các thư viện mã hóa mới nhất và an toàn nhất.

3.2.8 Tổng Kết

Lỗ hổng Eavesdropping (MitM) là một lỗ hổng bảo mật nghiêm trọng với mức độ tác động và khả năng khai thác rất cao. Việc phòng ngừa lỗ hổng này cần được đặt lên hàng đầu trong quá trình phát triển ứng dụng. Sử dụng HTTPS, TLS, certificate pinning, và các biện pháp bảo mật mạng là rất quan trọng để đảm bảo an toàn cho dữ liệu truyền qua mạng và bảo vệ người dùng khỏi các cuộc tấn công MitM. Việc tuân thủ các yêu cầu bảo mật của OWASP MASVS sẽ giúp đảm bảo rằng ứng dụng được bảo vệ tốt nhất có thể.

3.3 Hard-Coded Keys/Tokens/Secrets/URL

3.3.1 Giới Thiệu

Lỗ hổng liên quan đến việc sử dụng các khóa, token hoặc bí mật (secrets) được mã hóa cứng (hard-coded) trong mã nguồn của ứng dụng di động là một vấn đề bảo mật nghiêm trọng. Việc này có thể dẫn đến việc kẻ tấn công dễ dàng truy cập vào các thông tin nhạy cảm và tài nguyên quan trọng của ứng dụng. Bài nghiên cứu này sẽ phân tích lỗ hổng hard-coded keys/tokens/secrets/URL theo tiêu chuẩn OWASP Mobile Application Security Verification Standard (MASVS), cụ thể là ASVS V2 - MSTG-STORAGE-14, và đưa ra các biện pháp phòng ngừa hiệu quả.

3.3.2 Mô Tả Lỗ Hổng Hard-Coded Keys/Tokens/Secrets

- **Mã lỗ hổng:** CWE-259, CWE-312, CWE-798
- **Mô tả:** Lỗ hổng này liên quan đến việc sử dụng các khóa, token hoặc bí mật được mã hóa cứng trực tiếp trong mã nguồn của ứng dụng. Điều này làm tăng nguy cơ kẻ tấn công truy cập được các thông tin nhạy cảm bằng cách phân tích mã nguồn hoặc bằng các công cụ reverse engineering.
- **Hình thái hoạt động:** Kẻ tấn công có thể phân tích mã nguồn của ứng dụng hoặc sử dụng các công cụ reverse engineering để tìm và khai thác các khóa, token hoặc bí mật được mã hóa cứng.
- **Môi trường:** Áp dụng cho các ứng dụng di động sử dụng các khóa, token hoặc bí mật để truy cập vào các dịch vụ hoặc tài nguyên nhạy cảm.

- **Đối tượng tác động:** Bất kỳ ứng dụng di động nào lưu trữ các khóa, token hoặc bí mật trong mã nguồn mà không có biện pháp bảo vệ thích hợp.
- **Cách thức hoạt động:** Kẻ tấn công có thể phân tích mã nguồn hoặc sử dụng các công cụ như APKTool, JD-GUI, để trích xuất và khai thác các thông tin nhạy cảm được mã hóa cứng.

3.3.3 Mức Độ Tác Động

Cao: Lỗ hổng hard-coded keys/tokens/secrets có thể dẫn đến các hậu quả nghiêm trọng như:

- **Rò rỉ thông tin nhạy cảm:** Kẻ tấn công có thể truy cập vào các thông tin nhạy cảm như khóa API, token xác thực, và các bí mật khác.
- **Truy cập trái phép:** Kẻ tấn công có thể sử dụng các khóa, token hoặc bí mật để truy cập trái phép vào các dịch vụ hoặc tài nguyên của ứng dụng.
- **Mất kiểm soát:** Kẻ tấn công có thể kiểm soát các tài nguyên quan trọng và gây ra thiệt hại nghiêm trọng cho ứng dụng và người dùng.

3.3.4 Khả Năng Khai Thác

Cao: Lỗ hổng này rất dễ khai thác vì các khóa, token hoặc bí mật được lưu trữ trực tiếp trong mã nguồn:

- **Dễ thực hiện:** Kẻ tấn công không cần phải có kiến thức chuyên sâu về bảo mật hoặc kỹ thuật cao cấp.
- **Công cụ hỗ trợ:** Các công cụ reverse engineering như APKTool, JD-GUI, hoặc jadx có thể dễ dàng trích xuất mã nguồn và tìm các thông tin nhạy cảm.

3.3.5 Hậu Quả

Nghiêm Trọng: Hậu quả của việc bị khai thác lỗ hổng hard-coded keys/tokens/secrets có thể rất nghiêm trọng, bao gồm:

- **Rò rỉ dữ liệu:** Kẻ tấn công có thể truy cập và rò rỉ thông tin nhạy cảm của người dùng và ứng dụng.
- **Truy cập trái phép:** Kẻ tấn công có thể sử dụng các khóa và token để truy cập trái phép vào các dịch vụ và tài nguyên.
- **Thiệt hại tài chính:** Việc truy cập trái phép có thể dẫn đến tổn thất tài chính lớn cho công ty hoặc tổ chức.
- **Mất uy tín:** Sự cố bảo mật có thể dẫn đến mất uy tín nghiêm trọng cho công ty hoặc tổ chức, làm giảm lòng tin của khách hàng và đối tác.

3.3.6 Đánh Giá theo OWASP MASVS

BẢNG 2.3 Đánh giá hard-coded keys/tokens/secrets/URL

| Tiêu chí | Mô tả |
|----------------------|--|
| Phân loại theo MASVS | V2 - Data Storage and Privacy Requirements |
| Mức độ nghiêm trọng | High(Cao) |
| CWEs | CWE-259: Hard-coded Password, |

| | |
|------------------------------|---|
| | <p>CWE-312: Cleartext Storage of Sensitive Information,</p> <p>CWE-798: Use of Hard-coded Credentials</p> |
| Mô tả chi tiết | <p>Lỗ hổng này xảy ra khi các khóa, mã thông báo (tokens) hoặc bí mật (secrets) nhạy cảm được nhúng trực tiếp vào mã nguồn của ứng dụng thay vì được lưu trữ và quản lý một cách an toàn. Điều này làm cho việc trích xuất các thông tin này trở nên dễ dàng đối với kẻ tấn công thông qua kỹ thuật đảo ngược mã nguồn (reverse engineering).</p> |
| Khả năng khai thác | <p>High (Cao) - Kẻ tấn công có thể dễ dàng khai thác lỗ hổng này nếu có quyền truy cập vào mã nguồn hoặc tệp tin APK/IPA của ứng dụng. Các công cụ phân tích mã tĩnh và động có thể nhanh chóng phát hiện các chuỗi hardcode trong ứng dụng.</p> |
| Tác động | <p>High (Cao) - Kẻ tấn công có thể truy cập trái phép vào dữ liệu nhạy cảm được bảo vệ bởi các khóa, mã thông báo hoặc bí mật này. Họ cũng có thể giả mạo danh tính ứng dụng và thực hiện các hành vi trái phép, hoặc tấn công vào các hệ thống/dịch vụ khác sử dụng các khóa hoặc mã thông báo này.</p> |
| Khuyến nghị khắc phục | <p>Không hardcode các khóa, mã thông báo hoặc bí mật trong mã nguồn.</p> <p>Sử dụng các giải pháp lưu trữ an toàn như keychain (iOS) hoặc keystore (Android).</p> <p>Áp dụng kỹ thuật che giấu mã (code obfuscation).</p> <p>Sử dụng các dịch vụ quản lý bí mật an toàn như Hashicorp Vault, AWS Secrets Manager hoặc Azure Key Vault.</p> |

MSTG-STORAGE-14 - Possible Hard-coded Keys/Tokens/Secrets: Ứng dụng cần phải tuân thủ các nguyên tắc sau để phòng ngừa lỗ hổng:

- **Không lưu trữ khóa, token hoặc bí mật trong mã nguồn:** Sử dụng các phương pháp lưu trữ an toàn hơn như keystore (trên Android) hoặc Keychain (trên iOS).

- **Sử dụng các dịch vụ bảo mật:** Sử dụng các dịch vụ bảo mật bên ngoài để quản lý và lưu trữ các khóa và token.
- **Mã hóa các thông tin nhạy cảm:** Nếu bắt buộc phải lưu trữ các khóa, token hoặc bí mật trong mã nguồn, hãy đảm bảo rằng chúng được mã hóa mạnh mẽ.
- **Quản lý vòng đời khóa và token:** Thường xuyên thay đổi và quản lý vòng đời của các khóa và token để giảm thiểu rủi ro.

3.3.7 Biện Pháp Phòng Ngừa Chi Tiết

- **Sử dụng Secure Storage:** Sử dụng các phương pháp lưu trữ an toàn như Android Keystore System hoặc iOS Keychain để lưu trữ các khóa và token.
- **Sử dụng các dịch vụ bảo mật:** Sử dụng các dịch vụ bảo mật bên ngoài như AWS Secrets Manager, Azure Key Vault hoặc Google Cloud Secret Manager để quản lý và lưu trữ các khóa và token.
- **Mã hóa các thông tin nhạy cảm:** Sử dụng các thư viện mã hóa mạnh mẽ để mã hóa các thông tin nhạy cảm trước khi lưu trữ.
- **Quản lý vòng đời khóa và token:** Thường xuyên thay đổi và quản lý vòng đời của các khóa và token để giảm thiểu rủi ro. Thiết lập các cơ chế tự động để thay đổi khóa và token định kỳ.

3.3.8 Tổng Kết

Lỗi hồng hard-coded keys/tokens/secrets là một lỗi hồng bảo mật nghiêm trọng với mức độ tác động và khả năng khai thác rất cao. Việc phòng ngừa lỗi hồng này cần được đặt lên hàng đầu trong quá trình phát triển ứng dụng. Sử dụng các biện pháp như secure storage, mã hóa, và quản lý vòng đời khóa và token là rất quan trọng để đảm bảo an toàn cho ứng dụng di động. Tuân thủ các yêu cầu bảo mật của OWASP MASVS sẽ giúp giảm thiểu rủi ro và bảo vệ ứng dụng khỏi các cuộc tấn công khai thác lỗi hồng này.

3.4 File Integrity Checks

3.4.1 Giới Thiệu

Lỗi hồng liên quan đến việc không thực hiện kiểm tra tính toàn vẹn của tệp trong các ứng dụng di động là một vấn đề bảo mật quan trọng. Điều này có thể dẫn đến việc ứng dụng bị tấn công bằng cách thay đổi hoặc thay thế các tệp nhạy cảm. Bài nghiên cứu này sẽ phân tích lỗi hồng thiếu kiểm tra tính toàn vẹn của tệp theo tiêu chuẩn OWASP Mobile Application Security Verification Standard (MASVS), cụ thể là MASVS V8 - MSTG-RESILIENCE-3 - File Integrity Checks, và đưa ra các biện pháp phòng ngừa hiệu quả.

3.4.2 Mô Tả Lỗi Hồng Kiểm Tra Tính Toàn Vẹn của Tệp

- **Mã lỗi hồng:** CWE-353, CWE-494, CWE-502
- **Mô tả:** Lỗi hồng này liên quan đến việc ứng dụng không thực hiện kiểm tra tính toàn vẹn của các tệp quan trọng hoặc mã nguồn. Điều này cho phép kẻ tấn công thay đổi hoặc thay thế các tệp này để thực hiện các hành vi độc hại.
- **Hình thái hoạt động:** Kẻ tấn công có thể thay đổi hoặc thay thế các tệp mà ứng dụng không kiểm tra tính toàn vẹn, từ đó thực hiện các hành vi như chen mã độc, thay đổi hành vi của ứng dụng hoặc truy cập vào các dữ liệu nhạy cảm.

- **Môi trường:** Áp dụng cho các ứng dụng di động có chứa các tệp nhạy cảm hoặc mã nguồn mà không có biện pháp kiểm tra tính toàn vẹn.
- **Đối tượng tác động:** Bất kỳ ứng dụng di động nào không thực hiện kiểm tra tính toàn vẹn của các tệp quan trọng.
- **Cách thức hoạt động:** Kẻ tấn công có thể sử dụng các công cụ như APKTool, JADX để phân tích và thay đổi các tệp của ứng dụng mà không bị phát hiện do thiếu kiểm tra tính toàn vẹn.

3.4.3 Mức Độ Tác Động

- **Cao:** Lỗ hổng thiếu kiểm tra tính toàn vẹn của tệp có thể dẫn đến các hậu quả nghiêm trọng như:
 - Chèn mã độc: Kẻ tấn công có thể thay đổi các tệp để chèn mã độc vào ứng dụng.
 - Thay đổi hành vi ứng dụng: Kẻ tấn công có thể thay đổi hành vi của ứng dụng để thực hiện các hành vi trái phép.
 - Truy cập dữ liệu nhạy cảm: Kẻ tấn công có thể thay đổi các tệp để truy cập vào các dữ liệu nhạy cảm mà ứng dụng lưu trữ hoặc xử lý.

3.4.4 Khả Năng Khai Thác

- **Cao:** Lỗ hổng này rất dễ khai thác vì các tệp không được kiểm tra tính toàn vẹn:
 - Dễ thực hiện: Kẻ tấn công không cần phải có kiến thức chuyên sâu về bảo mật hoặc kỹ thuật cao cấp.
 - Công cụ hỗ trợ: Các công cụ reverse engineering như APKTool, JADX, hoặc các trình biên dịch lại tệp APK có thể dễ dàng thay đổi các tệp của ứng dụng.

3.4.5 Hậu Quả

- **Nghiêm Trọng:** Hậu quả của việc bị khai thác lỗ hổng thiếu kiểm tra tính toàn vẹn của tệp có thể rất nghiêm trọng, bao gồm:
 - Chèn mã độc: Ứng dụng có thể bị chèn mã độc gây nguy hại cho người dùng.
 - Thay đổi hành vi: Ứng dụng có thể bị thay đổi hành vi để thực hiện các hoạt động trái phép hoặc không mong muốn.
 - Truy cập dữ liệu nhạy cảm: Kẻ tấn công có thể truy cập và rò rỉ thông tin nhạy cảm của người dùng và ứng dụng.
 - Thiệt hại tài chính: Việc truy cập trái phép có thể dẫn đến tổn thất tài chính lớn cho công ty hoặc tổ chức.
 - Mất uy tín: Sự cố bảo mật có thể dẫn đến mất uy tín nghiêm trọng cho công ty hoặc tổ chức, làm giảm lòng tin của khách hàng và đối tác.

3.4.6 Đánh Giá theo OWASP MASVS

BẢNG 2.4 File Integrity Checks

| Tiêu chí | Mô tả |
|----------------------|------------------------------|
| Phân loại theo MASVS | V8 - Resilience Requirements |
| Mức độ nghiêm trọng | High(Cao) |

| | |
|---------------------------|--|
| CWEs | <p>CWE-353: Missing Support for Integrity Check,</p> <p>CWE-494: Download of Code Without Integrity Check,</p> <p>CWE-502: Deserialization of Untrusted Data</p> |
| Mô tả chi tiết | <p>Lỗ hổng xảy ra khi ứng dụng di động không thực hiện kiểm tra tính toàn vẹn của các tệp quan trọng, bao gồm tệp APK/IPA, mã nguồn, tệp cấu hình, và các tệp lưu trữ dữ liệu nhạy cảm. Điều này cho phép kẻ tấn công thay đổi hoặc thay thế các tệp này để thực hiện các hành vi độc hại như chèn mã độc, giả mạo ứng dụng, hoặc truy cập trái phép vào dữ liệu.</p> |
| Khả năng khai thác | <p>High (Cao) - Kẻ tấn công có thể dễ dàng khai thác lỗ hổng này nếu có quyền truy cập vào tệp APK/IPA của ứng dụng hoặc các tệp khác trên thiết bị. Các công cụ phân tích mã tĩnh và động có thể nhanh chóng phát hiện và thay đổi các tệp này mà không bị phát hiện do thiếu kiểm tra tính toàn vẹn.</p> |
| Tác động | <p>High (Cao) - Kẻ tấn công có thể gây ra nhiều hậu quả nghiêm trọng như:</p> <p>Chèn mã độc vào ứng dụng để thực hiện các hành vi trái phép như đánh cắp dữ liệu, điều khiển thiết bị, hoặc lây lan malware.</p> <p>Giả mạo danh tính ứng dụng để truy cập vào các tài nguyên và dữ liệu nhạy cảm.</p> <p>Thay đổi hành vi của ứng dụng để gây ra các lỗi hoặc hoạt động bất thường.</p> <p>Truy cập trái phép vào dữ liệu nhạy cảm được lưu trữ trên thiết bị.</p> |

| | |
|------------------------------|--|
| Khuyến nghị khắc phục | <p>Áp dụng kỹ thuật che giấu mã (code obfuscation) để làm khó việc phân tích và thay đổi mã nguồn.</p> <p>Sử dụng các công cụ phân tích mã tĩnh và động để phát hiện và sửa lỗi tiềm ẩn liên quan đến kiểm tra tính toàn vẹn tệp.</p> <p>Cập nhật thường xuyên các thư viện và frameworks của ứng dụng để đảm bảo an ninh.</p> <p>Thực hiện các bài kiểm tra thâm nhập và đánh giá bảo mật định kỳ để xác định và khắc phục các lỗ hổng.</p> |
|------------------------------|--|

3.4.7 Biện Pháp Khắc Phục Chi Tiết

- **Sử dụng chữ ký số:** Sử dụng chữ ký số để đảm bảo tính toàn vẹn của các tệp quan trọng. Điều này giúp phát hiện bất kỳ thay đổi nào trong các tệp khi chúng bị giả mạo hoặc bị thay thế.
- **Sử dụng hàm băm:** Sử dụng các hàm băm mạnh mẽ để kiểm tra tính toàn vẹn của các tệp trước khi sử dụng. Các giá trị băm nên được lưu trữ an toàn và so sánh với các giá trị băm tính toán tại thời điểm chạy.
- **Phát hiện thay đổi:** Áp dụng các biện pháp phát hiện và ngăn chặn thay đổi trái phép các tệp của ứng dụng. Điều này có thể bao gồm việc kiểm tra chữ ký hoặc hàm băm của các tệp trước khi sử dụng.
- **Kiểm tra tính toàn vẹn định kỳ:** Thực hiện kiểm tra tính toàn vẹn của các tệp quan trọng định kỳ để đảm bảo không có sự thay đổi trái phép.
- **Áp dụng kỹ thuật che giấu mã (code obfuscation):** Sử dụng các kỹ thuật che giấu mã để làm khó việc phân tích và thay đổi mã nguồn. Điều này làm tăng độ phức tạp của việc reverse engineering và giảm khả năng khai thác lỗ hổng.
- **Sử dụng các công cụ phân tích mã tĩnh và động:** Sử dụng các công cụ phân tích mã tĩnh và động để phát hiện và sửa lỗi tiềm ẩn liên quan đến kiểm tra tính toàn vẹn tệp. Điều này giúp phát hiện các lỗ hổng bảo mật trước khi mã được phát hành.
- **Cập nhật thường xuyên các thư viện và frameworks:** Cập nhật thường xuyên các thư viện và frameworks của ứng dụng để đảm bảo an ninh. Các bản cập nhật thường chứa các bản vá bảo mật và cải tiến bảo mật.
- **Thực hiện các bài kiểm tra thâm nhập và đánh giá bảo mật định kỳ:** Thực hiện các bài kiểm tra thâm nhập và đánh giá bảo mật định kỳ để xác định và khắc phục các lỗ hổng. Điều này giúp phát hiện sớm các lỗ hổng và đảm bảo rằng các biện pháp bảo mật đang hoạt động hiệu quả.

3.4.8 Tổng Kết

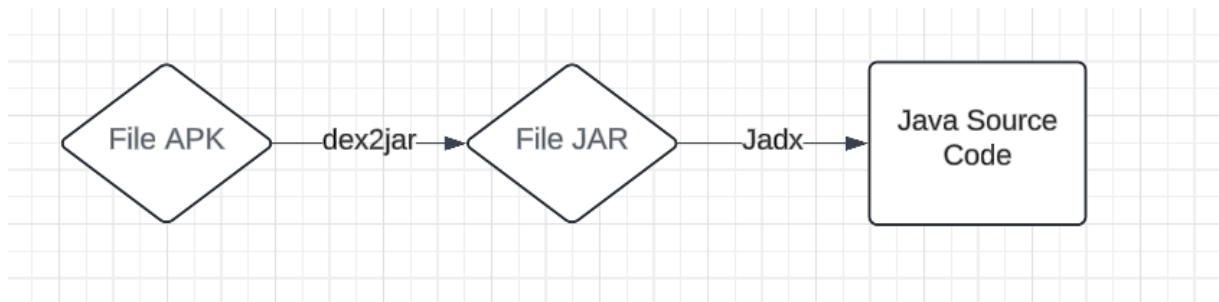
Lỗ hổng thiếu kiểm tra tính toàn vẹn của tệp là một lỗ hổng bảo mật nghiêm trọng với mức độ tác động và khả năng khai thác rất cao. Việc phòng ngừa lỗ hổng này cần được đặt lên hàng đầu trong quá trình phát triển ứng dụng. Sử dụng các biện pháp như chữ ký số, hàm băm, và phát hiện thay đổi là rất quan trọng để đảm bảo an toàn cho ứng dụng di động. Tuân thủ các yêu cầu bảo mật của OWASP MASVS sẽ giúp giảm thiểu rủi ro và bảo vệ ứng dụng khỏi các cuộc tấn công khai thác lỗ hổng này.

CHƯƠNG 3 : THỰC NGHIỆM PENTEST ỨNG DỤNG

3.1 giới thiệu về công cụ Pentest.

3.1.1 cơ chế hoạt động

Phân tích file APK nhằm khám phá cấu trúc, hành vi và xác định các lỗ hổng bảo mật của ứng dụng. Quá trình này bao gồm nhiều bước chi tiết, bắt đầu với việc kiểm tra file APK để xác nhận sự tồn tại và định dạng chính xác của file. Sau đó, thu thập thông tin về kích thước file, tên file và tính toán hash của file để đảm bảo tính toàn vẹn và không bị thay đổi. Một bước quan trọng trong quy trình này là chuyển đổi file APK. Sử dụng công cụ dex2jar, file APK được chuyển đổi thành file JAR, giúp chúng ta phân tích mã nguồn Java một cách dễ dàng hơn. Tiếp theo, công cụ Jadx được sử dụng để giải nén và chuyển đổi các file DEX trong APK thành mã nguồn Java, cho phép kiểm tra chi tiết cấu trúc và hành vi của ứng dụng.



Hình 3.1 Sơ đồ hoạt động của công cụ

3.1.2 Giới thiệu về tool

- **Go** là ngôn ngữ lập trình chính được sử dụng để viết chương trình phân tích. Go cung cấp các thư viện mạnh mẽ cho việc xử lý file, thực thi các lệnh hệ thống và xử lý chuỗi ký tự, giúp dễ dàng triển khai và mở rộng quá trình phân tích.
- **dex2jar** là công cụ được sử dụng để chuyển đổi file DEX trong APK thành file JAR. DEX (Dalvik Executable) là định dạng file chứa mã bytecode của Android, và việc chuyển đổi sang JAR cho phép chúng ta phân tích mã nguồn Java của ứng dụng
- **Jadx** là công cụ được sử dụng để giải nén và chuyển đổi file JAR thành mã nguồn Java. Jadx có khả năng decompile file DEX trong APK, giúp chúng ta có thể kiểm tra mã nguồn của ứng dụng một cách chi tiết

3.1.3 Cài đặt công cụ

Công cụ sẽ được cài đặt trên máy kali linux, chúng ta cần tải những tool cần thiết cho máy ảo như: golang, jadx, dex2jar, grep, etc [4].

Requirements:

- Install Git: `sudo apt-get install git`
- Install Golang: `sudo apt install golang-go`
- Install JADX: `sudo apt-get install jadx`
- Install Dex2jar: `sudo apt-get install dex2jar`

```
(root@kali)-[/home/kali/APKHunt]
# go run apkhunt.go -h

  _____
 /_ _ _ _ _ \
| H o m e |
|_ _ _ _ _ |
  \_ _ _ _ /

OWASP MASVS Static Analyzer

[+] APKHunt - a comprehensive static code analysis tool for Android apps
[+] Based on: OWASP MASVS - https://mobile-security.gitbook.io/masvs/
[+] Author: Sumit Kalaria & Mrunal Chawda
[*] Connect: Please do write to us for any suggestions/feedback.

APKHunt Usage:
  go run APKHunt.go [options] {apk file}

Options:
  -h      For help
  -p      Provide a single apk file-path
  -m      Provide the folder-path for multiple apk scanning
  -l      For logging (.txt file)

Examples:
  APKHunt.go -p /Downloads/android_app.apk
  APKHunt.go -p /Downloads/android_app.apk -l
  APKHunt.go -m /Downloads/android_apps/
  APKHunt.go -m /Downloads/android_apps/ -l

Note:
  - Tested on linux only!
  - Keep tools such as jadx, dex2jar, go, grep, etc.! installed
```

Hình 3.2 Khởi chạy công cụ với --help

3.2 Pentest và đưa ra kết quả

3.2.1 Kịch bản

Bước 1: Ban đầu sẽ khai thác các lỗ hổng trên app Android

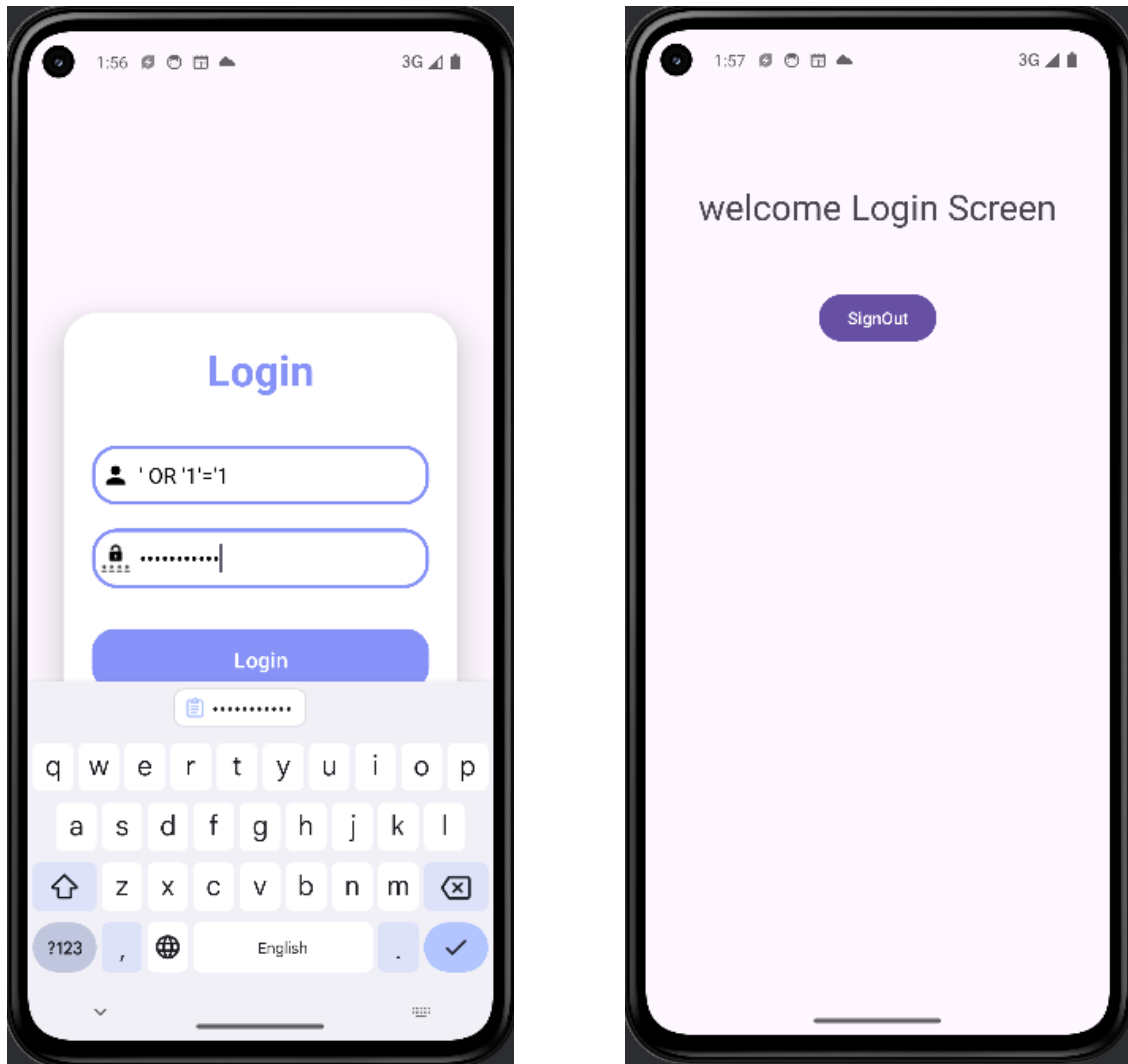
Bước 2: Build APK đưa vào công cụ Pentest thu thập kết quả

Bước 3: Khắc phục lỗ hổng, Pentest lần 2

3.2.2 Pentest SQL Injection

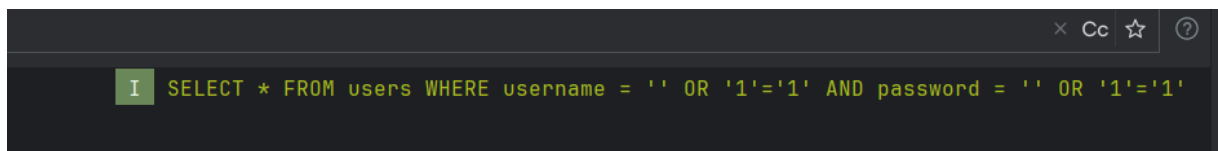
❖ Tấn công

- Đối tượng tấn công là chức năng đăng nhập của app Android, ta sẽ sử dụng cuối ' OR '1'='1 để nhập vào email và password.



Hình 3.3 Thực hiện login




Khi hệ thống bị tấn công bằng kỹ thuật SQL Injection với payload như OR '1'='1', mục đích của kẻ tấn công là thao túng truy vấn SQL để luôn trả về giá trị đúng, từ đó bỏ qua xác thực hoặc trích xuất dữ liệu từ cơ sở dữ liệu.

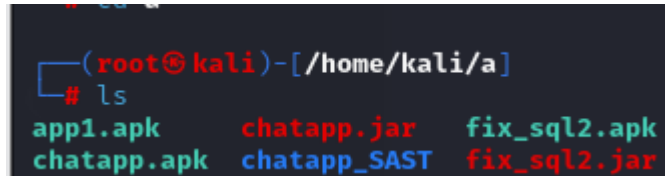


Hình 3.4 SQL nhận được.

➤ Khai thác lỗ hổng thành công

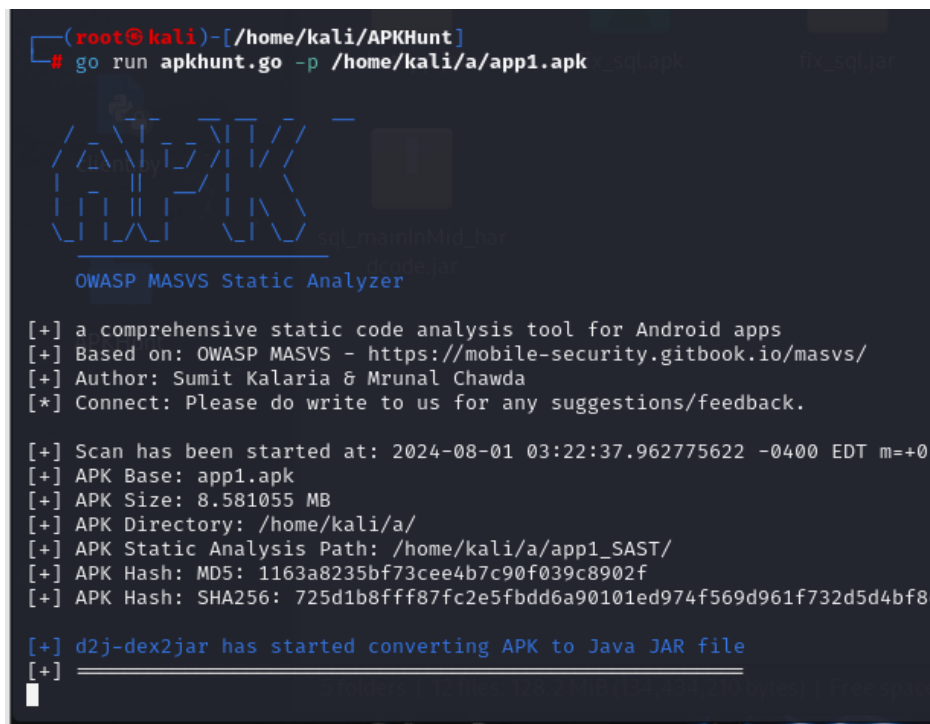
❖ Sử dụng công cụ để Pentest
Generate Signed Bundle (Build APK)

| Name | Date modified | Type | Size |
|--|------------------|------------------|----------|
|  baselineProfiles | 8/1/2024 2:10 PM | File folder | |
|  app1.apk | 8/1/2024 2:10 PM | Nox.apk | 8,788 KB |
|  output-metadata.json | 8/1/2024 2:10 PM | JSON Source File | 1 KB |



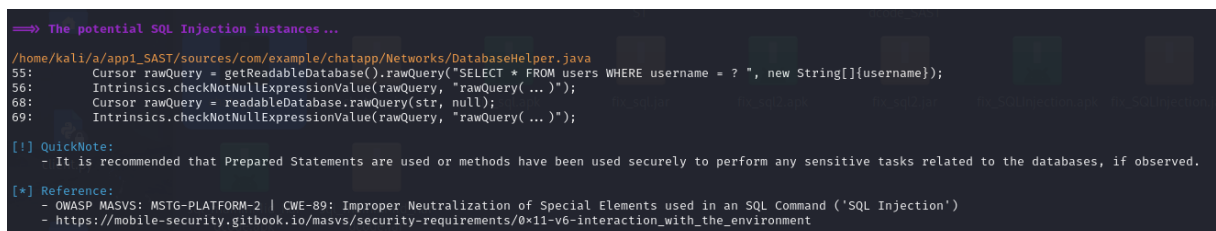
Hình 3.5 Chuẩn bị APK để Pentest

Bắt đầu Pentest



Hình 3.6 Đăng Pentest

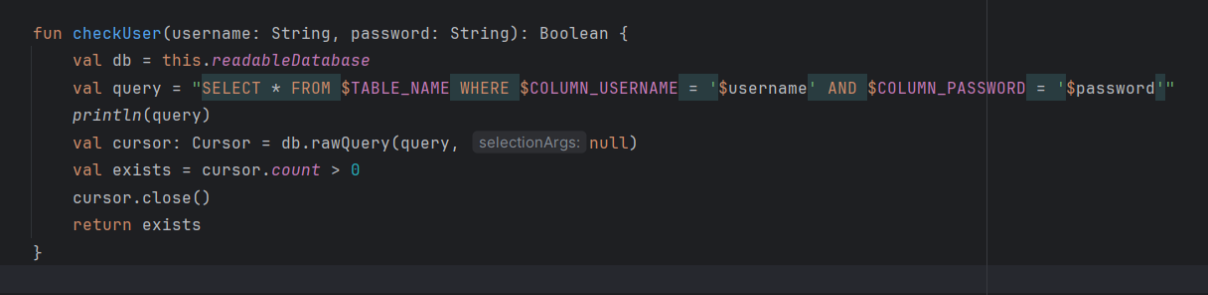
Kết quả Pentest



Hình 3.7 Cảnh báo SQL Injection

- Thông báo cho thấy rằng trình phân tích bảo mật đã phát hiện các trường hợp tiềm ẩn của SQL Injection trong mã nguồn của bạn, cụ thể là trong file DatabaseHelper.java. Các dòng mã được chỉ ra là:
- Dòng 55: `Cursor rawQuery = getReadableDatabase().rawQuery("SELECT * FROM users WHERE username = ?", new String[]{username});`
 - Dòng 68: `Cursor rawQuery = readableDatabase.rawQuery(str, null);`

Công cụ đã phát hiện rằng bạn đang sử dụng phương thức `rawQuery()` để thực hiện truy vấn SQL. Mặc dù dòng 55 có sử dụng tham số (?) để tránh SQL Injection, dòng 68 lại sử dụng biến chuỗi (`str`) mà không có sự bảo vệ rõ ràng, điều này có thể dẫn đến SQL Injection nếu biến chuỗi `str` được tạo từ đầu vào người dùng mà không được xác thực hoặc làm sạch đúng cách.



```
fun checkUser(username: String, password: String): Boolean {  
    val db = this.readableDatabase  
    val query = "SELECT * FROM $TABLE_NAME WHERE $COLUMN_USERNAME = '$username' AND $COLUMN_PASSWORD = '$password'"  
    println(query)  
    val cursor: Cursor = db.rawQuery(query, selectionArgs: null)  
    val exists = cursor.count > 0  
    cursor.close()  
    return exists  
}
```

Hình 3.8 Hàm gây ra lỗ hổng

❖ Khắc phục lỗ hổng và Pentest Lần 2

Phương pháp bạn đã sử dụng để kiểm tra người dùng với tên người dùng và mật khẩu là một cách tốt để tránh SQL Injection. Sử dụng các câu lệnh chuẩn bị sẵn hoặc phương thức truy vấn như `db.query()` giúp đảm bảo rằng các giá trị được truyền vào truy vấn SQL sẽ được xử lý an toàn, tránh nguy cơ SQL Injection.

```

fun checkUser(username: String, password: String): Boolean {
    val db = this.readableDatabase
    val selection = "$COLUMN_USERNAME = ? AND $COLUMN_PASSWORD = ?"
    val selectionArgs = arrayOf(username, password)

    val cursor: Cursor = db.query(
        TABLE_NAME,          // The table to query
        columns: null,          // The array of columns to return
        selection,              // The columns for the WHERE clause
        selectionArgs,          // The values for the WHERE clause
        groupBy: null,          // Group the rows
        having: null,           // Filter by row groups
        orderBy: null           // The sort order
    )

    val exists = cursor.count > 0
    cursor.close()
    db.close()
    return exists
}

```

Hình 3.9 Khắc phục lỗi hỏng

Build APK đưa vào Pentest lần 2

```

=> The potential SQL Injection instances ...
[+] Hunting begins based on "V8: Resilience Requirements"
[+]

```

Hình 3.10 Kết quả Pentest lần 2

- Pentest lần 2 công cụ đã trả ra kết quả ở mục SQL Injection là rỗng, Khắc phục đã thành công

❖ Tổng kết quy trình kiểm tra và Pentest, khắc phục

Bước 1: Khai thác lỗ hổng trên ứng dụng Android

- Lỗ hổng được phát hiện chủ yếu liên quan đến việc sử dụng các truy vấn SQL không an toàn, chẳng hạn như việc sử dụng trực tiếp dữ liệu người dùng mà không kiểm tra hoặc xử lý đúng cách.

Bước 2: Build APK và đưa vào công cụ Pentest để thu thập kết quả

- Sau khi phát hiện lỗ hổng, xây dựng lại tệp APK và chạy lại qua các công cụ Pentest để thu thập thông tin chi tiết về các lỗ hổng.
- Kết quả Pentest lần đầu cho thấy rằng có nhiều điểm yếu trong việc xử lý truy vấn SQL, bao gồm việc sử dụng trực tiếp các biến đầu vào của người dùng mà không qua kiểm tra hoặc xử lý.

Bước 3: Khắc phục lỗ hổng và Pentest lần 2

- Tiến hành khắc phục các lỗ hổng bằng cách thay đổi mã nguồn, áp dụng các biện pháp bảo mật như sử dụng câu lệnh chuẩn bị sẵn (prepared statements) và các phương thức truy vấn an toàn.
- Sau khi khắc phục, build lại tệp APK và tiến hành Pentest lần thứ hai để đảm bảo rằng các lỗ hổng đã được vá và không còn tồn tại.
- Kết quả Pentest lần thứ hai cho thấy rằng các lỗ hổng SQL Injection đã được khắc phục hoàn toàn. Không còn điểm yếu nào liên quan đến việc xử lý truy vấn SQL.

Kết luận

Quá trình kiểm tra và khắc phục lỗ hổng SQL Injection trên ứng dụng Android đã được thực hiện thành công theo các bước của MASVS (Mobile Application Security Verification Standard). Các bước chi tiết bao gồm việc khai thác lỗ hổng, thu thập kết quả, khắc phục và Pentest lại, đã đảm bảo rằng ứng dụng hiện tại đã an toàn trước các tấn công SQL Injection.

Kết quả của quá trình Pentest lần thứ hai xác nhận rằng các biện pháp bảo mật đã được áp dụng đúng cách và hiệu quả. Ứng dụng hiện tại đã tuân thủ các tiêu chuẩn bảo mật của MASVS, bảo vệ dữ liệu người dùng và đảm bảo tính toàn vẹn của cơ sở dữ liệu.

Các biện pháp khắc phục cụ thể bao gồm:

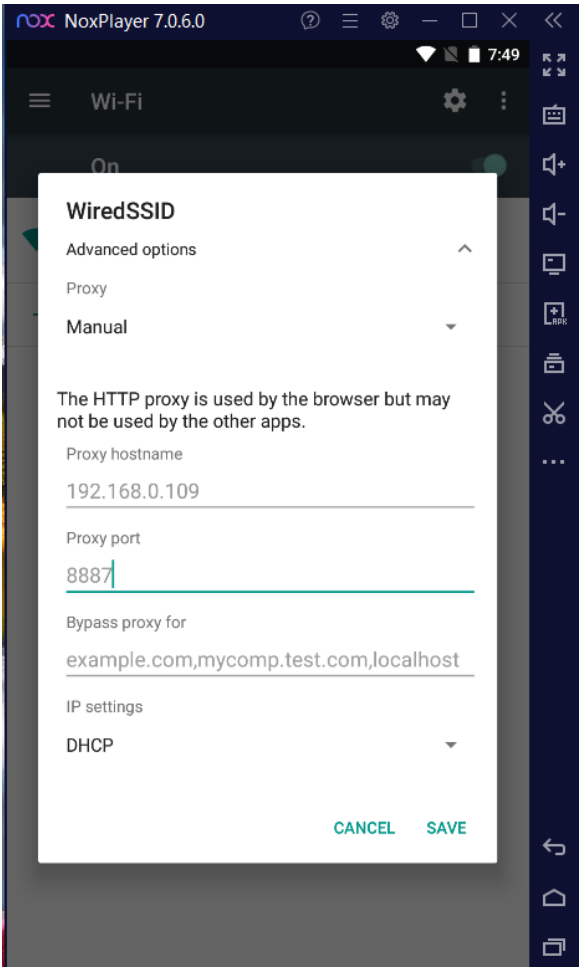
- Sử dụng câu lệnh chuẩn bị sẵn (prepared statements) cho tất cả các truy vấn SQL.
- Kiểm tra và xác thực dữ liệu đầu vào từ người dùng.
- Áp dụng các biện pháp mã hóa và bảo mật bổ sung khi cần thiết.

Việc áp dụng các biện pháp bảo mật này không chỉ giúp bảo vệ ứng dụng khỏi các tấn công SQL Injection mà còn cải thiện tổng thể tính bảo mật và tin cậy của ứng dụng Android.

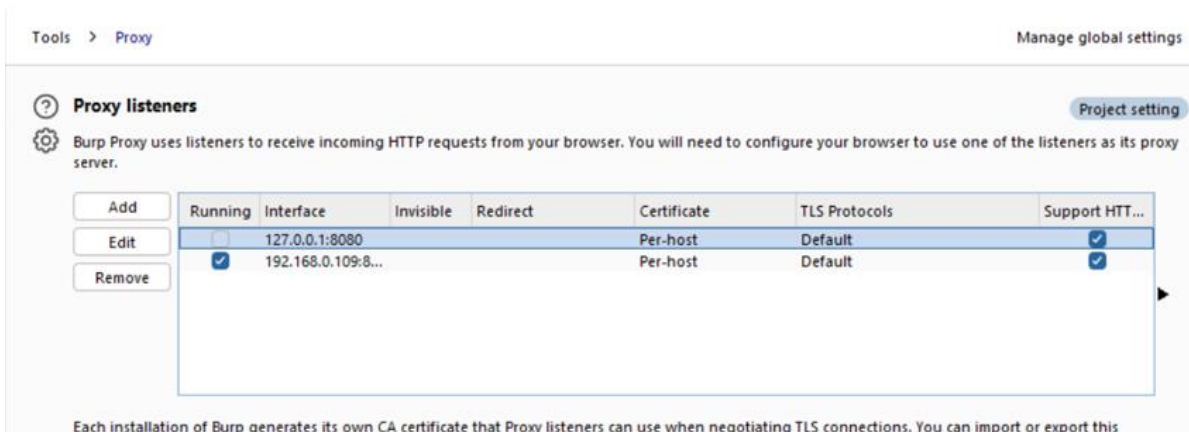
Kết luận này nhấn mạnh tầm quan trọng của việc thường xuyên kiểm tra và nâng cao các biện pháp bảo mật trong quá trình phát triển và triển khai ứng dụng, đảm bảo rằng ứng dụng luôn an toàn trước các nguy cơ bảo mật tiềm ẩn.

3.2.3 Pentest Eavesdropping (Man-In-The-Middle Attacks)**❖ Tấn công**

Chuẩn bị máy ảo có proxy setup, Burp Suite setup proxy chung với máy ảo Android

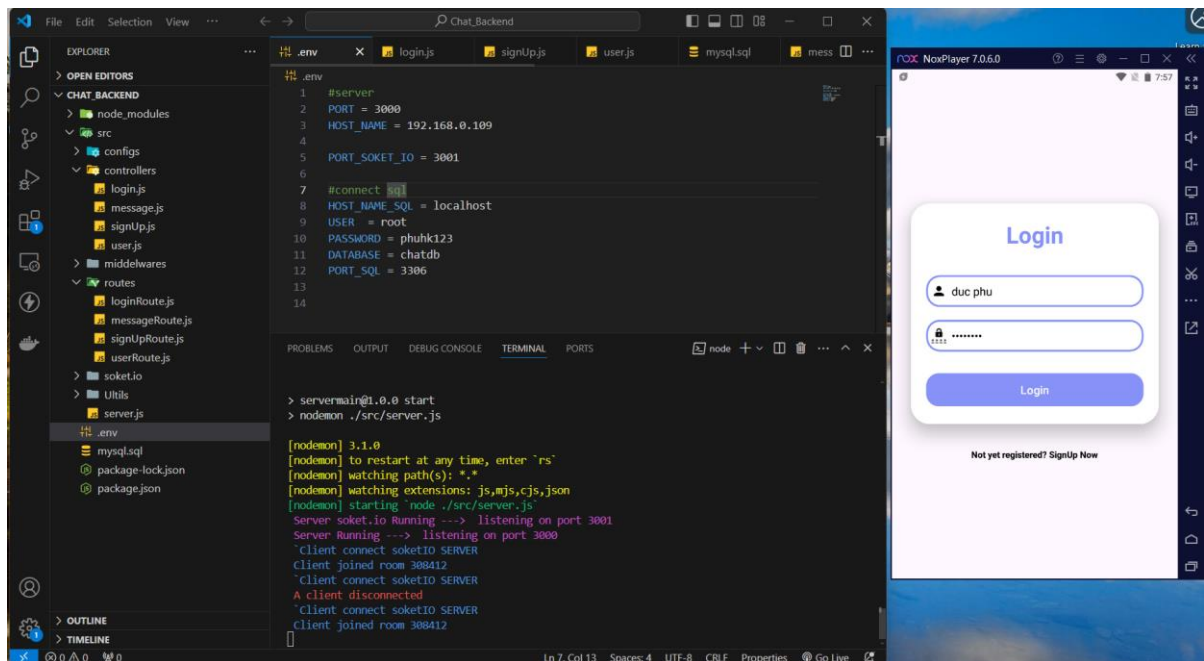


Hình 3.11 Set proxy cho thiết bị Android



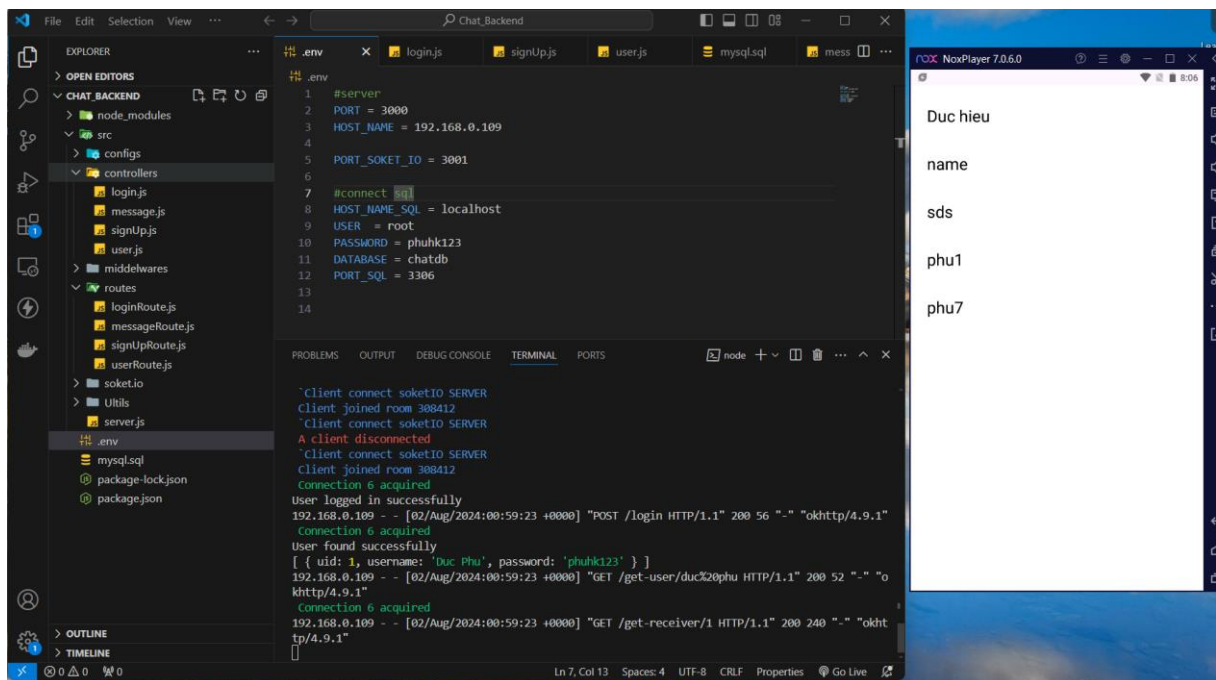
Hình 3.12 Set proxy cho Burp Suite

Máy chủ backend xử lý request từ bên app Android



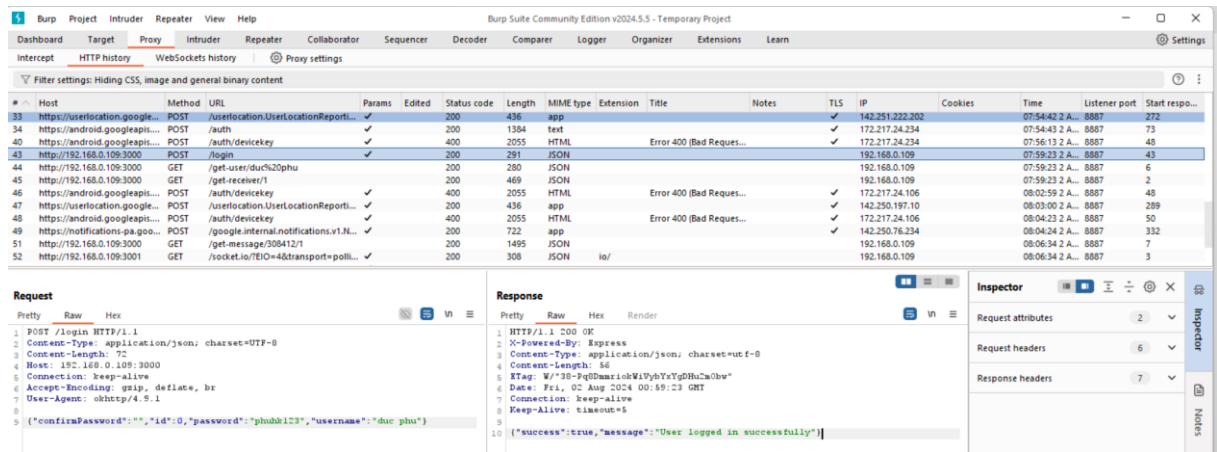
Hình 3.13 Chuẩn bị đăng nhập

Login thành công



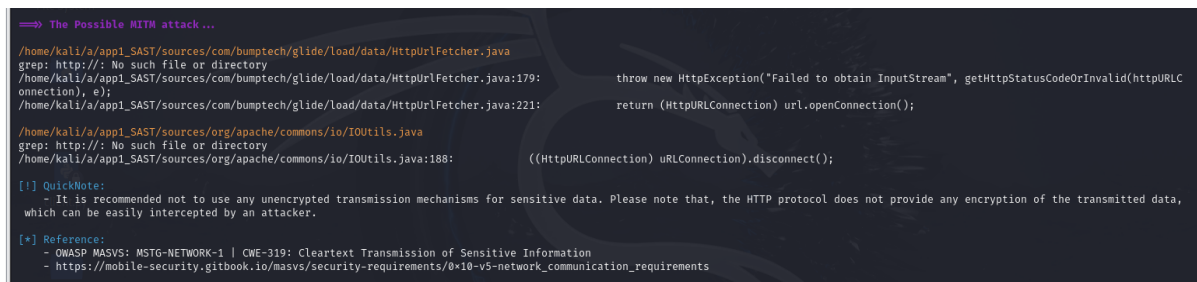
Hình 3.14 Đăng nhập

Kiểm tra Burp Suite ta thấy các các req/res đến từ máy



Hình 3.15 Burp Suite nghe lén thành công

- Tấn công thành công với Burp Suite, đã nghe lén được thông tin đăng nhập từ bên app Android.
 - ❖ Sử dụng công cụ để Pentest
- Generate APK đưa vào công cụ và bắt đầu Pentest ta có được kết quả



Hình 3.16 Kết quả Pentest MitM

- Công cụ Pentest đang cảnh báo rằng việc sử dụng giao thức HTTP (thay vì HTTPS) có thể dẫn đến việc truyền dữ liệu không được mã hóa, điều này có thể dễ dàng bị kẻ tấn công chặn và đánh cắp. Cụ thể, nó chỉ ra rằng có một số đoạn mã trong các tệp nguồn đang sử dụng HTTP, mà không đảm bảo an toàn cho dữ liệu nhạy cảm.
 - ❖ Khắc phục lỗ hổng và Pentest Lần 2
- Khắc phục: HTTPS giúp khắc phục các cuộc tấn công Man-In-The-Middle (MITM) bằng cách sử dụng chứng chỉ số để xác thực danh tính của máy chủ. Khi một kết nối HTTPS được thiết lập, trình duyệt của người dùng sẽ kiểm tra chứng chỉ số của máy chủ được cấp bởi một tổ chức chứng nhận đáng tin cậy. Nếu chứng chỉ hợp lệ, kết nối sẽ được mã hóa, đảm bảo rằng dữ liệu truyền tải giữa người dùng và máy chủ không thể bị đọc hoặc thay đổi bởi bên thứ ba. Quan trọng hơn, HTTPS ngăn chặn dữ liệu đi qua các proxy không hợp lệ hoặc bị giả mạo, vì những proxy này sẽ không thể cung cấp chứng chỉ hợp lệ, từ đó ngăn chặn các cuộc tấn công MITM hiệu quả.

Dưới đây là source code Server đã khắc phục sử dụng HTTPS có Certificates:

```

7  const fs = require('fs');
8  const https = require('https');
9
10 const port = process.env.PORT || 3000;
11 const hostname = process.env.HOST_NAME || 'localhost';
12
13 const options = {
14   key: fs.readFileSync(path.join(__dirname, "cetificates", "server.key")),
15   cert: fs.readFileSync(path.join(__dirname, "cetificates", "server.cert")),
16 };

```

Hình 3.17 Certificates

Bên app Android client ta cũng sẽ chuyển sang HTTPS có Certificates

```

private const val BASE_URL: String = "https://192.168.0.109:3000/"
fun getInstance(context: Context): ApiService {
    val client = getUnsafeOkHttpClient(context).build()

    return Retrofit.Builder()
        .baseUrl(BASE_URL)
        .client(client)
        .addConverterFactory(GsonConverterFactory.create())
        .build()
        .create(ApiService::class.java)
}

private fun getUnsafeOkHttpClient(context: Context): OkHttpClient.Builder {
    return try {
        // Load CAs from an InputStream
        val certificateFactory = CertificateFactory.getInstance("X.509")
        val inputStream = context.resources.openRawResource(R.raw.server) // Ensure this is the correct file
        val ca = certificateFactory.generateCertificate(inputStream)

        // Create a KeyStore containing our trusted CAs
        val keyStore = KeyStore.getInstance(KeyStore.getDefaultType())
        keyStore.load(stream: null, password: null)
        keyStore.setCertificateEntry("ca", ca)
    } catch (e: Exception) {
        // If the keyStore load fails, we'll use the default system certificates
        return OkHttpClient.Builder()
    }
}

```

Hình 3.18 Certificates in client

Sau khi khắc phục lỗi hỏng thì sẽ build và đưa app vào Pentest lần 2

```

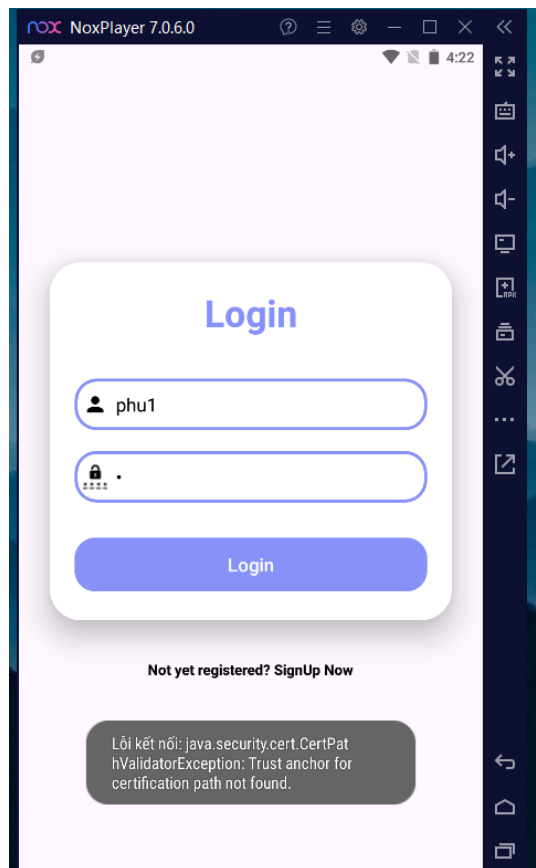
=> The Possible MITM attack ...
=> The Hard-coded Certificates/Key/Keystore files ...

```

Hình 3.19 Kết quả Pentest lần 2

Kết quả Pentest cho thấy ở mục của Man-In-The-Middle không có cảnh báo nào, khắc phục lỗi đã thành công

Tấn công lần 2 thì ta thấy dường như nó không đi qua được proxy không tin cậy



Hình 3.20 Test đăng nhập

❖ Tổng kết quy trình kiểm tra và Pentest, khắc phục

Bước 1: Khai thác lỗ hổng trên ứng dụng Android

- Sử dụng các công cụ Pentest để kiểm tra lỗ hổng MITM trên ứng dụng. Công cụ Burp Suite được sử dụng để mô phỏng tấn công MITM và phát hiện các điểm yếu trong giao thức truyền thông của ứng dụng.
- Các lỗ hổng được phát hiện chủ yếu liên quan đến việc sử dụng giao thức HTTP không an toàn để truyền dữ liệu nhạy cảm, dễ dàng bị chặn và can thiệp bởi kẻ tấn công.

Bước 2: Build APK và đưa vào công cụ Pentest để thu thập kết quả

- Sau khi phát hiện lỗ hổng, xây dựng lại tệp APK và chạy lại qua các công cụ Pentest để thu thập thông tin chi tiết về các lỗ hổng.
- Kết quả Pentest lần đầu cho thấy rằng có nhiều điểm yếu trong việc sử dụng giao thức HTTP không an toàn, bao gồm việc truyền dữ liệu nhạy cảm mà không mã hóa, dễ dàng bị chặn và thay đổi bởi kẻ tấn công MITM.

Bước 3: Khắc phục lỗ hổng và Pentest lần 2

- Tiến hành khắc phục các lỗ hổng bằng cách chuyển đổi từ giao thức HTTP sang HTTPS với chứng chỉ bảo mật. Điều này bao gồm:
 - Cài đặt chứng chỉ SSL/TLS cho máy chủ.
 - Cập nhật mã nguồn của ứng dụng để sử dụng HTTPS thay vì HTTP.
 - Đảm bảo tất cả các kết nối mạng đều được mã hóa và xác thực.

- Kết quả Pentest lần thứ hai cho thấy rằng các lỗ hổng MITM đã được khắc phục hoàn toàn.

Kết luận

Quá trình kiểm tra và khắc phục lỗ hổng Man-In-The-Middle (MITM) trên ứng dụng Android đã được thực hiện thành công theo các bước của MASVS (Mobile Application Security Verification Standard). Các bước chi tiết bao gồm việc khai thác lỗ hổng, thu thập kết quả, khắc phục và Pentest lại, đã đảm bảo rằng ứng dụng hiện tại đã an toàn trước các tấn công MITM.

Kết quả của quá trình Pentest lần thứ hai xác nhận rằng các biện pháp bảo mật đã được áp dụng đúng cách và hiệu quả. Ứng dụng hiện tại đã bảo vệ dữ liệu người dùng và đảm bảo tính toàn vẹn của giao thức truyền thông.

Các biện pháp khắc phục cụ thể bao gồm:

- Sử dụng giao thức HTTPS với chứng chỉ SSL/TLS để mã hóa toàn bộ dữ liệu truyền tải.
- Kiểm tra và xác thực chứng chỉ SSL/TLS để đảm bảo tính hợp lệ và an toàn của kết nối.
- Áp dụng các biện pháp bảo mật bổ sung khi cần thiết để bảo vệ dữ liệu người dùng.

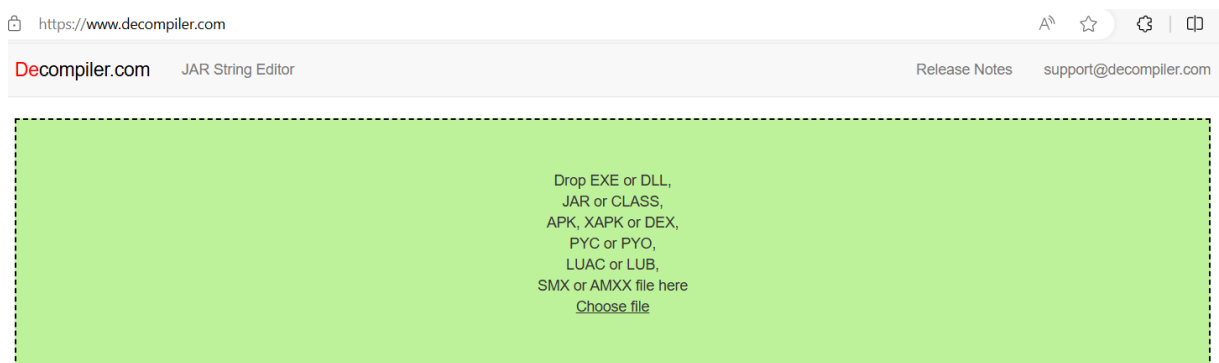
Việc áp dụng các biện pháp bảo mật này không chỉ giúp bảo vệ ứng dụng khỏi các tấn công MITM mà còn cải thiện tổng thể tính bảo mật và tin cậy của ứng dụng Android.

Kết luận này nhấn mạnh tầm quan trọng của việc thường xuyên kiểm tra và nâng cao các biện pháp bảo mật trong quá trình phát triển và triển khai ứng dụng, đảm bảo rằng ứng dụng luôn an toàn trước các nguy cơ bảo mật tiềm ẩn.

3.2.4 Pentest Hard-Coded Keys/Tokens/Secrets/URL

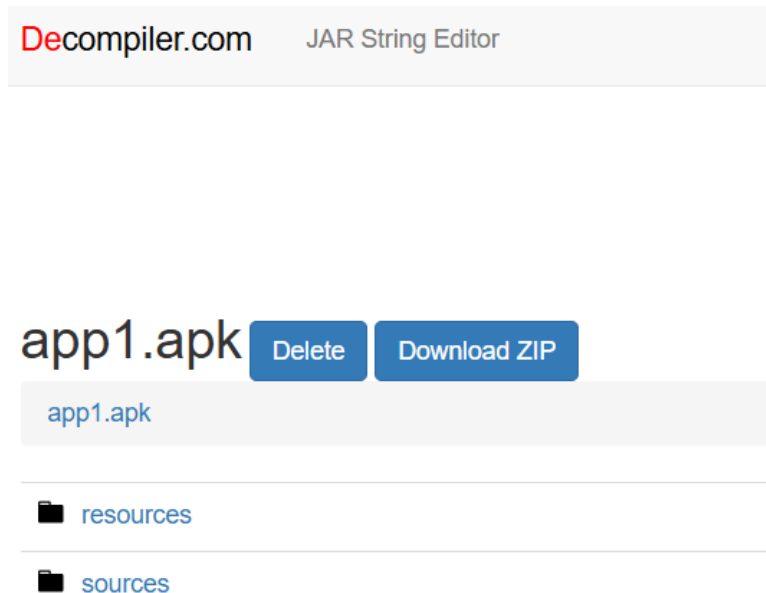
❖ Tấn công

Sử dụng tool Online Decompiler.com để dịch ngược File APK



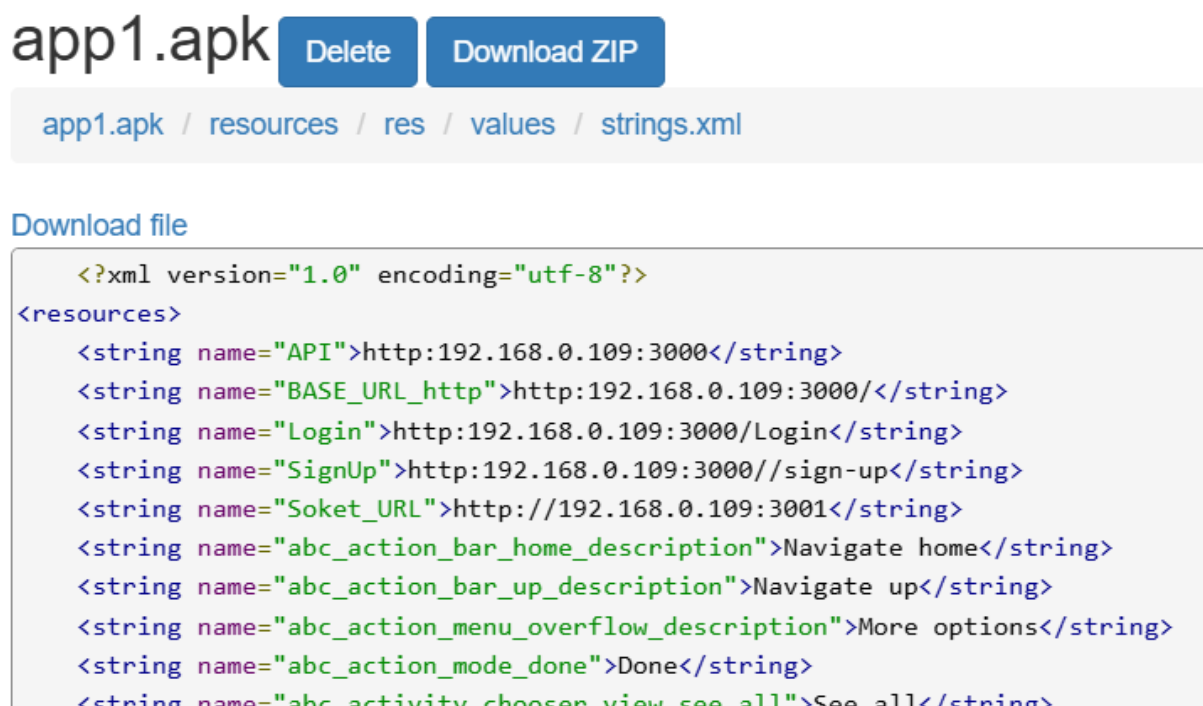
Hình 3.21 Decompiler.com

Tải file APK vào tool ta nhận được 2 bộ source



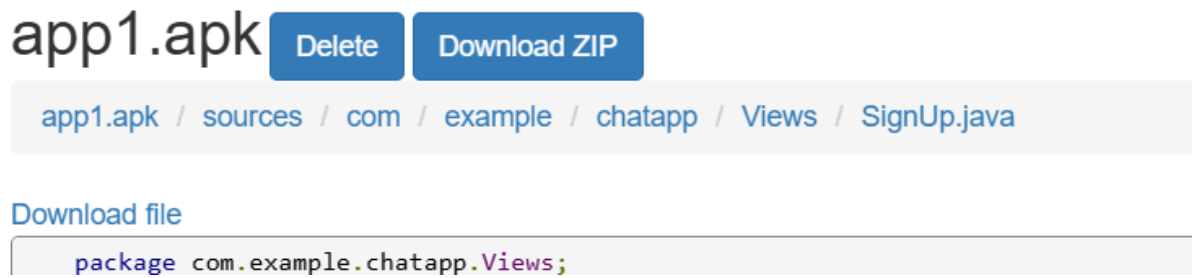
Hình 3.22 Dịch ngược thành công

Vào file string.xml theo đường dẫn resources/res/values/string.xml



Hình 3.23 File string.xml

Ta thấy dữ liệu trong file string.xml vẫn còn nguyên vẹn và để lộ URL với phương thức HTTP. Ta có thể sử dụng Postman để send API thử, ta để ý thấy có URL của SignUp, Login, trước tiên ta đến file SignUp.java.



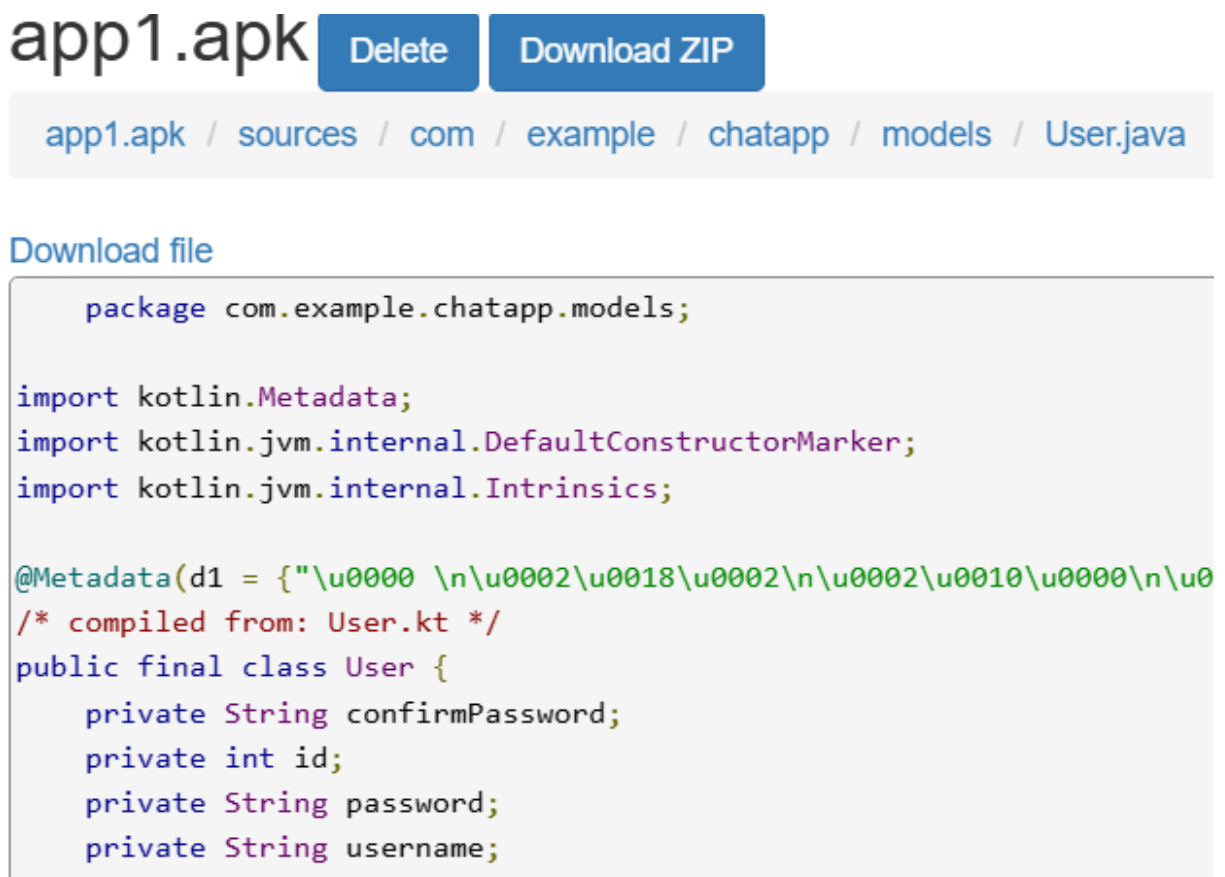
Hình 3.24 Path file SignUp.java

Tìm thấy hàm signUp

```
private final void signUp(User user) {
    RetrofitClient.INSTANCE.getInstance(this).SignUp(user).enqueue(new SignUp$signUp$1(this));
}
```

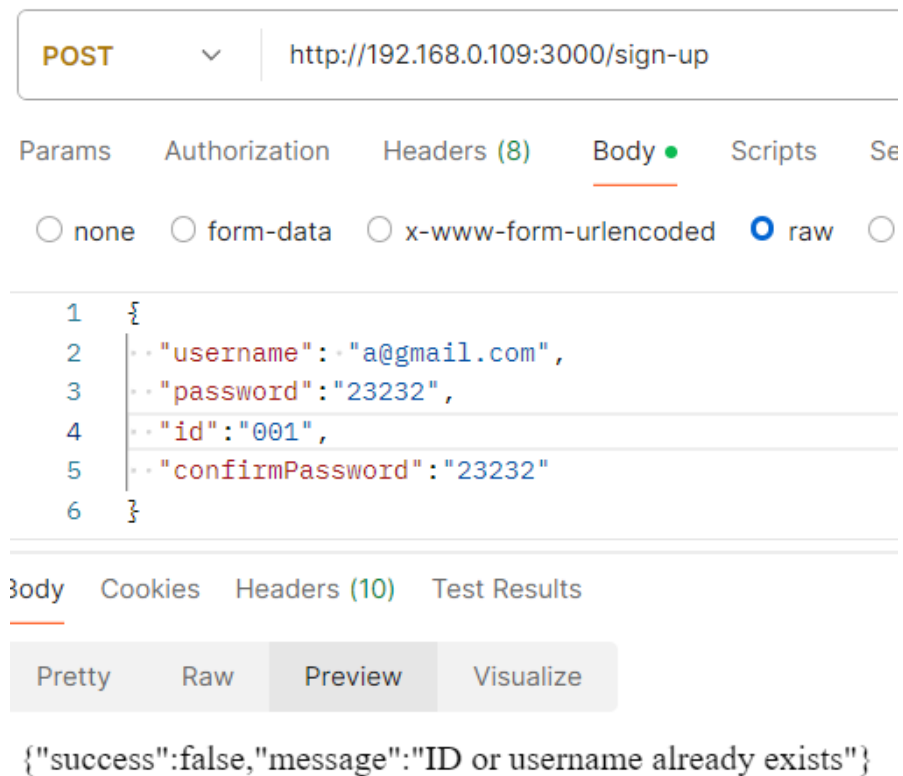
Hình 3.25 Hàm signUp

Ta thấy dữ liệu chuyển vào API là 1 Object User



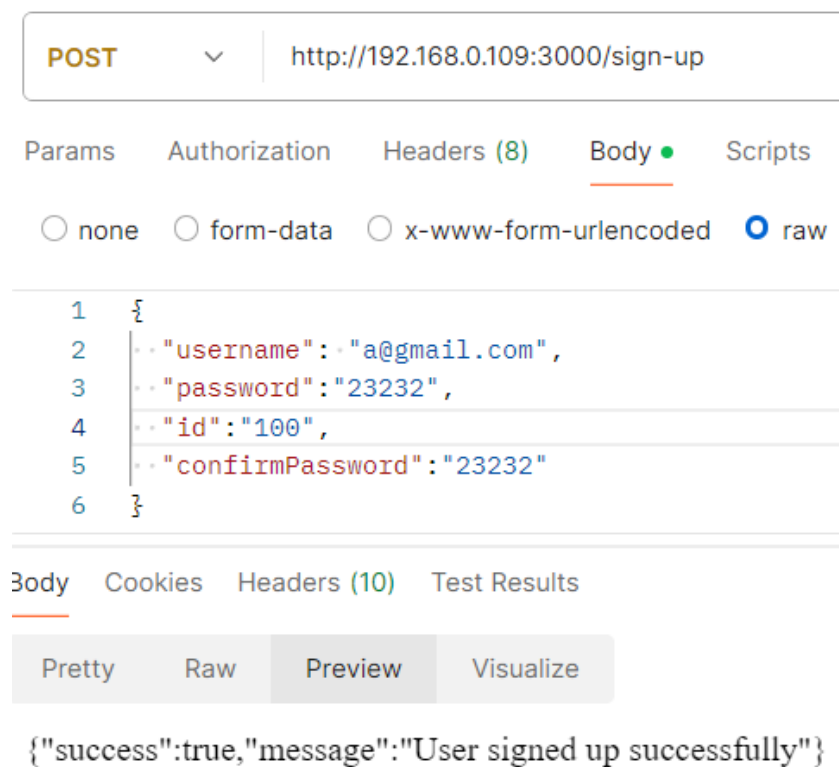
Hình 3.26 Model User

Object có 4 thuộc tính: confirmPassword, id, username, password, ta sẽ sử dụng Postman để test call API



Hình 3.27 Call api postman

Test thử lần đầu ta thấy đã trả về response, id hoặc username đã tồn tại, tiến hành thay đổi id và username



Hình 3.28 Call api thành công

- Call API đã thành công, ta có thể phát triển lỗ hổng này thành Dos hoặc spam đăng ký
- ❖ **Sử dụng công cụ để Pentest**

Đưa vào công cụ Pentest ta thấy công cụ đang list ra các biến hay các file có key word

```

=> The potential Hard-coded Keys/Tokens/Secrets ...

/home/kali/a/app2_SAST/resources/AndroidManifest.xml
2:<manifest xmlns:android="http://schemas.android.com/apk/res/android"

/home/kali/a/app2_SAST/resources/api-versions.xml
4535:    <field name="VISIBILITY_SECRET" since="21" />
4866:    <field name="ACTION_KEY" />
4870:    <field name="EXTRA_DATA_KEY" since="4" />
4881:    <field name="MENU_KEY" />
4909:    <field name="SUGGEST_COLUMN_TEXT_2_URL" since="8" />
5311:    <field name="EXTRA_PROVISIONING_WIFI_PAC_URL" since="21" />
6103:    <field name="EXTRA_PAIRING_KEY" since="19" />
7540:    <field name="EXTRA_REMOTE_INTENT_TOKEN" since="5" />
14174:    <field name="ERROR_NO_KEY" since="19" />
18335:    <field name="EGL_OPENGL_API" />
18338:    <field name="EGL_OPENGL_ES_API" />
18340:    <field name="EGL_OPENVG_API" />
22994:    <field name="CALENDAR_COLOR_KEY" since="15" />
23057:    <field name="COLOR_KEY" />
23106:    <field name="EVENT_COLOR_KEY" since="15" />
23263:    <field name="LIMIT_PARAM_KEY" since="17" />
23268:    <field name="OFFSET_PARAM_KEY" since="17" />
23436:    <field name="FILTER_TEXT_EXTRA_KEY" />
23437:    <field name="GROUP_NAME_EXTRA_KEY" />
23445:    <field name="TITLE_EXTRA_KEY" />
23543:    <field name="CONTENT_FILTER_URL" />
23555:    <field name="NUMBER_KEY" />
23627:    <field name="DIRECTORY_PARAM_KEY" since="11" />
23628:    <field name="LIMIT_PARAM_KEY" since="11" />
23846:    <field name="SEARCH_DISPLAY_NAME_KEY" since="16" />
23847:    <field name="SEARCH_PHONE_NUMBER_KEY" since="16" />
24104:    <field name="LOOKUP_KEY" />
24470:    <field name="DEFERRED_SNIPPETING_KEY" />
24570:    <field name="CONTACT_LOOKUP_KEY" />

```

Hình 3.29 Kết quả Pentest Hand-coded

Công cụ đã tìm ra được một số dữ liệu quan trọng

```

/home/kali/a/app2_SAST/resources/res/values/strings.xml
3:    <string name="API">http:192.168.0.109:3000</string>
4:    <string name="BASE_URL_http">http:192.168.0.109:3000</string>
5:    <string name="Soket_URL">http://192.168.0.109:3001</string>

/home/kali/a/app2_SAST/resources/res/xml/network_security_config.xml
2:<network-security-config xmlns:android="http://schemas.android.com/apk/res/android">

[!] QuickNote:
- It is recommended that the hard-coded keys/tokens/secrets should not be stored unless secur
malicious intentions.

[*] Reference:

```

Hình 3.30 Thông tin quan trọng bị lộ

- Pentest thành công, đã phát hiện ra 1 số URL quan trọng của ứng dụng
 - ❖ Khắc phục lỗ hổng và Pentest Lần 2
- Để khắc phục lỗ hổng này ta chỉ cần xóa các key word quan trọng bên trong file string.xml của mã nguồn, ta sẽ không lưu trữ cái gì quan trọng ở file đó cả

```
<resources>
  <string name="app_name">ChatAPP</string>
  <!-- Strings used for fragments for navigation -->
  <string name="first_fragment_label">First Fragment</string>
  <string name="second_fragment_label">Second Fragment</string>
  <string name="next">Next</string>
  <string name="previous">Previous</string>

  <string name="not_yet_registered_signup_now">Not yet registered? SignUp Now</string>
  <string name="login">Login</string>
  <string name="password">Password</string>
  <string name="username">Username</string>
  <string name="signup">SignUp</string>
  <string name="confirmpassword">confirmPassword</string>
  <string name="already_have_an_account_login">Already have an account? login</string>
  <string name="title_activity_login2">Login2</string>
  <string name="prompt_email">Email</string>
  <string name="prompt_password">Password</string>
  <string name="action_sign_in">Sign in or register</string>
  <string name="action_sign_in_short">Sign in</string>
  <string name="welcome">"Welcome !"</string>
  <string name="invalid_username">Not a valid username</string>
  <string name="invalid_password">Password must be >5 characters</string>
  <string name="login_failed">"Login failed"</string>
</resources>
```

Hình 3.31 Khắc phục loại bỏ key word trong string.xml

Sau khi xoá hết tất cả các key word quan trọng trong file string.xml ta sẽ đưa vào Pentest

```
==> The potential Hard-coded Keys/Tokens/Secrets ...

/home/kali/a/app2_SAST/resources/AndroidManifest.xml
2:<manifest xmlns:android="http://schemas.android.com/apk/res/android"

/home/kali/a/app2_SAST/resources/api-versions.xml
4535:    <field name="VISIBILITY_SECRET" since="21" />
4866:    <field name="ACTION_KEY" />
4870:    <field name="EXTRA_DATA_KEY" since="4" />
4881:    <field name="MENU_KEY" />
4909:    <field name="SUGGEST_COLUMN_TEXT_2_URL" since="8" />
5311:    <field name="EXTRA_PROVISIONING_WIFI_PAC_URL" since="21" />
6103:    <field name="EXTRA_PAIRING_KEY" since="19" />
7540:    <field name="EXTRA_REMOTE_INTENT_TOKEN" since="5" />
14174:    <field name="ERROR_NO_KEY" since="19" />
18335:    <field name="EGL_OPENGL_API" />
18338:    <field name="EGL_OPENGL_ES_API" />
18340:    <field name="EGL_OPENGL_API" />
22994:    <field name="CALENDAR_COLOR_KEY" since="15" />
23057:    <field name="COLOR_KEY" />
23106:    <field name="EVENT_COLOR_KEY" since="15" />
23263:    <field name="LIMIT_PARAM_KEY" since="17" />
23268:    <field name="OFFSET_PARAM_KEY" since="17" />
23436:    <field name="FILTER_TEXT_EXTRA_KEY" />
23437:    <field name="GROUP_NAME_EXTRA_KEY" />
23445:    <field name="TITLE_EXTRA_KEY" />
23543:    <field name="CONTENT_FILTER_URL" />
23555:    <field name="NUMBER_KEY" />
23627:    <field name="DIRECTORY_PARAM_KEY" since="11" />
```

Hình 3.32 Kết quả Pentest hand-coded lần 2

- ❖ Pentest lần 2 ta vẫn sẽ nhận được list các text có key work quan trọng nhưng sẽ không xuất hiện lại các key word và giá trị của key ở file string.xml nữa, đây là một lỗi trông rất đơn giản nhưng nếu bị khai thác thì nó rất là nghiêm trọng.

Bước 1: Khai thác lỗ hổng trên ứng dụng Android

- **Kỹ thuật kiểm tra:** Sử dụng các công cụ Pentest và phân tích mã nguồn để xác định sự tồn tại của các khóa, mã thông báo, bí mật, hoặc URL bị hard-coded trong mã nguồn ứng dụng.
- **Phát hiện:** Trong quá trình kiểm tra, các khóa nhạy cảm, mã thông báo và URL bị hard-coded đã được phát hiện trong tệp strings.xml, điều này có thể dẫn đến các nguy cơ bảo mật nghiêm trọng như truy cập trái phép và tấn công Man-In-The-Middle.

Bước 2: Build APK và đưa vào công cụ Pentest để thu thập kết quả

- **Build APK:** Sau khi xác định các lỗ hổng, ứng dụng được xây dựng lại và các công cụ Pentest được sử dụng để xác minh sự tồn tại của các khóa, mã thông báo và URL hard-coded.
- **Kết quả:** Các công cụ Pentest đã chỉ ra rằng ứng dụng chứa thông tin nhạy cảm bị hard-coded trong tệp strings.xml, điều này xác nhận rằng lỗ hổng chưa được khắc phục và có thể bị khai thác.

Bước 3: Khắc phục lỗ hổng và Pentest lần 2

- **Khắc phục:** Loại bỏ các từ khóa quan trọng, khóa, mã thông báo, và URL bị hard-coded khỏi tệp strings.xml. Thay vào đó, sử dụng các phương pháp bảo mật để quản lý thông tin nhạy cảm, chẳng hạn như lưu trữ bảo mật trên máy chủ và tải thông tin đó qua các kênh bảo mật khi cần thiết.
- **Pentest lần 2:** Xây dựng lại tệp APK và thực hiện Pentest lần thứ hai để xác minh rằng các lỗ hổng đã được khắc phục.

Kết luận

Quá trình kiểm tra và khắc phục các lỗ hổng liên quan đến Hard-Coded Keys/Tokens/Secrets/URL trên ứng dụng Android đã được thực hiện thành công theo các bước của MASVS (Mobile Application Security Verification Standard). Các bước chi tiết bao gồm khai thác lỗ hổng, thu thập kết quả, khắc phục và Pentest lại, đã đảm bảo rằng ứng dụng hiện tại đã an toàn khỏi các mối đe dọa liên quan đến thông tin nhạy cảm bị hard-coded.

Kết quả của Pentest lần thứ hai xác nhận rằng các lỗ hổng đã được khắc phục hoàn toàn. Cụ thể:

- **Loại bỏ thông tin nhạy cảm khỏi strings.xml:** Các từ khóa quan trọng, khóa, mã thông báo và URL bị hard-coded đã được loại bỏ khỏi tệp strings.xml. Thay vào đó, thông tin nhạy cảm hiện được quản lý và bảo vệ an toàn hơn bằng cách sử dụng các phương pháp bảo mật và lưu trữ an toàn.
- **Kiểm tra lại bảo mật:** Kết quả Pentest cho thấy không còn tồn tại các thông tin nhạy cảm bị hard-coded trong tệp strings.xml, ứng dụng hiện đáp ứng các tiêu chuẩn bảo mật của MASVS.

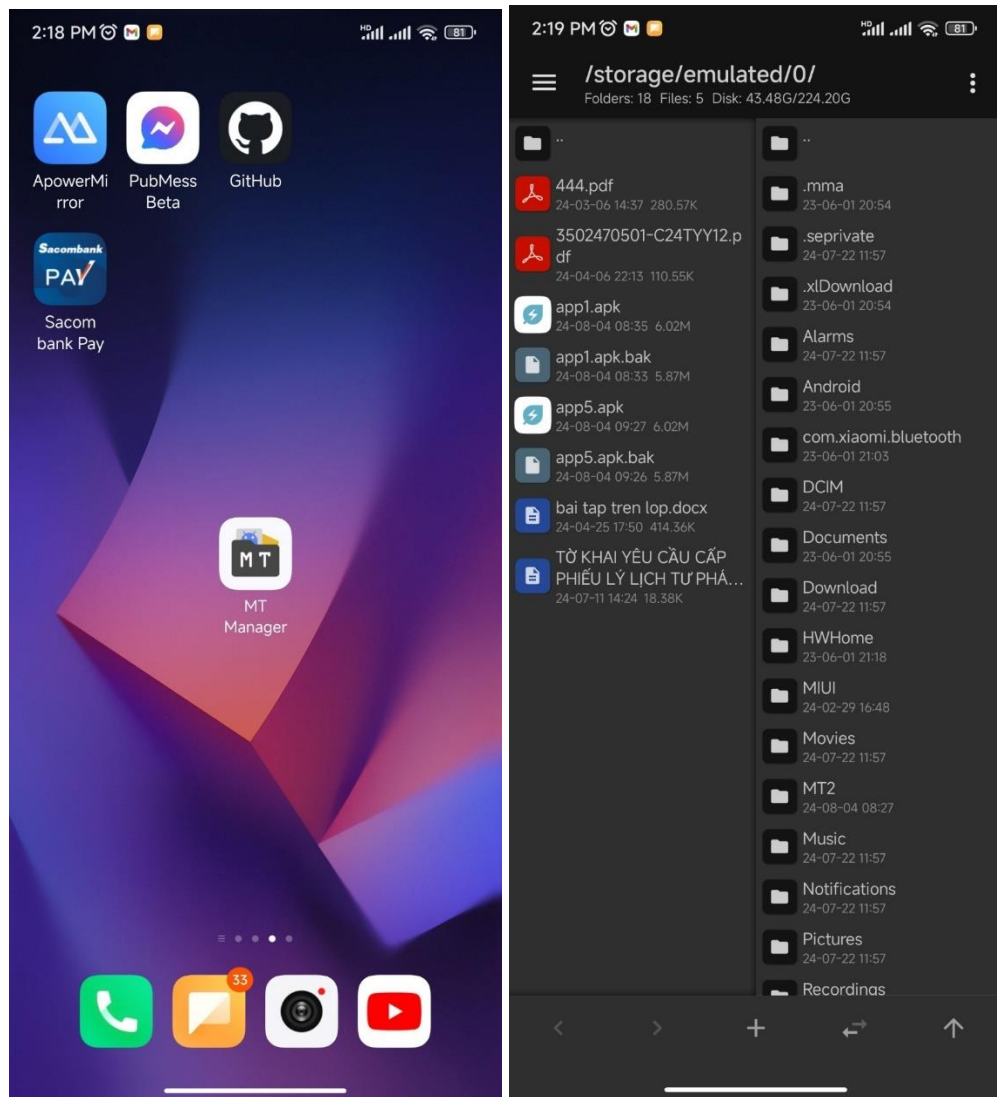
Kết luận này khẳng định rằng ứng dụng đã được bảo vệ tốt hơn trước các nguy cơ bảo mật liên quan đến việc hard-code thông tin nhạy cảm. Việc áp dụng các biện pháp bảo mật này không

chỉ giúp tăng cường bảo mật cho ứng dụng mà còn nâng cao độ tin cậy và bảo vệ dữ liệu người dùng. Việc kiểm tra bảo mật thường xuyên và cập nhật các biện pháp bảo mật là rất quan trọng để duy trì an toàn cho ứng dụng trong môi trường bảo mật luôn thay đổi.

3.2.5 Pentest file Integrity Checks

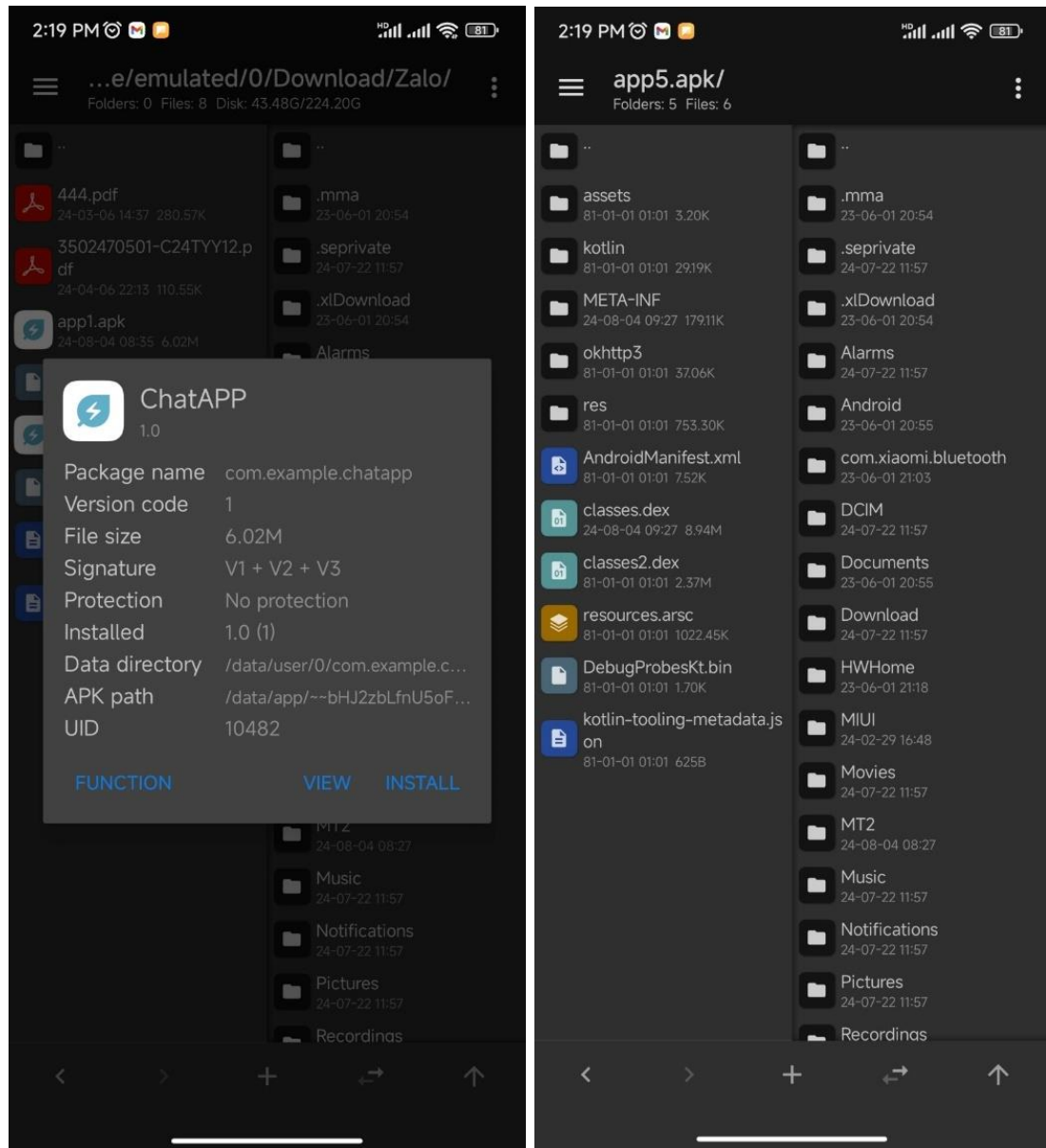
❖ Tấn công

Mô tả: ta sẽ sử dụng công cụ có tên MT Manager mở mã bytecode sửa đi url server bên trong 2 file .dex có tên là classes và classes2, sau khi thay đổi url của server gốc đổi thành URL của fake server ta sẽ ký lại ứng dụng và cài đặt chạy thử. Nếu chạy thành công ta có thể gửi ứng dụng đó cho người dùng.



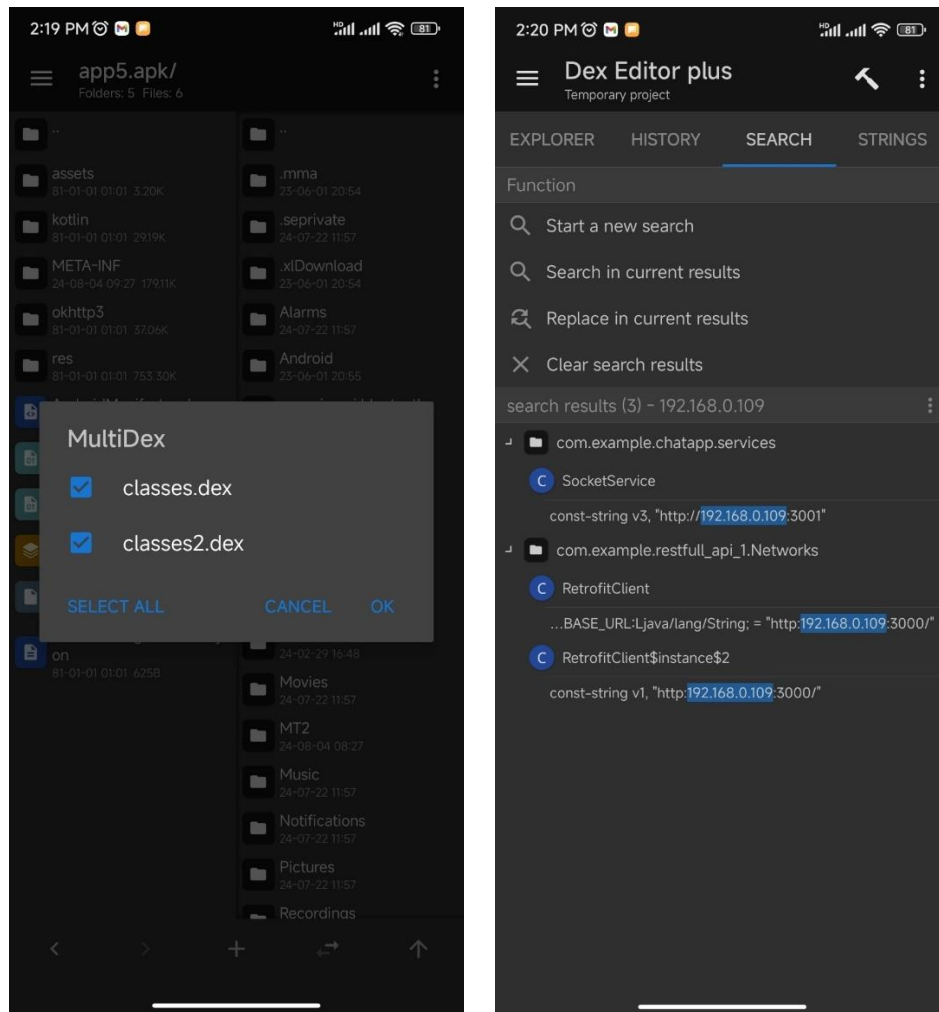
Hình 3.33 Khởi động MT Manager

Sau khi ta mở app PT Manager nên thì ta chọn app cần crack (app5.apk)



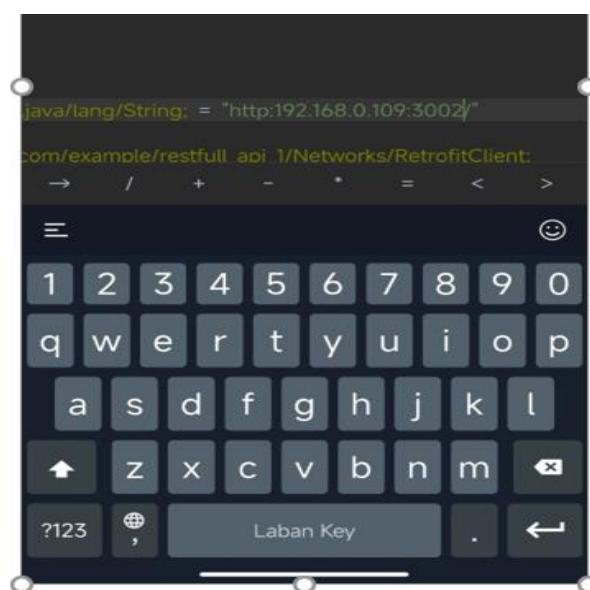
Hình 3.34 Mở app cần crack

Sau khi chọn view để mở app ta sẽ mở 2 file .dex là classes.dex và classes2.dex và đồng thời chuyển sang mục search để tìm kiếm từ khóa 192.168.0.109



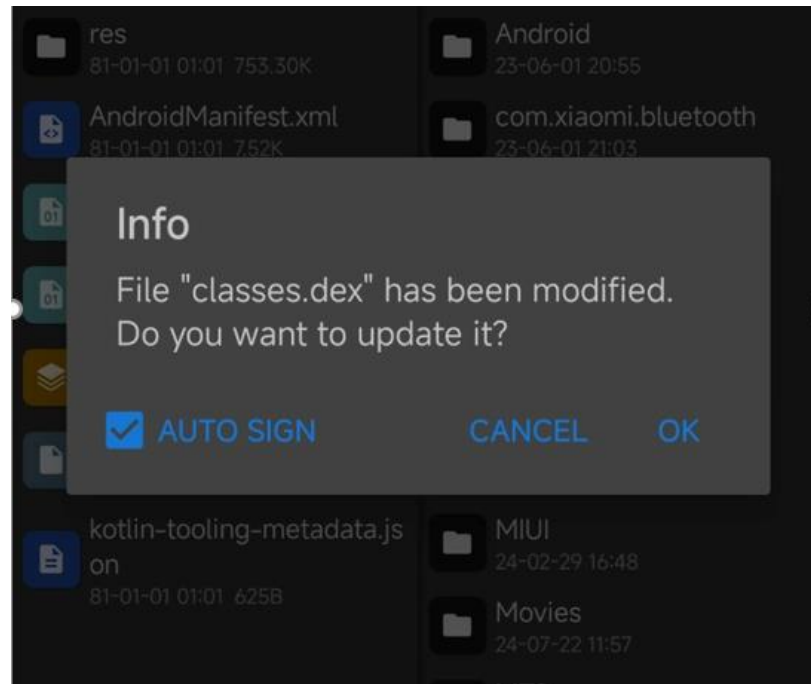
Hình 3.35 Mở 2 file *classes.dex* và *classes2.dex*

Sau khi search thì ta phát ra 2 vị trí có `http:192.168.0.109:3000`, ta sẽ sửa đổi port thành 3002



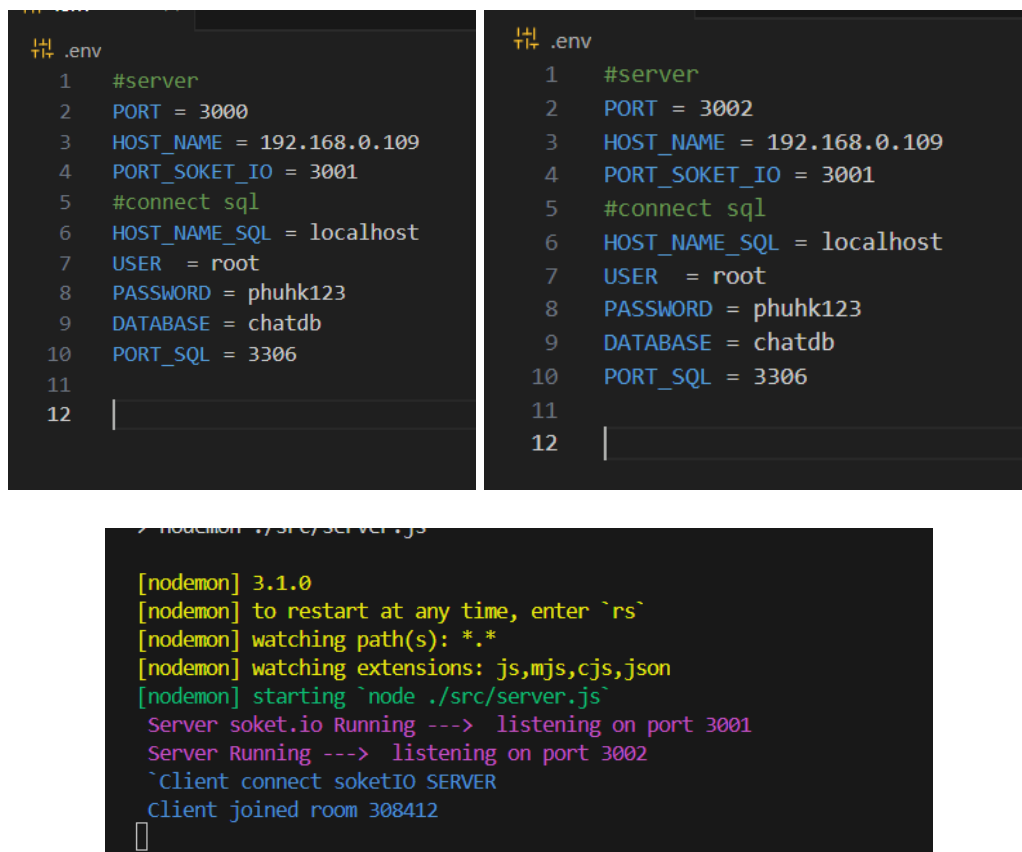
Hình 3.36 Sửa đổi URL

Sau khi sửa hết ta sẽ lưu lại và ký ứng dụng



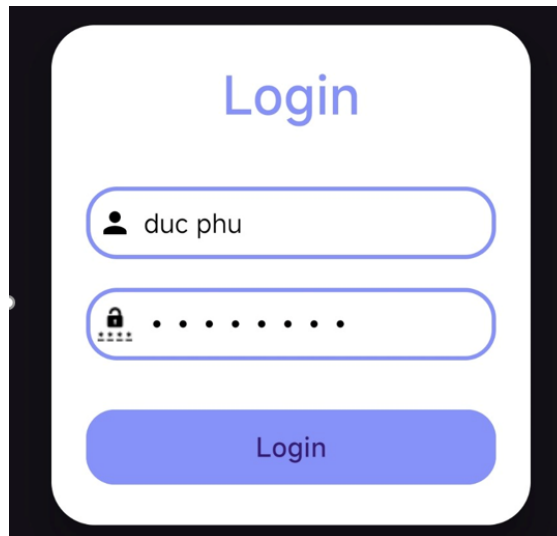
Hình 3.37 Ký lại ứng dụng

Sau khi ký thành công ta sẽ cài đặt app lên chạy thử đồng thời đổi file port server gốc sang port 3002



Hình 3.38 Chặn bị server với port 3002

Sau khi khởi động server lên ta sẽ mở app để đăng nhập thử



Hình 3.39 Test login

Kết quả đặt được

```
[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node ./src/server.js`
Server socket.io Running ---> listening on port 3001
Server Running ---> listening on port 3002
Client connect socketIO SERVER
Client joined room 308412
Connection 5 acquired
User logged in successfully
192.168.0.105 - - [04/Aug/2024:07:40:45 +0000] "POST /login HTTP/1.1" 200 56 "-" "okhttp/4.9.1"
Connection 5 acquired
User found successfully
[ { uid: 1, username: 'Duc Phu', password: 'phuhk123' } ]
192.168.0.105 - - [04/Aug/2024:07:40:46 +0000] "GET /get-user/duc%20phu HTTP/1.1" 200 52 "-" "okhttp/4.9.1"
Connection 5 acquired
192.168.0.105 - - [04/Aug/2024:07:40:46 +0000] "GET /get-receiver/1 HTTP/1.1" 200 296 "-" "okhttp/4.9.1"
```

Hình 3.40 Kết quả khi login

- Khai thác thành công, file apk đã bị thay đổi mã bytecode của URL Server ký lại, với url mới thì app vẫn sẽ hoạt động bình thường với fake Server

❖ Sử dụng công cụ để Pentest


Ta đưa vào công cụ để Pentest

```
(root@kali)-[/home/kali/APKHunt]
# go run apkhunt.go -check check -p /home/kali/a/appgoc.apk -p1 /home/kali/a/app5.apk
```

Hình 3.41 Lệnh test tính toàn vẹn của apk

Kết quả ta đạt được là file B đã bị thay đổi có nghĩa là file app5.apk đã bị thay đổi

```
(root@kali)-[/home/kali/APKHunt]
# go run apkhunt.go -check check -p /home/kali/a/appgoc.apk -p1 /home/kali/a/app5.apk
```



```
OWASP MASVS Static Analyzer

[+] a comprehensive static code analysis tool for Android apps
[+] Based on: OWASP MASVS - https://mobile-security.gitbook.io/masvs/
[+] Author: Sumit Kalaria & Mrunal Chawda
[*] Connect: Please do write to us for any suggestions/feedback.

[+] Hunting begins based on "V8: Resilience Requirements"
[+] _____
File B đã bị thay đổi.
```

Hình 3.42 Kết quả Pentest tính toàn vẹn của apk

Để test độ chính xác của công cụ ta thử đưa 2 file giống nhau vào

[illegible]

Hình 3.43 Pentest với 2 file giống nhau

Khi đưa 2 file giống nhau nhưng khác name vào thì công cụ vẫn check ra được là file B chưa bị thay đổi bytecode

- Kết quả Pentest của công cụ đã trả về là chính xác.

❖ Khắc phục lỗi hồng và Pentest Lần 2

Với lỗ hổng này thì bên phía người dùng nên chủ động phòng tránh. Người dùng có thể thực hiện nhiều biện pháp để tránh tình trạng bị dụ dỗ cài đặt và sử dụng các ứng dụng đã bị crack hoặc ứng dụng độc hại. Dưới đây là một số giải pháp giúp người dùng bảo vệ mình:

- Chỉ tải ứng dụng từ các nguồn tin cậy:
 - Google Play Store:** Tải ứng dụng từ cửa hàng chính thức của Google để giảm nguy cơ tải phải các ứng dụng độc hại.
 - Nguồn chính thức:** Truy cập trang web chính thức của nhà phát triển hoặc các cửa hàng ứng dụng được xác nhận khác.

- Kiểm tra đánh giá và nhận xét của người dùng:
 - **Đánh giá:** Đọc các đánh giá và nhận xét của người dùng khác để xem liệu có bất kỳ cảnh báo nào về ứng dụng.
 - **Số lượt tải:** Ứng dụng phổ biến với nhiều lượt tải thường an toàn hơn.
- Kiểm tra quyền truy cập yêu cầu
 - **Quyền truy cập:** Kiểm tra các quyền truy cập mà ứng dụng yêu cầu. Nếu ứng dụng yêu cầu quyền truy cập không cần thiết, đó có thể là dấu hiệu của một ứng dụng độc hại.

KẾT LUẬN

Trong nghiên cứu này, em đã tiến hành phân tích và phát hiện các lỗ hổng bảo mật theo tiêu chuẩn OWASP Mobile Application Security Verification Standard (MASVS) trên các ứng dụng Android. Bằng cách áp dụng các đặc điểm của lỗ hổng, em đã sử dụng công cụ để Pentest được những lỗ hổng quan trọng có thể bị khai thác bởi kẻ tấn công.

Qua quá trình nghiên cứu, em nhận thấy rằng nhiều ứng dụng Android hiện nay vẫn chưa đạt được các tiêu chuẩn bảo mật cần thiết, đặc biệt là trong các lĩnh vực như quản lý dữ liệu, xác thực người dùng, và truyền tải dữ liệu an toàn và mã code không an toàn do nhà phát triển tạo ra. Những lỗ hổng này không chỉ gây nguy hiểm cho dữ liệu cá nhân của người dùng mà còn làm giảm uy tín của các nhà phát triển ứng dụng.

Để khắc phục các vấn đề này, chúng tôi đề xuất một số biện pháp quan trọng mà các nhà phát triển ứng dụng cần thực hiện:

1. **Áp dụng các tiêu chuẩn bảo mật:** Tuân thủ nghiêm ngặt các tiêu chuẩn bảo mật như OWASP MASVS trong quá trình phát triển và kiểm thử ứng dụng.
2. **Kiểm tra và cập nhật thường xuyên:** Thực hiện kiểm tra bảo mật định kỳ và cập nhật các bản vá lỗi kịp thời để bảo vệ ứng dụng khỏi các lỗ hổng mới.
3. **Đào tạo và nâng cao nhận thức bảo mật:** Đào tạo các nhà phát triển về các phương pháp bảo mật hiệu quả và nâng cao nhận thức về tầm quan trọng của bảo mật ứng dụng.
4. **Sử dụng công cụ và dịch vụ bảo mật:** Sử dụng các công cụ và dịch vụ bảo mật chuyên dụng để phát hiện và khắc phục lỗ hổng một cách hiệu quả.

Hạn chế: trong các lỗ hổng của MASVS em mới Pentest và nghiên cứu chỉ một số lỗ hổng tiêu biểu, tuy nhiên còn rất nhiều lỗ hổng em chưa nghiên cứu và Pentest.

Định hướng tương lai: Chúng tôi dự định sẽ tiếp tục nghiên cứu và Pentest thêm nhiều lỗ hổng bảo mật khác để mở rộng phạm vi phân tích. Đồng thời, chúng tôi cũng sẽ hoàn thiện công cụ Pentest để nâng cao hiệu quả và độ chính xác trong việc phát hiện các lỗ hổng bảo mật.

Kết quả nghiên cứu này hy vọng sẽ góp phần vào việc nâng cao chất lượng bảo mật của các ứng dụng Android và khuyến khích các nhà phát triển chú trọng hơn đến việc bảo vệ dữ liệu người dùng. Bảo mật ứng dụng không chỉ là trách nhiệm của các nhà phát triển mà còn là yếu tố quan trọng để xây dựng niềm tin và sự hài lòng từ phía người dùng.

TÀI LIỆU THAM KHẢO

- [1] “Android (hệ điều hành),” 26 07 2024. [Trực tuyến]. Available: <https://vi.wikipedia.org/wiki/Android>. [Đã truy cập 30 07 2024].
- [2] “Cấu trúc của Android,” CÔNG TY TNHH ĐẦU TƯ VÀ DỊCH VỤ GIÁO GIỤC VIETJACK, 2015. [Trực tuyến]. Available: https://vietjack.com/Android/cau_truc_Android.jsp. [Đã truy cập 30 07 2024].
- [3] S. Kalaria, “OWASP MASVS Static Analyzer,” 2023. [Trực tuyến]. Available: <https://github.com/Cyber-Buddy/APKHunt>. [Đã truy cập 30 07 2024].
- [4] A. v. d. Stock, Daniel Cuthbert, Jim Manico, Josh C Grossman và Mark Burnett, “Aplication Security Verification Standard 4.0,” trong *OWASP*, 2019.