

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



# **BÁO CÁO THỰC TẬP TỐT NGHIỆP ĐẠI HỌC**

**Đề tài: “Pentest lỗ hổng trên ứng dụng Android”**

**Người hướng dẫn : HUỖNH TRỌNG THƯA**

**Sinh viên thực hiện : VÕ THANH PHONG**

**Mã số sinh viên : N20DCAT041**

**Lớp : D20CQAT01-N**

**Khoá : 2020 - 2025**

**Ngành : AN TOÀN THÔNG TIN**

**Hệ : CHÍNH QUY**

**TP.HCM, tháng 07/2024**

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



**BÁO CÁO THỰC TẬP  
TỐT NGHIỆP ĐẠI HỌC**

***Đề tài: “Pentest lỗ hổng trên ứng dụng Android”***

**Người hướng dẫn : HUỖNH TRỌNG THỨA**  
**Sinh viên thực hiện : VÕ THANH PHONG**  
**Mã số sinh viên : N20DCAT041**  
**Lớp : D20CQAT01-N**  
**Khoá : 2020 - 2025**  
**Ngành : AN TOÀN THÔNG TIN**  
**Hệ : CHÍNH QUY**

**TP.HCM, tháng 07/2024**

### NHẬN XÉT QUÁ TRÌNH THỰC TẬP CỦA SINH VIÊN

Họ và tên sinh viên : Võ Thanh Phong ..... Lớp: D20CQAT01-N.....  
 Nơi thực tập : Công ty Cổ phần Dịch vụ Công nghệ Tin Học HPT.....  
 Thời gian thực tập : 20/05/2024 - Hiện tại.....  
 Họ tên người nhận xét : Phạm Văn Hào.....

Nội dung đánh giá	Xếp loại (đánh dấu vào ô được chọn)				
	Tốt	Khá	Trung bình	Trung bình yếu	Yếu
<b>I. Tính kỷ luật và tư chất</b>					
I.1. Thực hiện nội quy		✓			
I.2. Thái độ làm việc		✓			
I.3. Năng lực tiếp thu		✓			
I.4. Khả năng vượt khó		✓			
I.5. Giao tiếp và ứng xử		✓			
<b>II. Khả năng chuyên môn</b>					
II.1. Kiến thức		✓			
II.2. Kỹ năng thực hành		✓			
II.3. Năng lực ngoại ngữ		✓			
II.4. Kỹ năng làm việc nhóm		✓			
II.5. Tính sáng tạo		✓			
<b>III. Kết quả thực hiện đề tài được giao</b>					
I.1. Thực hiện yêu cầu về nội dung		✓			
I.2. Thực hiện yêu cầu về tiến độ	✓				
<b>IV. Nhận xét khác và lời khuyên cho sinh viên:</b>					
<b>V. Điểm tổng kết thực tập tốt nghiệp (thang điểm 10):</b> <u>7</u> .....					

XÁC NHẬN CỦA TỔ CHỨC/DOANH NGHIỆP



*Lê Quốc Bảo*

NGƯỜI NHẬN XÉT  
(ký tên ghi rõ họ tên)

*Phạm Văn Hào*

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  
**Độc lập- Tự do- Hạnh phúc**

*TP. Hồ Chí Minh, ngày      tháng      năm 20...*

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**  
**THỰC TẬP TỐT NGHIỆP ĐẠI HỌC**

1. Tên đề tài: **Pentest lỗ hổng trên ứng dụng Android**
2. Sinh viên: **Võ Thanh Phong** **Lớp: D20CQAT01-N**
3. Giáo viên hướng dẫn: **Huỳnh Trọng Thừa**
4. Nơi công tác: **khoa Công nghệ thông tin 2**

**NỘI DUNG NHẬN XÉT**

1. Đánh giá chung:
- .....
- .....
- .....
- .....
2. Đánh giá chi tiết:
- .....
- .....
- .....
- .....
- .....
- .....
3. Nhận xét về tinh thần, thái độ làm việc:
- .....
- .....
4. Kết luận:
- .....
- .....
5. Điểm hướng dẫn:

**GIẢNG VIÊN HƯỚNG DẪN**  
(Ký, ghi rõ họ tên)

## LỜI CẢM ƠN

Trước hết, em xin gửi lời cảm ơn sâu sắc đến Thầy Huỳnh Trọng Thừa, người đã tận tình hướng dẫn, đóng góp ý kiến quý báu và cung cấp tài liệu cần thiết để em hoàn thành đồ án này. Sự chỉ dẫn tận tình và kiến thức sâu rộng của thầy đã giúp em vượt qua nhiều khó khăn, hiểu rõ hơn về chủ đề nghiên cứu và phát triển những ý tưởng mới mẻ. Thầy không chỉ là người truyền đạt tri thức mà còn là nguồn động viên lớn lao, luôn khuyến khích tôi nỗ lực và kiên trì theo đuổi mục tiêu của mình. Sự ủng hộ và tin tưởng của thầy đã truyền cảm hứng để em hoàn thành tốt công việc của mình.

Đồng thời, em không quên gửi lời cảm ơn chân thành đến các bạn sinh viên cùng khóa, những người đã cùng em chia sẻ kinh nghiệm, hỗ trợ lẫn nhau trong suốt quá trình học tập và nghiên cứu giúp em có thêm động lực để vượt qua những thử thách trong quá trình làm đồ án.

Cuối cùng, em xin bày tỏ lòng biết ơn sâu sắc đến gia đình, những người đã luôn ở bên, động viên và tạo mọi điều kiện thuận lợi để em có thể tập trung hoàn thành đồ án này. Sự quan tâm và chăm sóc của gia đình là động lực to lớn giúp em vượt qua mọi khó khăn và thử thách.

Em xin chân thành cảm ơn!

**MỤC LỤC**

MỞ ĐẦU .....	1
CHƯƠNG 1   CƠ SỞ LÝ THUYẾT .....	2
<b>1.1   Hệ điều hành Android</b> .....	2
1.1.1   Khái niệm.....	2
1.1.2   Lịch sử phát triển .....	2
1.1.3   Kiến trúc hệ điều hành Android.....	2
1.1.4   Kiến trúc ứng dụng Android .....	5
<b>1.2   OWASP Mobile Top 10</b> .....	5
1.2.1   Khái niệm.....	5
1.2.2   Top 10 lỗ hổng mobile 2024 dựa theo OWASP .....	5
<b>1.3   Kiểm thử ứng dụng Android</b> .....	10
1.3.1   Khái niệm.....	10
1.3.2   Phương pháp kiểm thử.....	10
1.3.3   Các giai đoạn kiểm thử .....	11
CHƯƠNG 2   XÂY DỰNG KỊCH BẢN .....	12
<b>2.1   Kịch bản</b> .....	12
2.1.1   Insecure Authentication/Authorization .....	12
2.1.2   Insufficient Input/Output Validation .....	13
2.1.3   Insecure Communication .....	14
2.1.4   Insufficient Binary Protection.....	16
2.1.5   Insufficient Cryptography .....	18
<b>2.2   Công cụ</b> .....	20
2.2.1   Burp suite .....	20
2.2.2   JADX .....	20
2.2.3   MTManager .....	21
2.2.4   NoxPlayer .....	22
CHƯƠNG 3   THỰC NGHIỆM .....	23
3.1   Insecure Authentication/Authorization.....	23
3.2   Insufficient Input/Output Validation .....	25
3.3   Insecure Communication.....	27
3.4   Insufficient Binary Protection .....	29
3.5   Insufficient Cryptography.....	36
KẾT LUẬN .....	40
TÀI LIỆU THAM KHẢO .....	41

**DANH MỤC HÌNH**

Hình 1.1 Sơ đồ cấu trúc hệ điều hành android .....	3
Hình 2.1 Giao diện đăng nhập của ứng dụng kịch bản 1.....	12
Hình 2.2 Giao diện đăng nhập thành công của ứng dụng kịch bản 1 .....	13
Hình 2.3 Hàm đăng nhập của Firebase.....	13
Hình 2.4 Giao diện đăng nhập của ứng dụng kịch bản 2.....	14
Hình 2.5 Giao diện đăng nhập thành công của ứng dụng kịch bản 2 .....	14
Hình 2.6 Đoạn mã chứa lỗ hổng bảo mật sql injection .....	14
Hình 2.7 Giao diện đăng nhập của ứng dụng kịch bản 3.....	15
Hình 2.8 Giao diện đăng nhập thành công của ứng dụng kịch bản 3 .....	15
Hình 2.9 Hàm đăng nhập của Firebase.....	15
Hình 2.10 Giao diện đăng nhập của ứng dụng kịch bản 4.....	16
Hình 2.11 Giao diện đăng nhập thành công của ứng dụng kịch bản 4.....	17
Hình 2.12 Giao diện quên mật khẩu của ứng dụng kịch bản 4.....	17
Hình 2.13 Giao diện xác minh OTP của ứng dụng kịch bản 4.....	18
Hình 2.14 Giao diện đăng nhập của ứng dụng kịch bản 5.....	19
Hình 2.15 Giao diện đăng nhập thành công của ứng dụng kịch bản 5 .....	19
Hình 2.16 Logo Burpsuit.....	20
Hình 2.17 Logo JADX-GUI.....	20
Hình 2.18 Logo MT manager .....	21
Hình 2.19 Logo NOX .....	22
Hình 3.1 Giao diện đăng nhập .....	23
Hình 3.2 Điền thông tin đăng nhập .....	23
Hình 3.3 Bắt request bằng intercept .....	23
Hình 3.4 Đưa request sang Intruder .....	24
Hình 3.5 Chọn giá trị trường password để bruteforce .....	24
Hình 3.6 Điều chỉnh thông số tấn công .....	24
Hình 3.7 Kết quả bruteforce .....	25
Hình 3.8 Đăng nhập thành công .....	25
Hình 3.9 Giao diện đăng nhập .....	26
Hình 3.10 Tấn công SQL Injection .....	26
Hình 3.11 Tấn công thành công.....	26
Hình 3.12 Mã sửa lỗi sql injection .....	26
Hình 3.13 Lỗi SQL Injection đã được khắc phục.....	27
Hình 3.14 Giao diện đăng nhập .....	27
Hình 3.15 Điền thông tin đăng nhập .....	27
Hình 3.16 Thông tin đăng nhập trong request.....	28
Hình 3.17 Hàm băm sha256 .....	28
Hình 3.18 Mật khẩu đã được băm .....	29
Hình 3.19 Giao diện JADX - GUI.....	29
Hình 3.20 Kiểm tra OTP bằng hàm equal giữa thông tin nhập vào và hàm getOTP() .....	30
Hình 3.21 Thông tin hàm getOTP() .....	30
Hình 3.22 Sử dụng MT manager để chỉnh file .dex .....	31
Hình 3.23 Kí file sau khi đã sửa .....	31
Hình 3.24 Cài đặt file apk đã chỉnh sửa .....	32
Hình 3.25 Điền email victim .....	32
Hình 3.26 Nhập mã OTP mà ta đã gán vào .....	33
Hình 3.27 Bypass thành công xác thực OTP .....	34
Hình 3.28 Đăng nhập thành công vào tài khoản victim .....	35
Hình 3.29 Kích hoạt Minify trong file build.gradle.kts.....	35
Hình 3.30 Rule cấu hình Proguard .....	36

Hình 3.31 File decompile sau khi obfuscate.....	36
Hình 3.32 Trang đăng nhập .....	37
Hình 3.33 Giao diện burpsuit bắt các request.....	37
Hình 3.34 Thông tin mật khẩu đã được băm .....	37
Hình 3.35 Băm mật khẩu bằng MD5.....	38
Hình 3.36 Hàm băm MD5 .....	38
Hình 3.37 Sử dụng công cụ 10015 để tìm hash collision.....	38
Hình 3.38 Đăng nhập vào tài khoản victim .....	39



## MỞ ĐẦU

Ngày nay, với sự phát triển nhanh chóng của công nghệ những chiếc điện thoại thông minh đang ngày càng trở nên hữu ích, tiện lợi hơn với hàng trăm, hàng ngàn ứng dụng làm đơn giản hóa rất nhiều công việc trong cuộc sống, với một chiếc điện thoại thông minh ta đã có thể nhắn tin, đặt báo thức, nghe nhạc, xem phim, đặt vé, đặt phòng, mua sắm, thanh toán hóa đơn và thực hiện nhiều hoạt động khác mà không cần di chuyển. Điều này không chỉ tiết kiệm thời gian mà còn mang lại nhiều tiện ích đáng kể cho người dùng.

Tuy nhiên, không phải bất kỳ ứng dụng nào cũng an toàn. Các ứng dụng trên điện thoại di động luôn tồn tại một hoặc nhiều lỗ hổng bảo mật đó là nơi để các hacker tấn công nhằm đánh cắp thông tin hoặc gây thiệt hại đến tài sản của chủ sở hữu ứng dụng hoặc người sử dụng ứng dụng đó. Trong bối cảnh này, việc bảo vệ thông tin cá nhân và tài sản trở nên cực kỳ quan trọng.

Để có thể làm giảm thiểu rủi ro khi sử dụng các ứng dụng di động đối với người sử dụng và tránh các lỗ hổng trong quá trình lập trình nên một ứng dụng di động. Trong bài báo cáo này, tôi sẽ tiến hành liệt kê và kiểm thử một số lỗ hổng dựa trên top 10 lỗ hổng nổi bật trên ứng dụng di động đặc biệt là ứng dụng trên nền tảng Android (Android là một hệ điều hành trên di động được sử dụng nhiều nhất trên thế giới). Danh sách này được dựa theo tiêu chuẩn của OWASP (Open Worldwide Application Security Project), một tổ chức phi lợi nhuận chuyên về bảo mật ứng dụng.

Pentest (kiểm thử xâm nhập) trên ứng dụng Android không chỉ giúp phát hiện ra các lỗ hổng bảo mật mà còn cung cấp các biện pháp phòng ngừa và khắc phục. Qua quá trình này, các nhà phát triển có thể cải thiện chất lượng và độ an toàn của ứng dụng, đồng thời bảo vệ người dùng khỏi các cuộc tấn công tiềm ẩn.

Bài báo cáo này có cấu trúc gồm các chương như sau:

Chương 1: Cơ sở lý thuyết, giới thiệu về các lỗ hổng, giới thiệu phương pháp pentest

Chương 2: Xây dựng kịch bản

Chương 3: Thực nghiệm

## CHƯƠNG 1 CƠ SỞ LÝ THUYẾT

### 1.1 Hệ điều hành Android

#### 1.1.1 Khái niệm

- Android là một hệ điều hành di động được xây dựng dựa trên phiên bản sửa đổi của nhân linux và phần mềm mã nguồn mở khác, được thiết kế chủ yếu cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng.
- Andorid là một hệ điều hành mã nguồn mở. Mã nguồn của nó thuộc về dự án Android Open Source Project, giấy phép chính được cấp phép bởi Apache License. Tuy nhiên, phần lớn các thiết bị android điều được cài sẵn các phần mềm độc quyền bổ sung, chủ yếu là dịch vụ di động của Google (Google Mobile Service) bao gồm các ứng dụng cốt lõi như google Chrome, Google Play và nền tảng phát triển dịch vụ Google Play liên quan

#### 1.1.2 Lịch sử phát triển

- Năm 2003, Android được ra mắt như là một dự án của công ty công nghệ Mỹ Android Inc., nhằm phát triển hệ điều hành cho điện thoại kỹ thuật số.
- Năm 2004, dự án thay đổi để trở thành hệ điều hành cho điện thoại thông minh.
- Năm 2005, Android Inc. được mua lại bởi công ty công cụ tìm kiếm Google Inc. của Mỹ. Tại đây nhóm phát triển Android đã quyết định sẽ xây dựng dự án dựa trên hệ điều hành Linux
- 05/11/2007, Google công bố thành lập Open Handset Alliance, một tập đoàn gồm hàng chục công ty công nghệ và điện thoại di động, bao gồm Intel Corporation, Motorola Inc., NVIDIA Corporation, Texas Instruments Incorporated, LG Electronics, Inc., Samsung Electronics, Sprint Nextel Corporation, and T-Mobile (Deutsche Telekom). Hiệp hội được thành lập nhằm phát triển và quảng bá Android như một hệ điều hành mã nguồn mở miễn phí có hỗ trợ các ứng dụng của bên thứ ba.
- 22/10/2008, điện thoại di động đầu tiên có hệ điều hành mới là T-Mobile G1[1]
- Năm 2012, Android trở thành hệ điều hành phổ biến nhất dành cho thiết bị di động, vượt qua IOS của Apple và tính đến 05/2024, khoảng 71,5% thiết bị di động chạy hệ điều hành Android [2].

#### 1.1.3 Kiến trúc hệ điều hành Android

- Hệ điều hành Android chứa một số thành phần khác nhau để hỗ trợ mọi nhu cầu của thiết bị Android. Phần mềm Android chứa nhân mã nguồn mở Linux với nhiều thư viện C/C++ được hiển thị thông qua các dịch vụ khung ứng dụng (application framework services)



Hình 1.1 Sơ đồ cấu trúc hệ điều hành android

- Trong tất cả các thành phần, Linux Kernel cung cấp các chức năng chính của hệ điều hành cho điện thoại thông minh và máy ảo Dalvik cung cấp nền tảng chạy ứng dụng Android. Hệ điều hành android là một tập hợp các thành phần phần mềm được chia thành năm phần và bốn lớp chính:

#### **Applications**

- Application là tầng trên cùng của kiến trúc hệ điều hành Android. Các ứng dụng như máy ảnh, thư viện, doanh bạ, v.v và các ứng dụng của bên thứ ba được tải xuống từ cửa hàng CHPlay như trò chơi, ứng dụng nghe nhạc, v.v sẽ được cài đặt trên lớp này.
- Lớp Application hoạt động cùng với Android run time với sự trợ giúp của các lớp và dịch vụ do Application Framework cung cấp.

#### **Application Framework**

- Application Framework cung cấp một số class quan trọng được sử dụng để tạo ứng dụng Android. Nó cung cấp một lớp trừu tượng chung cho việc truy cập phần cứng và giúp quản lý giao diện người dùng bằng cách sử dụng tài nguyên ứng dụng. Nói chung, nó cung cấp các dịch vụ giúp chúng ta có thể tạo ra một class cụ thể và sử dụng class đó để tạo ra các ứng dụng.
- Nó bao gồm nhiều loại dịch vụ khác nhau như là activity manager, notification manager, view system, package manager v.v rất hữu ích cho việc phát triển ứng dụng của chúng ta theo các yêu cầu tiên quyết.
- Lớp Application Framework cung cấp nhiều dịch vụ cao cấp hơn cho các ứng dụng dưới dạng các class java. Các nhà phát triển ứng dụng được phép sử dụng các dịch vụ này trong các ứng dụng. Android framework bao gồm các dịch vụ chính sau:
  - + Activity Manager: kiểm soát toàn bộ các khía cạnh của vòng đời ứng dụng và hoạt động của stack
  - + Content Providers: cho phép ứng dụng công bố và chia sẻ dữ liệu với các ứng dụng khác
  - + Resource Manager: cung cấp quyền truy cập vào các tài nguyên không phải là code như chuỗi, cài đặt màu sắc và bố cục giao diện người dùng

+ Notification Manager: cho phép ứng dụng hiển thị thông báo và cảnh báo tới người dùng

+ View System: Một tập hợp các chế độ xem có thể mở rộng được sử dụng để tạo giao diện người dùng ứng dụng

### **Android Runtime**

- Môi trường Android Runtime chứa các thành phần như là các thư viện cốt lõi và máy ảo Dalvik. Nó cung cấp nền tảng cho Application Framework và hỗ trợ ứng dụng của chúng ta với sự trợ giúp của các thư viện cốt lõi.
- Giống như máy ảo Java, máy ảo Dalvik là máy ảo dựa trên thanh ghi, được thiết kế và tối ưu hoá cho Android để đảm bảo rằng một thiết bị có thể chạy nhiều phiên bản một cách hiệu quả.
- Nó phụ thuộc vào lớp Linux Kernel để phân luồng và quản lý bộ nhớ cấp thấp. Các thư viện cốt lõi cho phép ta triển khai các ứng dụng Android bằng ngôn ngữ lập trình Java hoặc Kotlin

### **Platform Libraries**

- Lớp này bao gồm các thư viện C/C++ cốt lõi và các thư viện bằng java như Media, Graphics, Surface Manager, OpenGL, etc. để hỗ trợ phát triển Android
- + app: Cung cấp quyền truy cập vào mô hình ứng dụng và là nền tảng của mọi ứng dụng Android
- + content: Tạo điều kiện truy cập nội dung, xuất bản và nhắn tin giữa các ứng dụng và các thành phần ứng dụng
- + database: Được sử dụng để truy cập vào dữ liệu được public bởi nhà cung cấp và bao gồm cơ sở dữ liệu SQLite, các lớp quản lý
- + OpenGL: Giao diện Java cho API kết xuất đồ hoạ OpenGL ES 3D
- + os: Cung cấp cho ứng dụng quyền truy cập vào các dịch vụ hệ điều hành tiêu chuẩn, bao gồm tin nhắn, các dịch vụ hệ thống và giao tiếp giữa các tiến trình
- + text: được sử dụng để hiển thị và thao tác văn bản trên màn hình thiết bị
- + view: Các khối xây dựng cơ bản của giao diện người dùng ứng dụng
- + widget: Một bộ sưu tập phong phú các thành phần giao diện người dùng dựng sẵn như nút, nhãn, chế độ xem danh sách, trình quản lý bố cục, nút radio, v.v
- + Webkit: Một tập hợp các lớp nhằm cho phép khả năng duyệt web được tích hợp vào các ứng dụng
- + media: Thư viện media hỗ trợ phát và ghi định dạng âm thanh và video
- + surface manager: Nó chịu trách nhiệm quản lý truy cập vào hệ thống hiển thị phụ
- + SQLite: Hỗ trợ cơ sở dữ liệu và FreeType hỗ trợ font chữ
- + SSL: Là công nghệ bảo mật nhằm thiết lập liên kết được mã hoá giữa máy chủ web và trình duyệt web

### **Linux Kernel**

- Là thành phần cốt lõi của kiến trúc Android. Nó quản lý tất cả các trình điều khiển có sẵn như màn hình, máy ảnh, Bluetooth, âm thanh, bộ nhớ, v.v, được yêu cầu trong Runtime
- Cung cấp một lớp trừu tượng giữa phần cứng thiết bị và các thành phần kiến trúc Android khác. Nó chịu trách nhiệm quản lý bộ nhớ, nguồn, thiết bị, v.v
- Các tính năng:
  - + Security: Xử lý bảo mật giữa ứng dụng và hệ thống

- + Memory Management: Quản lý bộ nhớ từ đó mang lại sự tự do phát triển ứng dụng
- + Process Management: Quản lý các tiến trình, phân bổ tài nguyên cho bất kì tiến trình nào khi chúng cần
- + Network Stack: Xử lý giao tiếp mạng
- + Driver Model: Đảm bảo ứng dụng hoạt động bình thường trên thiết bị

#### 1.1.4 Kiến trúc ứng dụng Android

- Tập APK (Android Package Kit) là tập lưu trữ sử dụng cho hệ điều hành Android. Công dụng chính của nó là phân phối và cài đặt các ứng dụng trên thiết bị Android. Tập APK chứa tất cả các thành phần cần thiết để cài đặt và chạy ứng dụng. Các ứng dụng được cài đặt trên thiết bị Android thông qua tập APK sẽ được cài đặt trên phân vùng hệ thống của thiết bị.
- Các thành phần chính của tập APK:
  - + MANIFEST.MF: Chứa danh sách tên và hàm băm (thường là SHA256 được mã hóa Base64) cho tất cả các tập trong APK.
  - + AndroidManifest.xml: Tập kê khai mô tả tên gói, hoạt động, quyền, tài nguyên, phiên bản, và các thành phần khác của ứng dụng.
  - + assets: Chứa nội dung mà nhà phát triển gói kèm với ứng dụng. AssetManager có thể truy xuất các tài sản này, bao gồm hình ảnh, video, tài liệu, cơ sở dữ liệu, v.v
  - + lib: Chứa các thư viện gốc có mã được biên dịch cho các kiến trúc phần cứng thiết bị khác nhau.
  - + res: Chứa các tài nguyên ứng dụng không biên dịch trước, như các tập XML xác định danh sách trạng thái màu sắc, bố cục giao diện người dùng, phông chữ, giá trị, v.v.
  - + resources.arsc: Chứa các tài nguyên đã được biên dịch. Tập này liên kết mã với tài nguyên.
  - + class.dex: Chứa tất cả các lớp Java ở định dạng tập dex (Dalvik Executable), được Android Runtime thực thi.

### 1.2 OWASP Mobile Top 10

#### 1.2.1 Khái niệm

- OWASP viết tắt của Open Worldwide Application Security Project – Dự án bảo mật ứng dụng toàn cầu mở. Là một cộng đồng trực tuyến tạo ra các bài viết, phương pháp, tài liệu, công cụ và công nghệ miễn phí trong lĩnh vực bảo mật IOT, phần mềm hệ thống, phần mềm web.

#### 1.2.2 Top 10 lỗ hổng mobile 2024 dựa theo OWASP

##### **M1: Improper Credential Usage**

- Đây là lỗ hổng sử dụng sai thông tin xác thực, bao gồm mã hoá cứng thông tin xác thực, lưu trữ thông tin xác thực dưới dạng văn bản thuần túy hoặc không xác thực thông tin đúng cách. Điều này có thể dẫn đến vi phạm quyền riêng tư của người dùng và truy cập trái phép vào dữ liệu và hệ thống.
- Một số biểu hiện:
  - + Mã hoá cứng thông tin xác thực: Thông tin xác thực được mã hoá cứng trong mã nguồn của ứng dụng hoặc trong các tệp cấu hình
  - + Lưu trữ thông tin xác thực không an toàn: Thông tin xác thực của người dùng được lưu ở cục bộ, ở các vùng dễ dàng truy cập

- + Sử dụng chính sách mật khẩu yếu: Các ứng dụng không sử dụng các chính sách mật khẩu mạnh sẽ dễ bị tấn công brute force và bẻ khoá hơn
- + Mã hoá không đầy đủ: Các thông tin xác thực được truyền đi mà không được mã hoá hoặc mã hoá yếu
- + Quy trình xác thực không an toàn: Thiếu các quy trình xác thực an toàn như xác thực đa yếu tố
- + Không triển khai quản lý phiên: Phiên đăng nhập không hết hạn sau khi không hoạt động hoặc thời gian tồn tại của phiên quá lâu
- **Rủi ro:**
  - + Đánh cắp thông tin: Truy cập trái phép vào thông tin cá nhân có thể dẫn đến hành vi đánh cắp thông tin cá nhân, gây thiệt hại về tài chính và danh tiếng cho người dùng
  - + Gian lận tài chính: Việc sử dụng thông tin xác thực không đúng cách có thể dẫn đến các giao dịch trái phép và các hành vi xâm phạm vào dữ liệu tài chính nhạy cảm
  - + Vi phạm quyền riêng tư: Lỗ hổng thông tin xác thực có thể dẫn đến vi phạm quyền riêng tư, xâm phạm dữ liệu người dùng
  - + Thiệt hại về danh tiếng: Ảnh hưởng đến danh tiếng của ứng dụng và nhà phát triển ứng dụng.

### **M2: Inadequate Supply Chain Security**

- Đây là lỗ hổng trong quá trình xây dựng, phát triển và cung ứng ứng dụng, kẻ tấn công có thể thao túng chức năng ứng dụng bằng cách khai thác các lỗ hổng trong chuỗi cung ứng ứng dụng di động.
- **Một số biểu hiện:**
  - + Các lỗ hổng trong vòng đời ứng dụng: Quá trình cung ứng ứng dụng trải qua rất nhiều giai đoạn từ giai đoạn phát triển đến giai đoạn triển khai. Bất kì lỗ hổng nào trong khâu bảo mật ở bất kì giai đoạn nào đều có thể mở ra cơ hội tấn công cho các tin tặc.
  - + Sự phụ thuộc của bên thứ ba và mã nguồn mở: Các ứng dụng di động thường dựa vào các thư viện, frame và thành phần nguồn mở của bên thứ ba để nâng cao chức năng và hợp lý hoá quá trình phát triển. Nếu các yếu tố này không được theo dõi và cập nhật thường xuyên chúng có thể trở thành các lỗ hổng tiềm ẩn.
  - + Rủi ro trong nội bộ: Sự xâm phạm của người trong nội bộ dù cố ý hay vô ý điều có thể tạo điều kiện cho việc truy cập trái phép vào các thành phần quan trọng của quy trình phát triển phần mềm như các kho lưu trữ mã nguồn, môi trường xây dựng
  - + Phishing và social engineering: Các cuộc tấn công social engineering đặc biệt là lừa đảo là một nguyên nhân phổ biến dẫn đến vi phạm an ninh quá trình cung ứng. Nếu các nhà phát triển hoặc nhân viên tham gia chuỗi cung ứng trở thành nạn nhân của các cuộc tấn công lừa đảo, thông tin xác thực bị đánh cắp có thể cấp quyền cho kẻ tấn công truy cập trái phép vào các hệ thống quan trọng.
- **Rủi ro:**
  - + Vi phạm dữ liệu: Kẻ tấn công có thể đánh cắp các thông tin quan trọng như thông tin đăng nhập, thông tin cá nhân, thông tin tài chính
  - + Lây nhiễm mã độc: Kẻ tấn công có thể chèn các mã độc vào ứng dụng từ đó đánh cắp thông tin hoặc thực hiện các hành vi độc hại cho người dùng ứng dụng

+ Ảnh hưởng đến danh tiếng: Nhà phát triển ứng dụng sẽ phải đối mặt với sự mất lòng tin đến từ người dùng và có thể phải đối mặt với các hậu quả pháp lý và quy định do cuộc tấn công gây ra, chẳng hạn như phạt tiền, kiện tụng hoặc điều tra của chính phủ.

### **M3: Insecure Authentication/Authorization**

- Xác thực và uỷ quyền là trụ cột cơ bản của bất kỳ hệ thống bảo mật nào bao gồm cả ứng dụng di động. Việc xác thực và uỷ quyền không an toàn có thể dẫn đến vô số rủi ro, bao gồm truy cập trái phép, vi phạm dữ liệu và tài khoản người dùng bị xâm phạm, thao túng thông tin nhạy cảm và xâm phạm tính toàn vẹn của ứng dụng.
- Một số biểu hiện:
  - + Xác thực yếu: ứng dụng sử dụng cơ chế xác thực một yếu tố chỉ yêu cầu email và mật khẩu, sử dụng các chính sách mật khẩu yếu.
  - + Thiếu sót trong uỷ quyền: Cho phép người dùng bình thường có các quyền hạn của quản lý như truy cập và chỉnh sửa các thông tin nhạy cảm như lịch sử mua hàng, thay đổi thông tin đơn hàng
- Rủi ro:
  - + Đánh cắp thông tin: Việc sử dụng cơ chế xác thực đơn giản và cho phép sử dụng mật khẩu yếu có thể dẫn đến tấn công bruteforce
  - + Truy trái phép vào dữ liệu: Uỷ quyền sai hoặc thiếu sót có thể dẫn đến việc truy cập đến các thông tin nhạy cảm của doanh nghiệp
  - + Gian lận: Cho phép người dùng chỉnh sửa các thông tin như lịch sử mua hàng, đơn hàng có thể dẫn đến gian lận mua bán ảnh hưởng đến tài chính.

### **M4: Insufficient Input/Output Validation**

- Đây là lỗ hổng về việc thiếu xác thực hoặc xác thực chưa đầy đủ từ các nguồn bên ngoài, chẳng hạn như thông tin đầu vào của người dùng hoặc dữ liệu mạng trong ứng dụng di động có thể gây ra các lỗ hổng bảo mật nghiêm trọng.
- Một số biểu hiện:
  - + Xác thực đầu vào không đầy đủ: Khi đầu vào của người dùng không được kiểm tra kỹ lưỡng, kẻ tấn công có thể thao túng nó bằng cách nhập dữ liệu độc hại hoặc không mong muốn.
  - + Xác thực đầu ra không đầy đủ: Nếu dữ liệu đầu ra không được xác thực và chuẩn hoá đúng cách, kẻ tấn công có thể chen các mã độc và thực thi trên trình duyệt của người dùng.
  - + Không xác thực tính toán vẹn dữ liệu: Nếu không có xác thực tính toán vẹn dữ liệu ứng dụng sẽ dễ bị hỏng dữ liệu hoặc xử lý không chính xác.
- Rủi ro:
  - + Thực thi mã độc: Kẻ tấn công có thể khai thác lỗ hổng này để thực thi mã trái phép trong môi trường của ứng dụng, bỏ qua các biện pháp bảo mật
  - + Vi phạm dữ liệu: Xác thực không đầy đủ có thể cho phép kẻ tấn công thao túng dữ liệu đầu vào, có khả năng dẫn đến truy cập và khai thác trái phép dữ liệu nhạy cảm
  - + Giảm đoạn hoạt động của ứng dụng: Đầu vào có thể làm hỏng dữ liệu gây ra gián đoạn, treo máy, làm ứng dụng mất khả năng hoạt động
  - + Thiệt hại về danh tiếng: Khai thác thành công lỗ hổng này có thể gây tổn hại về danh tiếng cho vi phạm dữ liệu và mất niềm tin của khách hàng

**M5: Insecure Communication**

- Đây là lỗ hổng về việc truyền tải các dữ liệu nhạy cảm thông qua các kênh không an toàn, dữ liệu chưa được mã hoá hoặc sử dụng các giao thức mã hoá lỗi thời điều này làm cho quá trình giao tiếp dễ bị tấn công bởi các phương thức như nghe lén, tấn công Man-in-the-middle.
- Một số biểu hiện:
  - + Thiếu mã hoá: các dữ liệu nhạy cảm như thông tin đăng nhập, thông tin tài khoản ngân hàng không đã mã hoá trước khi gửi đi có thể dẫn đến lộ thông tin
  - + Sử dụng các chứng chỉ SSL không an toàn: Ứng dụng chấp nhận sử dụng chứng chỉ không an toàn như các chứng chỉ tự ký, đã hết hạn, không đúng máy chủ.
  - + Sử dụng các giao thức không an toàn: Ứng dụng sử dụng các giao thức không an toàn để truyền các thông tin nhạy cảm như các giao thức lỗi thời hoặc mã hoá yếu
- Rủi ro:
  - + Lộ thông tin cá nhân: Các lỗ hổng này có thể dẫn đến các kiểu tấn công nghe lén làm lộ thông tin người dùng, chiếm tài khoản, giả mạo, vv
  - + Lộ thông tin nhạy cảm: Làm lộ các thông tin nhạy cảm của doanh nghiệp ảnh hưởng đến danh tiếng hoặc có thể ảnh hưởng đến tài chính
  - + Tấn công Man-in-the-middle: Sử dụng các giao thức không an toàn sẽ tạo điều kiện cho kẻ tấn công có thể chỉnh sửa dữ liệu, thông tin gây thiệt hại về thông tin hoặc tài sản.

**M6: Inadequate Privacy Controls**

- Đây là lỗ hổng liên quan đến việc không có các bước vững chắc để giữ an toàn cho thông tin người dùng trong ứng dụng dành cho thiết bị di động như các loại mã hoá phức tạp, thiếu kiểm soát quyền truy cập, quản lý phiên không hiệu quả.
- Một số biểu hiện:
  - + Mã hoá yếu: Thiếu các cơ chế mã hoá có thể dẫn đến việc chỉnh sửa dữ liệu, đánh cắp dữ liệu.
  - + Quản lý phiên kém: Các phiên có thời gian hoạt động dài, phiên không tự hết hạn khi không được sử dụng có thể dẫn đến các tấn công chiếm phiên đăng nhập
  - + Cơ chế chấp thuận không đầy đủ: Các ứng dụng thường yêu cầu quyền truy cập vào nhiều chức năng của thiết bị và dữ liệu người dùng. Thiếu cơ chế chấp thuận có thể dẫn đến việc ứng dụng thu thập nhiều dữ liệu hơn mức cần thiết hoặc không có sự cho phép của người dùng.
- Rủi ro:
  - + Dữ liệu người dùng bị thao túng: Nó có thể khiến hệ thống không thể sử dụng được đối với người dùng đó
  - + Vi phạm các quy định pháp luật: Có thể dẫn đến vi phạm các luật về quyền riêng tư như Chính sách quyền riêng tư (Việt Nam), GDPR (Châu Âu), LGPD (Hoa kì), v.v
  - + Thiệt hại về danh tiếng: Nếu hành vi vi phạm quyền riêng tư ảnh hưởng đến người dùng trên quy mô lớn thì hành vi đó có thể xuất hiện trên các phương tiện truyền thông, do đó tạo ra dư luận tiêu cực cho nhà cung cấp ứng dụng

**M7: Insufficient Binary Protection**

- Đây là lỗ hổng về việc thiếu các biện pháp bảo vệ file nhị phân dẫn đến lộ các dữ liệu bí mật như khoá API, các logic kinh doanh quan trọng hoặc các mô hình AI được đào



tạo trước, ngoài ra tệp nhị phân có thể bị thao túng để truy cập các tính năng trả phí miễn phí hoặc bỏ qua các bước kiểm tra bảo mật khác.

- Một số biểu hiện:
  - + Thiếu biện pháp kiểm tra tính toàn vẹn: Thiếu các biện pháp kiểm tra tính toàn vẹn của tệp nhị phân có thể dẫn đến file bị điều chỉnh, sửa đổi hoặc chen mã độc
  - + Thiếu biện pháp chống dịch ngược: Thiếu các biện pháp chống dịch ngược có thể dẫn đến tệp nhị phân bị dịch ngược và làm lộ các thông tin như logic, loại mã hoá, khoá API.
- Rủi ro:
  - + Thao túng ứng dụng: Các kẻ tấn công có thể chỉnh sửa tệp nhị phân để nó thực thi theo ý đồ của kẻ tấn công như vượt qua các bước kiểm tra bảo mật, mở khoá các chức năng trả phí
  - + Rò rỉ thông tin: Rò rỉ khoá API cho các API thương mại có thể gây ra chi phí đáng kể nếu chúng bị lạm dụng trên quy mô lớn.
  - + Giả mạo ứng dụng: Việc thiếu kiểm tra tính toàn vẹn có thể khiến tệp nhị phân bị chỉnh sửa như thêm mã độc

#### **M8: Security Misconfiguration**

- Đây là lỗi hổng đến việc cấu hình sai cài đặt bảo mật, quyền và các biện pháp kiểm soát có thể dẫn đến lỗi hổng và truy cập trái phép.
- Một số biểu hiện:
  - + Cài đặt mặc định không an toàn: Ứng dụng di động thường có các cấu hình mặc định, các cấu hình này có thể là các cài đặt bảo mật yếu hoặc bật các quyền không cần thiết
  - + Mã hoá hoặc băm yếu: Các thuật toán mã hoá sử dụng key đơn giản hoặc sử dụng chung key và các hàm băm yếu như md5, sha1 có thể trở thành đối tượng tấn công
  - + Quyền truy cập tệp không an toàn: Lưu trữ các tệp ứng dụng với các quyền có thể đọc hoặc ghi trên toàn cục
- Rủi ro:
  - + Truy cập trái phép vào dữ liệu nhạy cảm: Cấu hình sai có thể cho phép kẻ tấn công truy cập thông tin nhạy cảm, chẳng hạn như thông tin xác thực người dùng, dữ liệu cá nhân hoặc dữ liệu kinh doanh bí mật.
  - + Chiếm đoạt tài khoản hoặc mạo danh: Cơ chế xác thực yếu hoặc được định cấu hình sai có thể dẫn đến việc chiếm đoạt tài khoản hoặc mạo danh người dùng hợp pháp.
  - + Vi phạm dữ liệu: Cấu hình bảo mật không đầy đủ có thể dẫn đến vi phạm dữ liệu, làm lộ dữ liệu nhạy cảm cho các cá nhân không được ủy quyền.
  - + Xâm phạm hệ thống phụ trợ: Cấu hình sai trong ứng dụng di động có thể tạo cơ hội cho kẻ tấn công xâm phạm hệ thống hoặc cơ sở hạ tầng phụ trợ.

#### **M9: Insecure Data Storage**

- Việc lưu trữ dữ liệu không an toàn trong ứng dụng di động có thể thu hút nhiều tác nhân đe dọa khác nhau nhằm khai thác các lỗi hổng và giành quyền truy cập trái phép vào thông tin nhạy cảm.
- Biểu hiện phổ biến: việc lưu trữ dữ liệu không an toàn có thể biểu hiện trong nhiều tình huống khác nhau, bao gồm mã hóa không đủ, kiểm soát truy cập yếu và xử lý thông tin xác thực của người dùng không đúng cách.
- Rủi ro:

- + Truy cập trái phép: Dữ liệu không được bảo mật đầy đủ có thể dễ bị truy cập trái phép, ảnh hưởng đến tính bảo mật của thông tin nhạy cảm
- + Vi phạm dữ liệu: Mã hóa yếu hoặc không mã hóa ở khi lưu trữ có thể dẫn đến vi phạm dữ liệu, tiết lộ thông tin người dùng
- + Rò rỉ thông tin nhạy cảm: Việc bảo vệ không đầy đủ có thể dẫn đến rò rỉ thông tin nhạy cảm, ảnh hưởng đến quyền riêng tư của người dùng.

#### **M10: Insufficient Cryptography**

- Mật mã là nền tảng để bảo mật thông tin nhạy cảm trong các ứng dụng di động. Mật mã không đầy đủ biểu thị những điểm yếu trong việc triển khai các cơ chế mật mã, khiến dữ liệu dễ bị truy cập trái phép hoặc giả mạo.
- Biểu hiện chính: Mật mã không đầy đủ có thể biểu hiện theo nhiều cách khác nhau, bao gồm thuật toán mã hóa yếu, quản lý khóa không đúng hoặc thiếu cơ chế bảo vệ khóa mật mã.
- Rủi ro:
  - + Tấn công trung gian: Việc bảo vệ không đầy đủ có thể mở ra con đường cho các cuộc tấn công trung gian, cho phép kẻ tấn công chặn và thao túng dữ liệu.
  - + Tấn công bruteforce: Các thuật toán mã hóa yếu có thể dễ bị tấn công vũ phu, trong đó những kẻ tấn công cố gắng giải mã dữ liệu được mã hóa một cách có hệ thống.
  - + Rò rỉ khóa mật mã: Việc lưu trữ hoặc truyền khóa mật mã không an toàn có thể dẫn đến rò rỉ, ảnh hưởng đến tính bảo mật của dữ liệu được mã hóa.

### **1.3 Kiểm thử ứng dụng Android**

#### **1.3.1 Khái niệm**

- Kiểm thử bảo mật ứng dụng Android là quá trình đánh giá và xác minh các biện pháp bảo mật trong các ứng dụng Android nhằm đảm bảo rằng chúng an toàn trước các mối đe dọa và tấn công tiềm tàng. Mục tiêu chính của kiểm thử bảo mật là phát hiện và khắc phục các lỗ hổng bảo mật để bảo vệ dữ liệu người dùng và ngăn chặn các hành vi tấn công độc hại.

#### **1.3.2 Phương pháp kiểm thử**

##### **Black Box Testing**

- Đây là phương pháp kiểm thử phần mềm trong đó chức năng của phần mềm chưa được biết đến đối với người kiểm thử. Người kiểm thử không có thông tin nội bộ về cấu trúc hoặc thiết kế của sản phẩm mà chỉ dựa trên các yêu cầu và thông số kỹ thuật để thực hiện kiểm thử. Điều này tương tự như một người dùng bình thường hoặc một kẻ tấn công từ bên ngoài, không có kiến thức về mã nguồn hay cấu trúc bên trong của ứng dụng.
- Mục tiêu: Phát hiện các lỗ hổng và lỗi bảo mật thông qua các cuộc tấn công như người dùng bình thường hoặc kẻ tấn công bên ngoài. Mục tiêu là đảm bảo rằng ứng dụng hoạt động đúng với các yêu cầu đã định và không có lỗ hổng bảo mật nào có thể bị khai thác bởi người dùng không có thông tin nội bộ.

##### **White Box Testing**

- Đây là phương pháp kiểm thử phần mềm khi người kiểm thử có các thông tin nội bộ của ứng dụng như mã nguồn, cấu trúc của ứng dụng và logic hoạt động. Người kiểm thử có thể xem xét và phân tích mã nguồn, thiết kế của hệ thống và các đường dẫn dữ liệu để xác định các lỗ hổng bảo mật và lỗi logic.

- Mục tiêu: Phát hiện các lỗ hổng bảo mật bên trong mã nguồn và logic hệ thống. Mục tiêu là đảm bảo rằng mã nguồn của ứng dụng không có lỗi logic hoặc lỗ hổng bảo mật nào có thể bị khai thác.

#### **Gray Box Testing**

- Kiểm thử hộp xám là phương pháp kiểm thử kết hợp cả hai phương pháp trên (Black Box Testing và White Box Testing). Người kiểm thử có một phần kiến thức về hệ thống nội bộ, bao gồm một số thông tin về mã nguồn, cấu trúc của ứng dụng và logic hệ thống. Điều này cho phép người kiểm thử sử dụng cả thông tin nội bộ và quan điểm của người dùng bên ngoài để phát hiện lỗ hổng bảo mật một cách hiệu quả.
- Mục tiêu: Sử dụng kiến thức về hệ thống để phát hiện lỗ hổng bảo mật một cách hiệu quả hơn so với chỉ sử dụng một phương pháp duy nhất. Mục tiêu là kết hợp các ưu điểm của cả Black Box Testing và White Box Testing để đảm bảo rằng ứng dụng không có lỗ hổng bảo mật nào có thể bị khai thác từ cả bên trong lẫn bên ngoài.

### **1.3.3 Các giai đoạn kiểm thử**

- Discovery:
  - + Giai đoạn này yêu cầu pentester thu thập dữ liệu cần thiết để hiểu các sự kiện dẫn đến việc khai thác thành công các ứng dụng di động. Tập hợp thông tin tình báo là giai đoạn chính trong một bài kiểm thử thâm nhập.
  - + Khả năng tiết lộ các manh mối bí mật có thể làm sáng tỏ một lỗ hổng bảo mật có thể là sự khác biệt giữa một cuộc kiểm thử thâm nhập thành công và thất bại.
- Assessment/Analysis: Giai đoạn này yêu cầu pentester kiểm tra mã nguồn của ứng dụng di động và xác định các điểm nhập và lỗ hổng tiềm ẩn có thể bị khai thác. Việc phân tích ứng dụng di động đặc biệt ở chỗ pentester phải đánh giá các ứng dụng cả trước và sau khi cài đặt.
- Exploitation:
  - + Giai đoạn này yêu cầu pentester thao tác các lỗ hổng đã được phát hiện để chiếm quyền điều khiển ứng dụng di động theo cách mà lập trình viên không mong đợi.
  - + Pentester cố gắng sử dụng lỗ hổng để đánh cắp dữ liệu hoặc thực hiện các hành động độc hại và sau đó thực hiện leo thang đặc quyền để trở thành người dùng có quyền cao nhất (root) và loại bỏ tất cả các giới hạn về các hoạt động có thể thực hiện.
- Reporting: Đây là giai đoạn cuối cùng của phương pháp, yêu cầu ghi lại và trình bày các vấn đề đã phát hiện theo cách dễ hiểu cho ban quản lý. Đây cũng là giai đoạn biến một cuộc kiểm thử thâm nhập từ một cuộc tấn công thành một báo cáo có giá trị.

## CHƯƠNG 2 XÂY DỰNG KỊCH BẢN

### 2.1 Kịch bản

- Trong báo cáo này, em tập trung vào việc pentest 5 lỗ hổng bảo mật của ứng dụng Android phổ biến và nguy hiểm nhất. Việc lựa chọn 5 lỗ hổng này dựa trên các yếu tố như mức độ phổ biến, mức độ nguy hiểm, và tính khả thi khi thực hiện pentest. Cụ thể, các lỗ hổng này không chỉ thường xuyên xuất hiện trong các ứng dụng di động mà còn có khả năng gây ra những hậu quả nghiêm trọng nếu không được phát hiện và khắc phục kịp thời. Bằng cách tập trung vào những lỗ hổng này, tôi mong muốn cung cấp một cái nhìn toàn diện và cụ thể về những rủi ro bảo mật trên Android, đồng thời đề xuất các biện pháp để bảo vệ ứng dụng khỏi các cuộc tấn công tiềm ẩn.

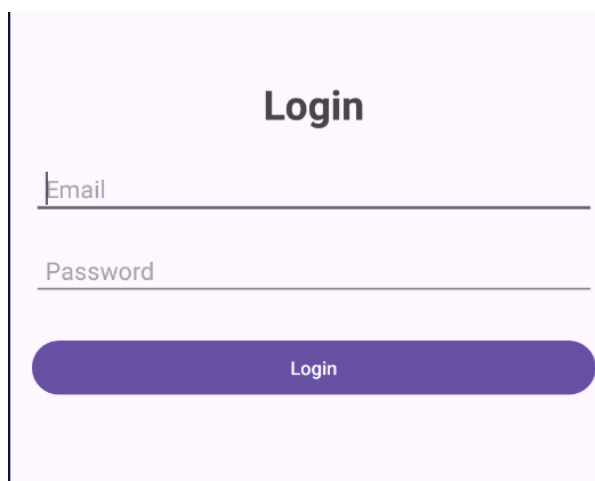
#### 2.1.1 Insecure Authentication/Authorization

##### Ý tưởng

- Một ứng dụng có chức năng đăng nhập và sử dụng cơ sở dữ liệu SQLite nhưng không có các cơ chế xác thực và chuẩn hoá đầu vào của người dùng dẫn đến tồn tại lỗ hổng sql injection cho ứng dụng đó

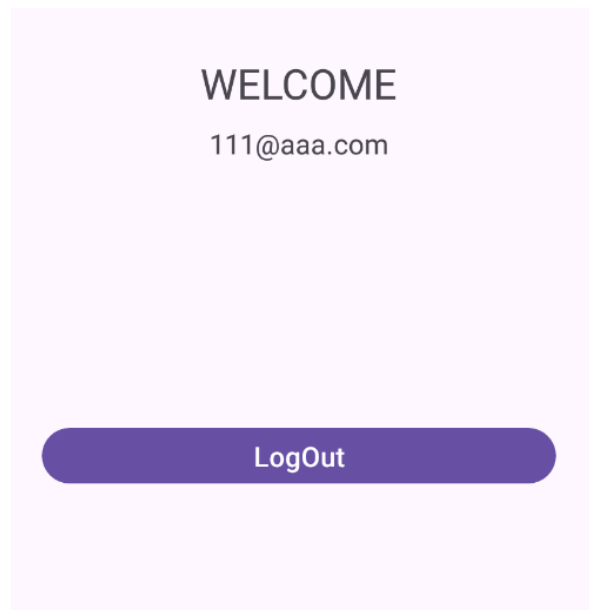
##### Mô tả ứng dụng

- Ứng dụng có 2 trang giao diện:
  - + Giao diện đăng nhập bao gồm 2 trường nhập thông tin đăng nhập username và password:

The image shows a login screen with a light purple background. At the top, the word "Login" is centered in a bold, dark font. Below it, there are two input fields: the first is labeled "Email" and the second is labeled "Password". Both fields have a thin purple border. At the bottom of the form, there is a rounded rectangular button with a solid purple background and the word "Login" in white text.

Hình 2.1 Giao diện đăng nhập của ứng dụng kịch bản 1

- + Giao diện sau khi đăng nhập bao gồm nút đăng xuất và hiển thị email đã đăng nhập:



Hình 2.2 Giao diện đăng nhập thành công của ứng dụng kịch bản 1

- Khi người dùng điền thông tin đăng nhập và ấn nút login ứng dụng sẽ tra cứu thông tin đăng nhập dựa trên cơ sở dữ liệu Authentication của firebase bằng hàm sẵn có của thư viện firebase

```
private void loginUser(String email, String password) {
    mAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(new OnCompleteListener<SignInResult>() {
            @Override
            public void onComplete(@NonNull Task<SignInResult> task) {
                // ...
            }
        });
}
```

Hình 2.3 Hàm đăng nhập của Firebase

### Kịch bản khai thác

- Điền thông tin username của tài khoản muốn tấn công và password ngẫu nhiên
- Sử dụng burp suit để bắt request yêu cầu xác thực của ứng dụng đến server firebase
- Tiến hành bruteforce mật khẩu

### Hướng khắc phục

- Đối với cơ sở dữ liệu là sql khi truy vấn không được sử dụng trực tiếp dữ liệu người dùng để query nên sử dụng các biện pháp như PreparedStatement hoặc rawquery để bảo vệ trước sql injection

## 2.1.2 Insufficient Input/Output Validation

### Ý tưởng

- Một ứng dụng có chức năng đăng nhập nhưng không có mã hoá mật khẩu và sử dụng chính sách mật khẩu yếu chỉ yêu cầu 3 kí tự. Kẻ tấn công sử dụng các công cụ bắt request và tiến hành bruteforce mật khẩu

### Mô tả ứng dụng

- Ứng dụng có 2 trang giao diện:
  - + Giao diện đăng nhập bao gồm 2 trường nhập thông tin đăng nhập username và password:

**Login**

Username

Password

Login

Hình 2.4 Giao diện đăng nhập của ứng dụng kịch bản 2

+ Giao diện sau khi đăng nhập bao gồm nút đăng xuất:

**Welcome!**

Logout

Hình 2.5 Giao diện đăng nhập thành công của ứng dụng kịch bản 2

- Khi người dùng điền thông tin đăng nhập và ấn nút login ứng dụng sẽ trực tiếp sử dụng thông tin đăng nhập đó để truy vấn cơ sở dữ liệu:

```
String username = editTextUsername.getText().toString();
String password = editTextPassword.getText().toString();
String query = "SELECT * FROM users WHERE username = '" + username + "' AND password = '" + password + "'";
Cursor cursor = database.rawQuery(query, selectionArgs: null);
```

Hình 2.6 Đoạn mã chứa lỗ hổng bảo mật sql injection

### Kịch bản khai thác

- Điền thông tin username với cấu trúc hợp username của tài khoản muốn đăng nhập và mã sql injection
- Điền thông tin password ngẫu nhiên
- Tiến hành ấn nút đăng nhập và kiểm tra kết quả

### Hướng khắc phục

- Đối với cơ sở dữ liệu là sql khi truy vấn không được sử dụng trực tiếp dữ liệu người dùng để query nên sử dụng các biện pháp như PreparedStatement hoặc rawquery để bảo vệ trước sql injection

## 2.1.3 Insecure Communication

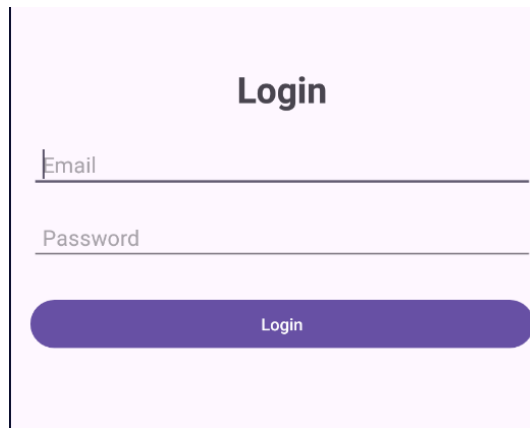
### Ý tưởng

- Một ứng dụng bán hàng có chức năng đăng nhập thông qua xác thực tài khoản dựa trên cơ sở dữ liệu được lưu trên Firebase nhưng không sử dụng các biện pháp mã hoá hoặc băm mật khẩu. Kẻ tấn công sử dụng kỹ thuật tấn công Man-in-the-middle để nghe lén thông tin đăng nhập của người dùng

### Mô tả ứng dụng

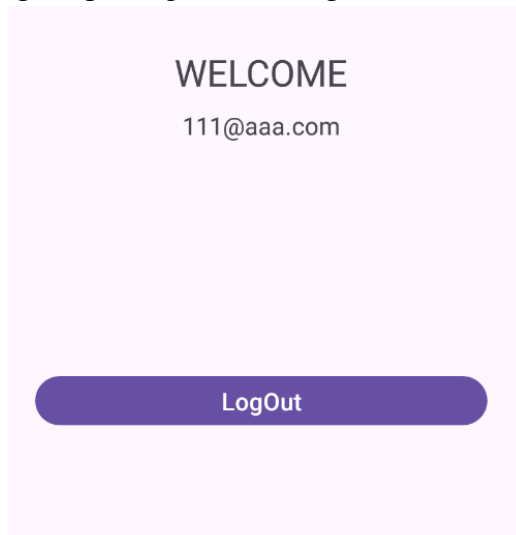
- Ứng dụng có 2 trang giao diện:

+ Giao diện đăng nhập bao gồm 2 trường nhập thông tin đăng nhập username và password:

A login form with a light purple background. At the top, the word "Login" is centered in a bold, dark font. Below it, there are two input fields: the first is labeled "Email" and the second is labeled "Password". Both labels are in a light purple font. Below the input fields, there is a rounded rectangular button with a dark purple background and the word "Login" in white text.

Hình 2.7 Giao diện đăng nhập của ứng dụng kịch bản 3

+ Giao diện sau khi đăng nhập bao gồm nút đăng xuất và hiển thị email đã đăng nhập:

A welcome screen with a light purple background. At the top, the word "WELCOME" is centered in a bold, dark font. Below it, the email address "111@aaa.com" is displayed in a smaller, dark font. At the bottom, there is a rounded rectangular button with a dark purple background and the text "LogOut" in white text.

Hình 2.8 Giao diện đăng nhập thành công của ứng dụng kịch bản 3

- Khi người dùng điền thông tin đăng nhập và ấn nút login ứng dụng sẽ tra cứu thông tin đăng nhập dựa trên cơ sở dữ liệu Authentication của firebase bằng hàm sẵn có của thư viện firebase

```
private void loginUser(String email, String password) {  
    mAuth.signInWithEmailAndPassword(email, password)  
    addOnCompleteListener(activity, this, new OnC
```

Hình 2.9 Hàm đăng nhập của Firebase

### Kịch bản khai thác

- Điền thông tin username và mật khẩu đăng nhập đúng
- Sử dụng burp suit để bắt request từ ứng dụng đến firebase server
- Kiểm tra thông tin của request

### Hướng khắc phục

- Đối với các thông tin nhạy cảm như mật khẩu đăng nhập nên được mã hoá hoặc băm với các thuật toán phức tạp ngay trên ứng dụng trước khi được truyền đi đến server để đảm bảo an toàn trên đường truyền.

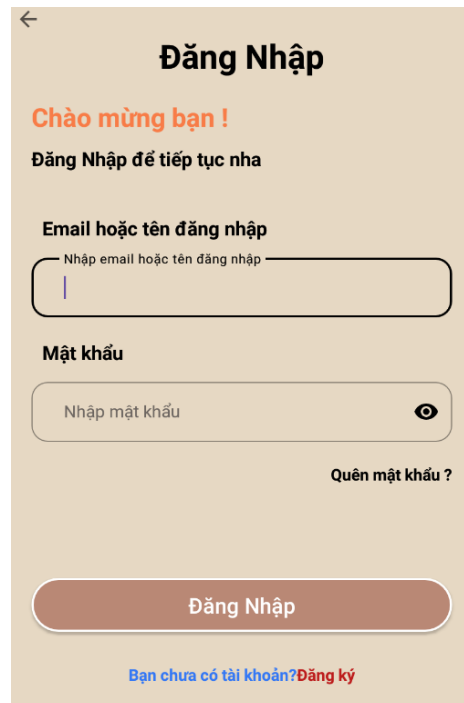
#### 2.1.4 Insufficient Binary Protection

##### Ý tưởng

- Một ứng dụng bán caffe thiếu các biện pháp bảo mật chống dịch ngược. Kẻ tấn công dùng công cụ như jadx để dịch ngược ứng dụng xác định có lỗ hổng tại hàm thực hiện chức năng xác thực OTP khi lựa chọn quên mật khẩu, từ đó kẻ tấn công sử dụng công cụ chỉnh sửa file apk của ứng dụng cho phép kẻ tấn công kiểm soát mã OTP từ đó có thể cài lại mật khẩu của bất kì user nào chỉ bằng email hoặc số điện thoại và chiếm tài khoản của user đó

##### Mô tả ứng dụng

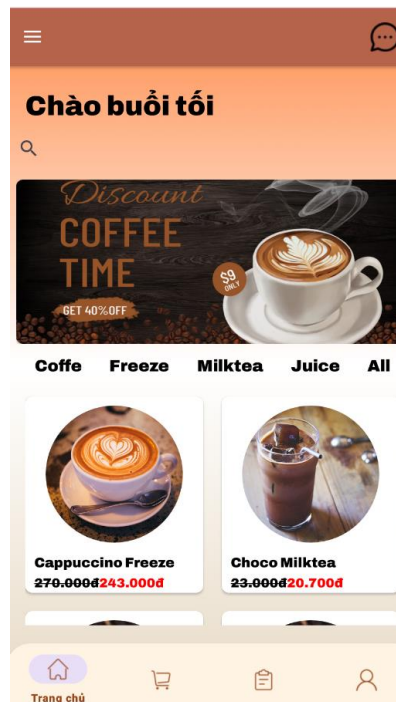
- Ứng dụng có các trang giao diện chính:
  - + Giao diện đăng nhập bao gồm 2 trường nhập thông tin đăng nhập username và password:



Hình 2.10 Giao diện đăng nhập của ứng dụng kịch bản 4

- + Giao diện sau khi đăng nhập thành công:





Hình 2.11 Giao diện đăng nhập thành công của ứng dụng kịch bản 4

+ Giao diện quên mật khẩu và OTP:



## Quên Mật Khẩu

Nhập email hoặc SĐT

Gửi OTP

Hình 2.12 Giao diện quên mật khẩu của ứng dụng kịch bản 4

**Xác Minh****Nhập mã xác nhận**

Thời gian còn lại: 118 giây

Không nhận được mã **Gửi lại****Xác nhận***Hình 2.13 Giao diện xác minh OTP của ứng dụng kịch bản 4***Kịch bản khai thác**

- Sử dụng công cụ jadx – gui để kiểm tra lỗ hổng OTP
- Sử dụng công cụ MTManager để chỉnh sửa chức năng xác thực OTP trong file apk
- Thử nghiệm sử dụng email đã đăng kí trong ứng dụng và chức năng quên mật khẩu để cài lại mật khẩu
- Đăng nhập vào tài khoản của email

**Hướng khắc phục**

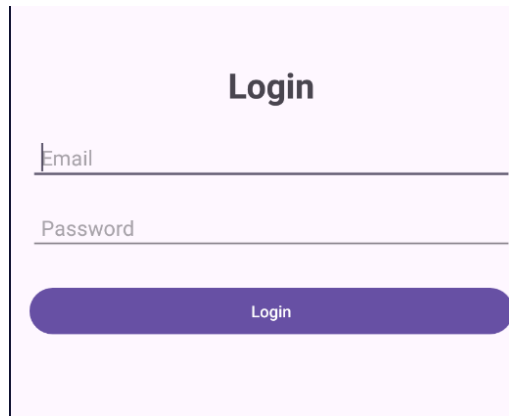
- Nên sử dụng chức năng obfuscate trong android studio trong quá trình xuất file apk để làm rối mã bảo vệ trước tấn công reverse engineering

**2.1.5 Insufficient Cryptography****Ý tưởng**

- Một ứng dụng có chức năng đăng nhập có sử dụng băm mật khẩu để bảo vệ nhưng sử dụng thuật toán băm lỗi thời, yếu có thể bị tấn công hash collision. Kẻ tấn công sử dụng phương pháp tấn công Man-in-the-middle để tiến hành nghe lén request bắt lấy mật khẩu được băm bằng md5 sau đó tiến hành giải mã và đăng nhập vào tài khoản.

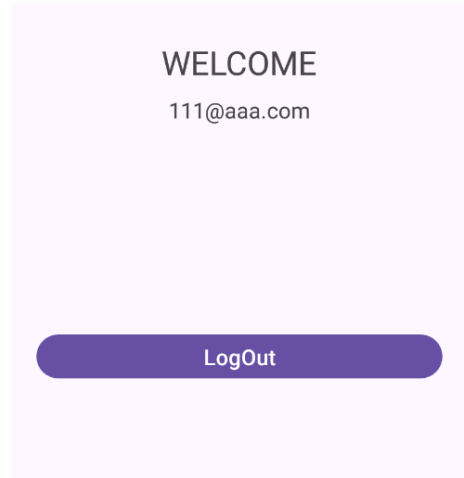
**Mô tả ứng dụng**

- Ứng dụng có 2 trang giao diện:
  - + Giao diện đăng nhập bao gồm 2 trường nhập thông tin đăng nhập username và password:

A login form with a light purple background. At the top, the word "Login" is centered in bold. Below it are two input fields: "Email" and "Password", each with a horizontal line for text entry. At the bottom, there is a purple rounded button with the text "Login" in white.

Hình 2.14 Giao diện đăng nhập của ứng dụng kịch bản 5

+ Giao diện sau khi đăng nhập bao gồm nút đăng xuất và hiển thị email đã đăng nhập:

A welcome screen with a light purple background. At the top, the word "WELCOME" is centered in bold. Below it, the email address "111@aaa.com" is displayed. At the bottom, there is a purple rounded button with the text "LogOut" in white.

Hình 2.15 Giao diện đăng nhập thành công của ứng dụng kịch bản 5

- Khi người dùng điền thông tin đăng nhập và ấn nút login ứng dụng sẽ tra cứu thông tin đăng nhập sẽ được băm bằng thuật toán băm md5 trước khi xác thực trên firebase

#### **Kịch bản khai thác**

- Điền thông tin username và mật khẩu đăng nhập đúng
- Sử dụng burp suit để bắt request từ ứng dụng đến firebase server
- Giải mã hash bằng hash collision
- Kiểm tra mật khẩu được giải có chính xác

#### **Hướng khắc phục**

- Đối với các thông tin nhạy cảm như mật khẩu đăng nhập nên được mã hoá hoặc băm với các thuật toán phức tạp ngay trên ứng dụng trước khi được truyền đi đến server để đảm bảo an toàn trên đường truyền.

## 2.2 Công cụ

### 2.2.1 Burp suite



*Hình 2.16 Logo Burpsuit*

- Burp Suite là một công cụ phổ biến và mạnh mẽ được sử dụng trong lĩnh vực bảo mật thông tin để kiểm thử xâm nhập ứng dụng web và di động. Nó được phát triển bởi công ty PortSwigger và cung cấp một loạt các tính năng hỗ trợ kiểm thử bảo mật.
- Burp Suite bao gồm nhiều công cụ khác nhau, giúp kiểm tra và phân tích bảo mật của ứng dụng. Một số công cụ tiêu biểu trong Burp Suite bao gồm:
  - + Burp Proxy: Là công cụ chính, hoạt động như một proxy giữa trình duyệt của người dùng và ứng dụng web. Nó cho phép người kiểm thử chặn, sửa đổi và phân tích các yêu cầu và phản hồi HTTP/HTTPS.
  - + Burp Intruder: Dùng để thực hiện các cuộc tấn công brute-force hoặc fuzzing, nhằm phát hiện các lỗ hổng bảo mật liên quan đến đầu vào của người dùng.
  - + Burp Repeater: Cho phép người kiểm thử gửi các yêu cầu HTTP/HTTPS lặp đi lặp lại với các tham số tùy chỉnh để phân tích phản hồi của máy chủ.
  - + Burp Decoder: Dùng để mã hoá hoặc giải mã dữ liệu, hỗ trợ trong việc phân tích các mã hoá như Base64, mã hoá URL, v.v.
  - + Burp Comparer: So sánh hai đoạn dữ liệu để tìm ra sự khác biệt, hữu ích trong việc phân tích sự thay đổi giữa các phản hồi hoặc yêu cầu.

### 2.2.2 JADX



*Hình 2.17 Logo JADX-GUI*

- JADX là một công cụ mã mạnh mẽ nguồn mở giúp dịch ngược các tập tin APK của Android và DEX thành mã nguồn Java. Công cụ này rất hữu ích trong việc phân tích và kiểm tra bảo mật ứng dụng Android, giúp các chuyên gia bảo mật và nhà phát triển hiểu rõ hơn về cấu trúc và logic của ứng dụng.
- Một số chức năng tiêu biểu của Jadx
  - + Dịch ngược mã nguồn Java: Giúp người dùng dễ dàng đọc và hiểu mã nguồn gốc của ứng dụng Android.

- + Hỗ trợ đa nền tảng: Chạy trên nhiều hệ điều hành như Windows, macOS, và Linux.
- + Tìm kiếm mã nguồn: Cho phép tìm kiếm và điều hướng qua mã nguồn dịch ngược để phân tích các phương thức và lớp cụ thể.
- + Xem cấu trúc tệp: Hiển thị cấu trúc tệp của APK hoặc DEX, giúp người dùng dễ dàng điều hướng qua các lớp, phương thức, và thuộc tính.

### 2.2.3 MTManager



*Hình 2.18 Logo MT manager*

- MT Manager là một ứng dụng quản lý tệp mạnh mẽ dành cho hệ điều hành Android. Nó không chỉ đơn giản là một trình quản lý tệp thông thường mà còn tích hợp nhiều tính năng nâng cao, bao gồm khả năng chỉnh sửa tệp APK và DEX, dịch ngược, và chỉnh sửa mã nguồn. Điều này làm cho MT Manager trở thành một công cụ hữu ích cho cả người dùng thông thường lẫn các nhà phát triển và chuyên gia bảo mật.
- Một số tính năng tiêu biểu:
- + Quản lý tệp: MT Manager cung cấp các chức năng cơ bản của một trình quản lý tệp, cho phép người dùng sao chép, di chuyển, xóa, đổi tên, và chia sẻ các tệp và thư mục trên thiết bị Android của họ.
  - + Chỉnh sửa tệp APK và DEX: MT Manager có khả năng chỉnh sửa các tệp APK và DEX, bao gồm việc dịch ngược (decompile) và tái biên dịch (recompile) các tệp này. Điều này cho phép người dùng chỉnh sửa mã nguồn của ứng dụng Android trực tiếp trên thiết bị của họ.
  - + Chỉnh sửa mã nguồn: Tích hợp trình chỉnh sửa mã nguồn, giúp người dùng dễ dàng chỉnh sửa mã nguồn Java hoặc Smali sau khi dịch ngược từ các tệp APK hoặc DEX.
  - + Hex Editor: MT Manager cung cấp một trình chỉnh sửa hex, cho phép người dùng chỉnh sửa các tệp ở cấp độ byte, rất hữu ích cho việc chỉnh sửa tệp nhị phân hoặc thực hiện các thao tác chỉnh sửa nâng cao khác.
  - + Duyệt root: Đối với các thiết bị đã root, MT Manager cho phép truy cập vào các thư mục hệ thống và thực hiện các thao tác yêu cầu quyền truy cập root.

## 2.2.4 NoxPlayer

*Hình 2.19 Logo NOX*

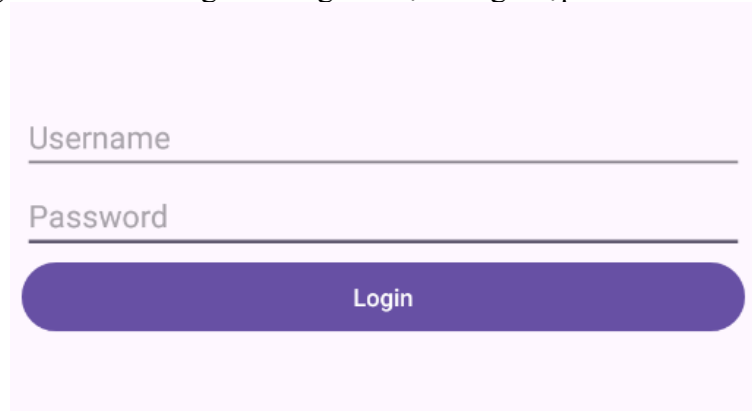
- NoxPlayer là một trình giả lập Android mạnh mẽ dành cho máy tính chạy hệ điều hành Windows và macOS. Nó cung cấp một môi trường Android ảo trên máy tính, cho phép người dùng cài đặt và sử dụng các ứng dụng và trò chơi Android như trên thiết bị thật. NoxPlayer được nhiều người sử dụng vì hiệu suất cao, tính ổn định, và tính năng phong phú
- Một số chức năng chính:
  - + Môi trường thử nghiệm an toàn: NoxPlayer cho phép tạo ra một môi trường thử nghiệm an toàn và cô lập để kiểm tra các ứng dụng Android mà không ảnh hưởng đến thiết bị thật
  - + Giả lập nhiều thiết bị và cấu hình: NoxPlayer cho phép giả lập nhiều thiết bị với các cấu hình khác nhau để kiểm tra xem ứng dụng có hoạt động và bảo mật tốt trên tất cả các loại thiết bị hay không.

## CHƯƠNG 3 THỰC NGHIỆM

### 3.1 Insecure Authentication/Authorization

#### Kiểm thử

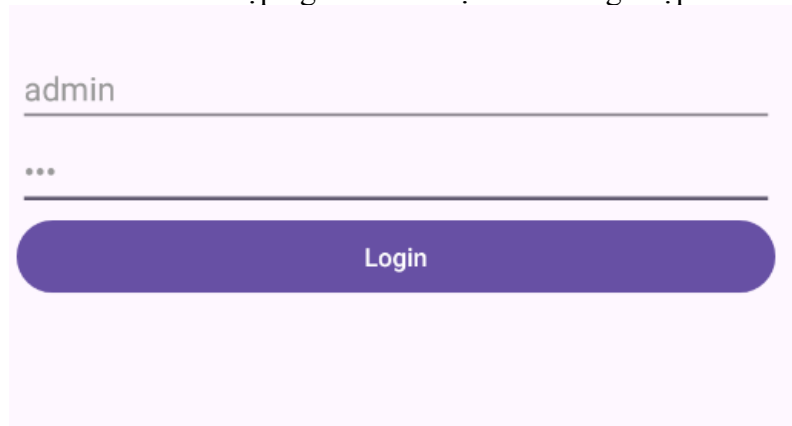
- Mở ứng dụng có chứa lỗ hổng và vào giao diện đăng nhập



A screenshot of a web application's login interface. It features two input fields: 'Username' and 'Password', both with placeholder text. Below the fields is a prominent blue 'Login' button.

Hình 3.1 Giao diện đăng nhập

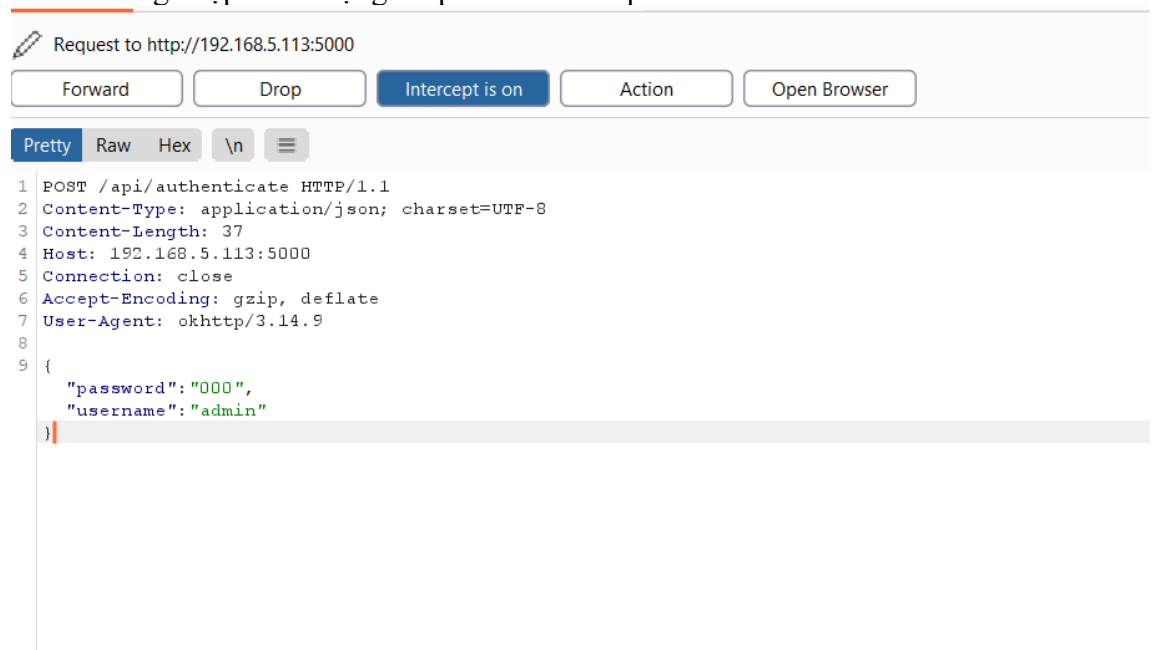
- Nhập username là admin và nhập ngẫu nhiên mật khẩu đăng nhập



The same login interface as in Hình 3.1, but now the 'Username' field contains the text 'admin' and the 'Password' field contains a series of dots, representing a randomly generated password.

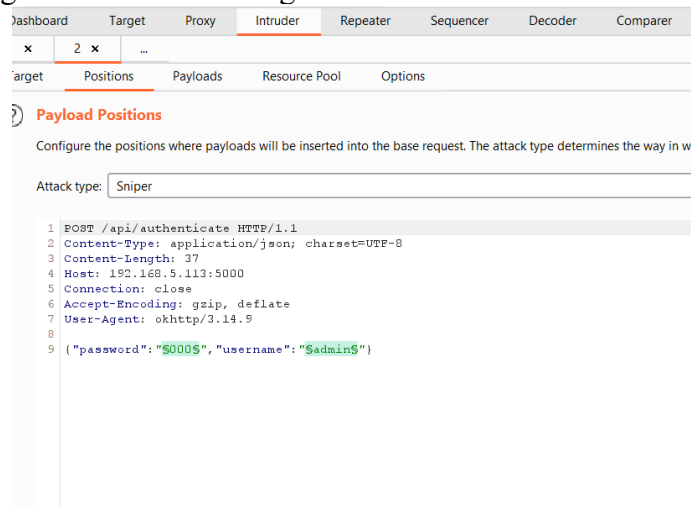
Hình 3.2 Điền thông tin đăng nhập

- Ấn nút đăng nhập và sử dụng burpsuit để bắt request



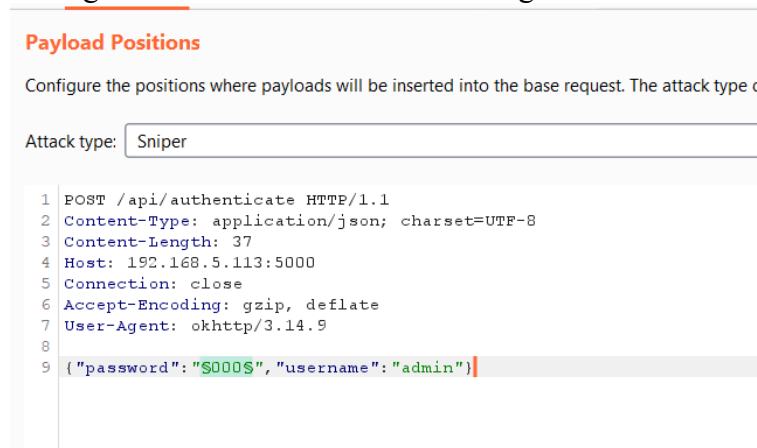
Hình 3.3 Bắt request bằng intercept

- Đưa request sang Intruder để tấn công bruteforce

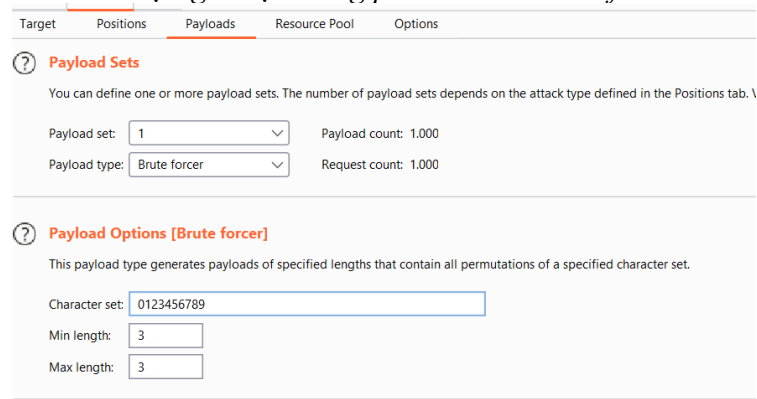


Hình 3.4 Đưa request sang Intruder

- Điều chỉnh các thông số cần thiết và tiến hành tấn công



Hình 3.5 Chọn giá trị trường password để bruteforce



Hình 3.6 Điều chỉnh thông số tấn công

- Ta đã có được mật khẩu



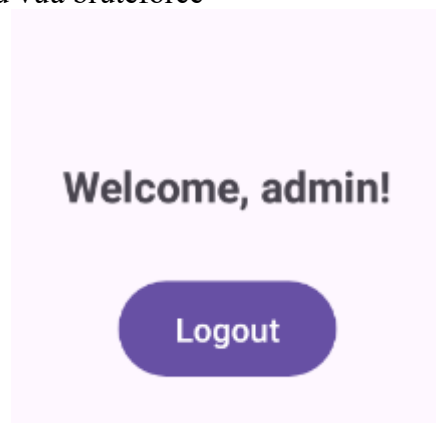
Request	Payload	Status ^	Error	Timeout	Length
22	123	200	<input type="checkbox"/>	<input type="checkbox"/>	216
0		401	<input type="checkbox"/>	<input type="checkbox"/>	209
1	111	401	<input type="checkbox"/>	<input type="checkbox"/>	209
2	211	401	<input type="checkbox"/>	<input type="checkbox"/>	209
3	311	401	<input type="checkbox"/>	<input type="checkbox"/>	209
4	121	401	<input type="checkbox"/>	<input type="checkbox"/>	209
5	221	401	<input type="checkbox"/>	<input type="checkbox"/>	209
6	321	401	<input type="checkbox"/>	<input type="checkbox"/>	209
7	131	401	<input type="checkbox"/>	<input type="checkbox"/>	209
8	231	401	<input type="checkbox"/>	<input type="checkbox"/>	209
9	331	401	<input type="checkbox"/>	<input type="checkbox"/>	209
10	112	401	<input type="checkbox"/>	<input type="checkbox"/>	209

Request	Response
<pre> 1 POST /api/authenticate HTTP/1.1 2 Content-Type: application/json; charset=UTF-8 3 Content-Length: 37 4 Host: 192.168.5.113:5000 5 Connection: close 6 Accept-Encoding: gzip, deflate 7 User-Agent: okhttp/3.14.9 8 9 { 10   "password": "123", 11   "username": "admin" 12 } </pre>	

Hình 3.7 Kết quả bruteforce

- Đăng nhập bằng mật khẩu vừa bruteforce



Hình 3.8 Đăng nhập thành công

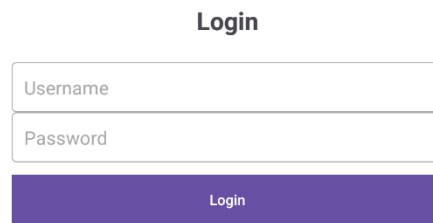
### Khắc phục

- Sử dụng các chính sách mật khẩu mạnh như yêu cầu mật khẩu dài hơn, phức tạp hơn để tránh bị tấn công bruteforce.

## 3.2 Insufficient Input/Output Validation

### Kiểm thử

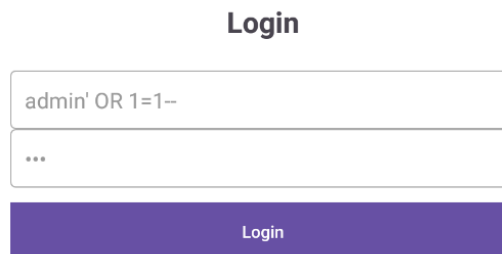
- Mở ứng dụng chứa lỗ hổng và vào giao diện đăng nhập:



The screenshot shows a login form with the title "Login". It contains two input fields: "Username" and "Password". Below these fields is a purple button labeled "Login".

Hình 3.9 Giao diện đăng nhập

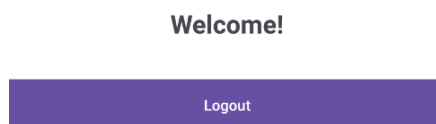
- Tiến hành điền thông tin đăng nhập như sau:
  - + Username: admin' OR 1=1—
  - + Password: 123



The screenshot shows the login form with the title "Login". The "Username" field contains the text "admin' OR 1=1—" and the "Password" field contains "123". The "Login" button is visible at the bottom.

Hình 3.10 Tấn công SQL Injection

- Sau khi nhấn nút đăng nhập ta đã đăng nhập thành công



The screenshot shows a page with the title "Welcome!". Below the title is a purple button labeled "Logout".

Hình 3.11 Tấn công thành công

### Khắc phục

- Sử dụng Raw query để tránh tấn công sql injection

```
private Cursor safe(){ no usages
    String username = editTextUsername.getText().toString();
    String password = editTextPassword.getText().toString();

    String query = "SELECT * FROM users WHERE username = ? AND password = ?";
    return database.rawQuery(query, new String[]{username, password});
}
```

Hình 3.12 Mã sửa lỗi sql injection

- Lúc này input của người dùng sẽ được chuyển sang dạng string trước khi được đưa vào trong query
- Thử nghiệm cho ứng dụng sau khi đã được khắc phục

**Login**

Login

Invalid login

Hình 3.13 Lỗi SQL Injection đã được khắc phục

### 3.3 Insecure Communication

#### Kiểm thử

- Mở ứng dụng và vào phần đăng nhập

**Login**

Email

Password

Login

Hình 3.14 Giao diện đăng nhập

- Điền thông tin đăng nhập và ấn login

**Login**

111@aaa.com

\*\*\*\*\*

Login

Hình 3.15 Điền thông tin đăng nhập

- Sử dụng burpsuit để giả lập tấn công Man-in-the-middle

```

POST /identitytoolkit/v3/relyingparty/verifyPassword?key=AIzaSyBiParLXKh-ejzC 7
Host: www.googleapis.com 8
Content-Type: application/json 9
X-Android-Package: com.example.apptest1 10
X-Android-Cert: 53A454D0D5D64EA1548ABE2BE74B762058F43A91 11
Accept-Language: en-US 12
X-Client-Version: Android/Fallback/X230000000/FirebaseCore-Android 13
X-Firebase-Gmpid: 1:507341880522;android:1c576820e5dd062b110833 14
X-Firebase-Client: H4sIAAAAAAAAAAKtWykhNLCpJSk0sKVayio7VUSpLLSrOzM9TslIyUqoFAF 15
Content-Length: 103 16
User-Agent: Dalvik/2.1.0 (Linux; U; Android 7.1.2; SM-N976N Build/QP1A.190711 17
Connection: Keep-Alive 18
Accept-Encoding: gzip, deflate 19
{ 20
  { 21
    "email": "111@aaa.com", 22
    "password": "123455",
    "returnSecureToken": true, 23
    "clientType": "CLIENT_TYPE_ANDROID" 24
  } 25
} 26
27

```

Hình 3.16 Thông tin đăng nhập trong request

- Ta đã bắt được thông tin đăng nhập thành công

### Khắc phục

- Ta có thể sử dụng các loại mã hoá hoặc các hàm băm để mã hoá thông tin đăng nhập tránh bị đánh cắp thông tin

```

package com.example.apptest1;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class HashUtil { 2 usages

    public static String getSHA256Hash(String input) { 2 usages
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");

            byte[] hashBytes = digest.digest(input.getBytes());

            StringBuilder hexString = new StringBuilder();
            for (byte b : hashBytes) {
                String hex = Integer.toHexString(0xff & b);
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);
            }
            return hexString.toString();
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
}

```

Hình 3.17 Hàm băm sha256

```
POST /identitytoolkit/v3/relyingparty/verifyPassword?key=AIzaSyBiParLXKh-ejzc
Host: www.googleapis.com
Content-Type: application/json
X-Android-Package: com.example.apptest1
X-Android-Cert: 53A454D0D5D64EA1548ABE2BE74B762058F43A91
Accept-Language: en-US
X-Client-Version: Android/Fallback/X23000000/FirebaseCore-Android
X-Firebase-Gmpid: 1:507341880522:android:1c576820e5dd062b110833
X-Firebase-Client: H4sIAAAAAAAAAAKtWykhNLCpJShK0sKVayio7VUSpLLSrOzM9TslIyUqoFAF
Content-Length: 161
User-Agent: Dalvik/2.1.0 (Linux; U; Android 7.1.2; SM-N976N Build/QP1A.190711
Connection: Keep-Alive
Accept-Encoding: gzip, deflate

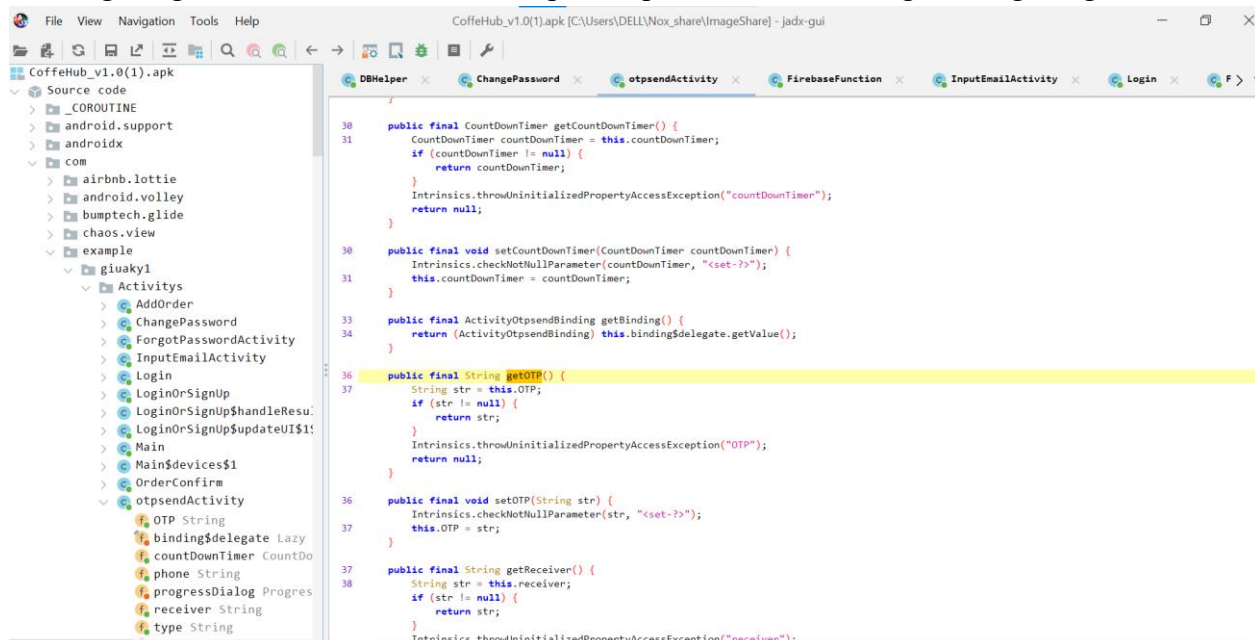
{
  "email": "111@aaa.com",
  "password": "fc1f09ab08ebdd072ea6da53a5691abcc18c9163b1be1f0921a5adb50e3f507",
  "returnSecureToken": true,
  "clientType": "CLIENT_TYPE_ANDROID"
}
```

Hình 3.18 Mật khẩu đã được bẻ

### 3.4 Insufficient Binary Protection

#### Kiểm thử

- Sử dụng công cụ Jadx để tiến hành decompile và phân tích tìm lỗ hổng của ứng dụng



Hình 3.19 Giao diện JADX - GUI

- Ở đây ta tìm thấy hàm kiểm tra mã OPT có cơ chế xác thực đơn giản bằng cách gọi hàm getOTP nếu có mã sẽ trả về 1 chuỗi OPT vậy ta có thể khiến hàm này luôn trả về 1 chuỗi mà ta muốn điều này giúp ta có thể bypass chức năng xác thực OPT

```
private final void oTPProcessing() {
    this.progressDialog = ProgressDialog.show(this, "App", "Loading...", true);
    String otp = String.valueOf(getBinding().pinview.getText());
    if (TextUtils.isEmpty(otp) || otp.length() < 6) {
        getBinding().pinview.setError(getString(R.string.b_n_ch_a_nh_p_m_otp));
    } else if (getType().equals("mail")) {
        System.out.println((Object) ("otp send : " + getOTP()));
        System.out.println((Object) ("otp nhập vào : " + getOTP()));
    } if (otp.equals(getOTP())) {
        ProgressDialog progressDialog = this.progressDialog;
        if (progressDialog != null) {
            progressDialog.dismiss();
        }
        startActivity(new Intent(this, ForgotPasswordActivity.class).putExtra("receiver", getReceiver()));
        return;
    }
    ProgressDialog progressDialog2 = this.progressDialog;
    if (progressDialog2 != null) {
        progressDialog2.dismiss();
    }
    getBinding().pinview.setText("");
    Toast.makeText(getApplicationContext(), getString(R.string.otp_kh_ng_ch_nh_x_c), 0).show();
} else {
    OTP Athen Phone.Companion.OTPAuthenAndRegister(getOTP(), otp, this, new Function1<Boolean, Unit>() {
        OTP Athen Phone.Companion.OTPAuthenAndRegister(getOTP(), otp, this, new Function1<Boolean, Unit>() {

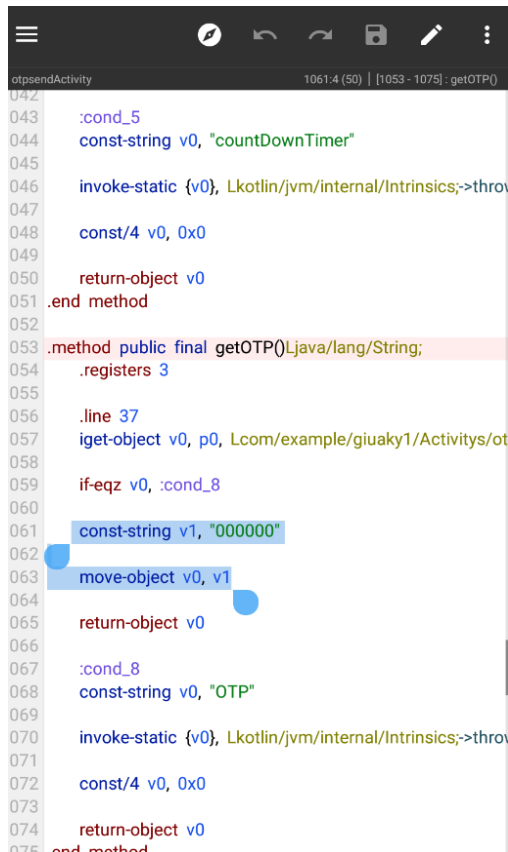
```

Hình 3.20 Kiểm tra OTP bằng hàm equal giữa thông tin nhập vào và hàm getOTP()

```
public final String getOTP() {
    String str = this.OTP;
    if (str != null) {
        return str;
    }
    Intrinsics.throwUninitializedPropertyAccessException("OTP");
    return null;
}
```

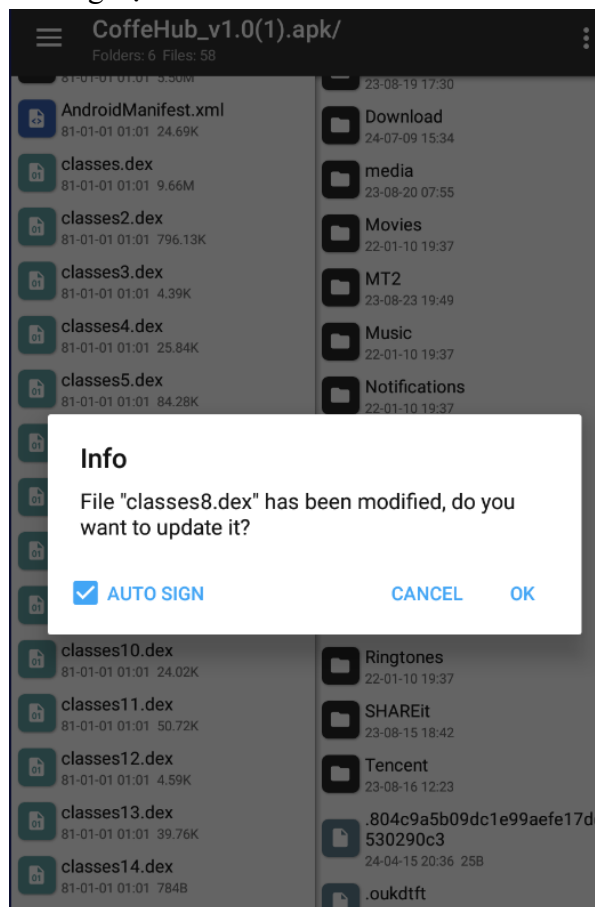
Hình 3.21 Thông tin hàm getOTP()

- Sau khi tìm được lỗi hỏng ta sử dụng ứng dụng MT manager để xem và chỉnh sửa file apk

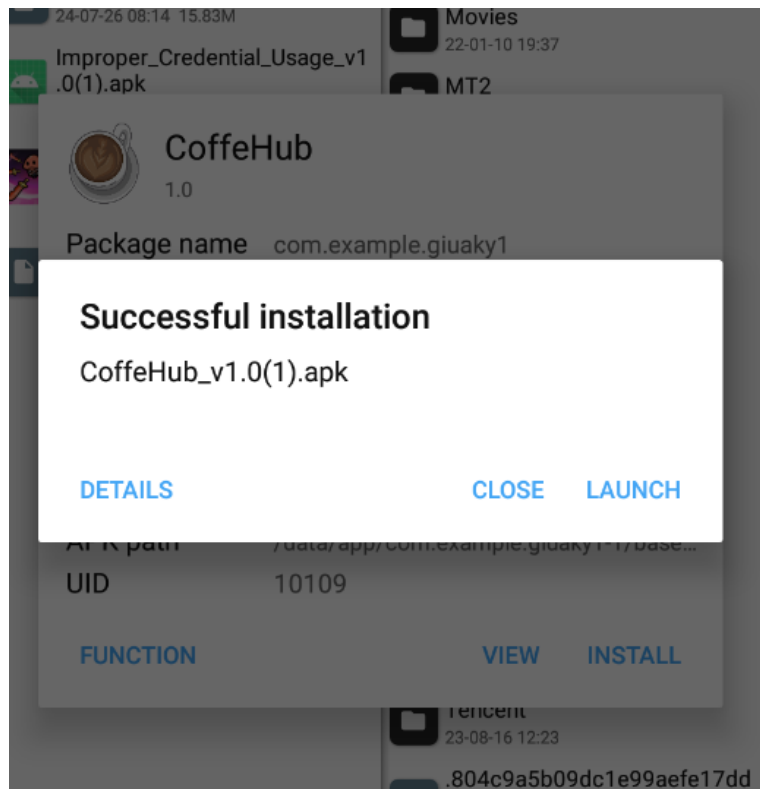


Hình 3.22 Sử dụng MT manager để chỉnh file .dex

- Ký apk lại và cài đặt thử nghiệm



Hình 3.23 Ký file sau khi đã sửa



Hình 3.24 Cài đặt file apk đã chỉnh sửa

- Điền thông tin email hoặc số điện thoại của mục tiêu trong phần quên mật khẩu



## Quên Mật Khẩu

Nhập email hoặc SĐT

Gửi OTP

Hình 3.25 Điền email victim

- Sau khi nhấn nút gửi OPT ta chỉ cần điền string mà ta đã cài đặt





## Xác Minh

### Nhập mã xác nhận

Thời gian còn lại: 114 giây

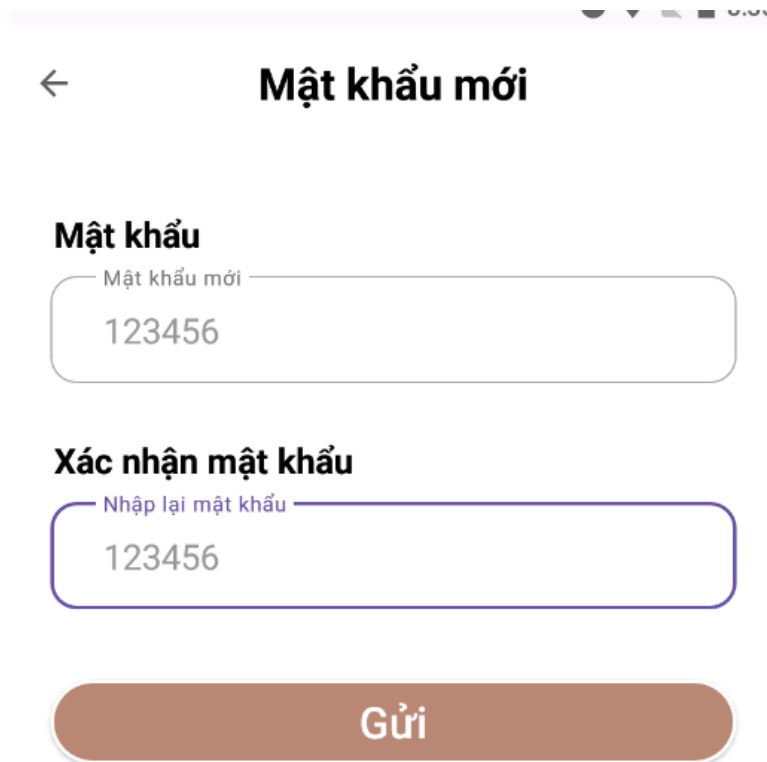
0	0	0	0	0	0
---	---	---	---	---	---

Không nhận được mã **Gửi lại**

Xác nhận

*Hình 3.26 Nhập mã OTP mà ta đã gán vào*

- Ta đã có thể cài đặt lại mật khẩu của mục tiêu



← **Mật khẩu mới**

**Mật khẩu**

Mật khẩu mới

123456

**Xác nhận mật khẩu**

Nhập lại mật khẩu

123456

**Gửi**

*Hình 3.27 Bypass thành công xác thực OTP*

- Đăng nhập vào tài khoản bằng mật khẩu mới



Hình 3.28 Đăng nhập thành công vào tài khoản victim

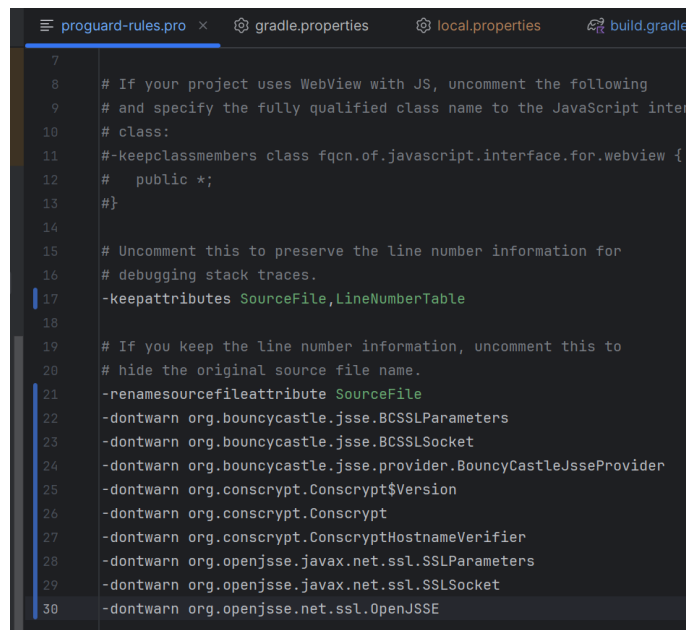
### Khắc phục

- Ở Android studio có cung cấp cho ta cơ chế obfuscate giúp làm rối mã khi decompile từ đó làm giảm khả năng tấn công
- + Enable chức năng Proguard bằng cách set `isMinifyEnabled = True`

```
buildTypes {
    release {
        isMinifyEnabled = true
        isShrinkResources = true
        proguardFiles(
            getDefaultProguardFile( name: "proguard-android-optimize.txt"),
            "proguard-rules.pro"
        )
    }
}
```

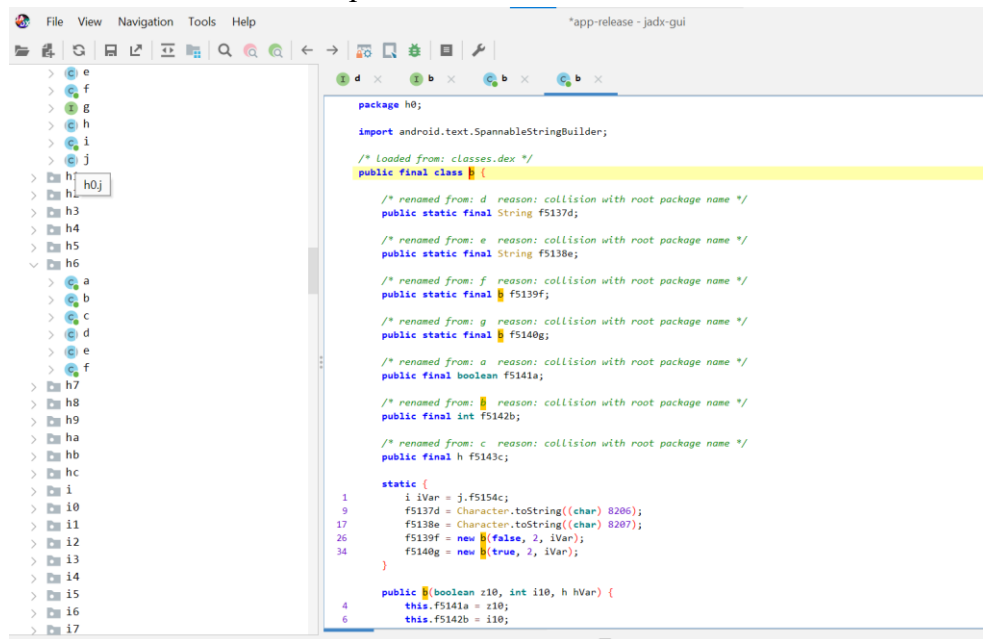
Hình 3.29 Kích hoạt Minify trong file build.gradle.kts

- + Cấu hình rule cho Proguard



Hình 3.30 Rule cấu hình Proguard

+ Sau khi build ta sẽ được file apk đã được obfuscate

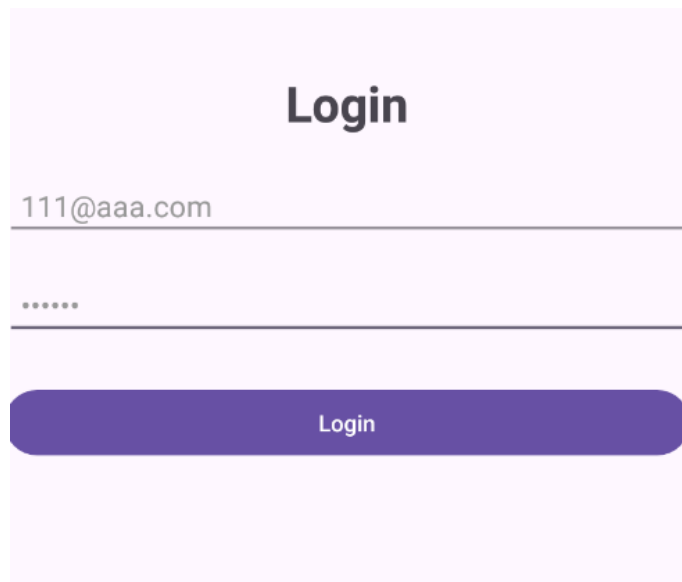


Hình 3.31 File decompile sau khi obfuscate

### 3.5 Insufficient Cryptography

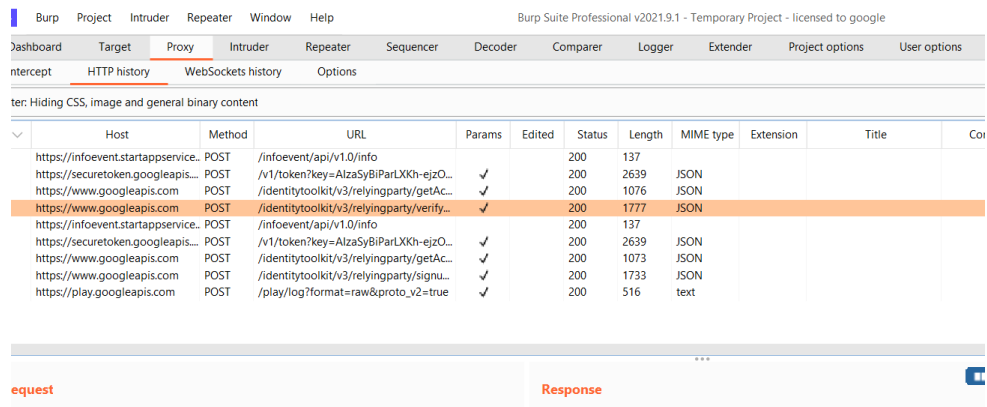
#### Kiểm thử

- Mở ứng dụng và vào giao diện đăng nhập và điền thông tin đăng nhập



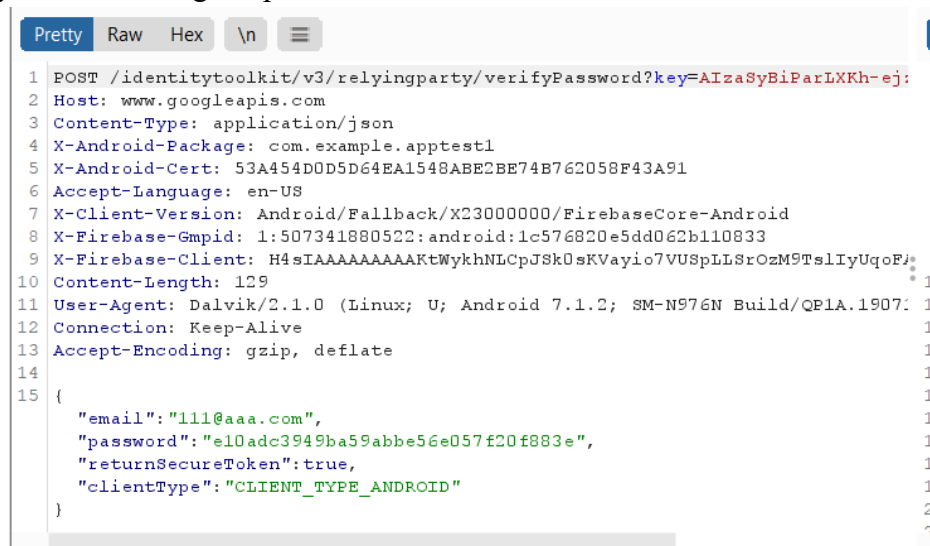
Hình 3.32 Trang đăng nhập

- Sử dụng burpsuit để tiến hành mô phỏng tấn công Man-in-the-middle



Hình 3.33 Giao diện burpsuit bắt các request

- Ta thấy mật khẩu đăng nhập đã bị bẫy



Hình 3.34 Thông tin mật khẩu đã được bẫy

- Decompile apk của ứng dụng ta thấy ứng dụng sử dụng md5 để băm mật khẩu

```
// renamed from: lambda$onCreate$0$com.example.apptest1Login reason: not valid java name :/
public /* synthetic */ void m84lambda$onCreate$0$com.example.apptest1Login(View view) {
    String email = this.binding.email.getText().toString();
    String password = this.binding.password.getText().toString();
    if (email.isEmpty() || password.isEmpty()) {
        Toast.makeText(this, "Email and password must not be empty", 0).show();
    } else {
        loginUser(email, HashUtil.generateMD5(password));
    }
}
```

Hình 3.35 Băm mật khẩu bằng MD5

```
public static String generateMD5(String input) {
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(input.getBytes());
        byte[] digest = md.digest();
        StringBuilder sb = new StringBuilder();
        for (byte b : digest) {
            sb.append(String.format("%02x", Byte.valueOf(b)));
        }
        return sb.toString();
    } catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}
```

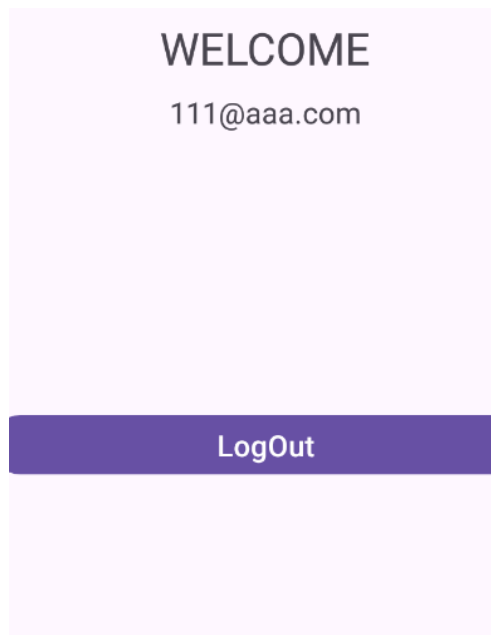
Hình 3.36 Hàm băm MD5

- Sử dụng các công cụ hoặc các website có sẵn để tiến hành băm khoá mật khẩu hash bằng hash collision



Hình 3.37 Sử dụng công cụ 10015 để tìm hash collision

- Ta đã có mật khẩu và đăng nhập thành công



*Hình 3.38 Đăng nhập vào tài khoản victim*

**Khắc phục**

- Luôn bảo đảm tính bí mật của thông tin dùng để giải mã (private key, salt, IV, .v.v)
- Tránh sử dụng các hàm băm yếu (sha1, md5, .v.v) hoặc mã hoá yếu (RC4, DES, .v.v)

## KẾT LUẬN

- Pentest lỗ hổng trên ứng dụng Android là một phần quan trọng của quy trình bảo mật nhằm đảm bảo an toàn cho dữ liệu và hệ thống của người dùng. Quá trình này bao gồm việc xác định, khai thác và khắc phục các lỗ hổng bảo mật tiềm ẩn trong ứng dụng.
- Trong bối cảnh mối đe dọa an ninh ngày càng tăng, việc thực hiện pentest định kỳ và liên tục cập nhật kiến thức về các lỗ hổng mới là rất cần thiết. Điều này không chỉ giúp bảo vệ ứng dụng và người dùng mà còn góp phần nâng cao uy tín và độ tin cậy của các nhà phát triển. Kết quả từ quá trình pentest sẽ cung cấp thông tin quý giá giúp cải thiện chất lượng và bảo mật của ứng dụng, đảm bảo rằng ứng dụng không chỉ đáp ứng nhu cầu người dùng mà còn bảo vệ họ khỏi các nguy cơ bảo mật tiềm ẩn.



## TÀI LIỆU THAM KHẢO

- [1] "Android operating system", Online. britannica. Accessed on: June 25, 2024. [Online]. Available: <https://www-britannica-com.translate.goog/technology/Android-operating-system>
- [2] "Mobile Operating System Market Share Worldwide", Online. statcounter. Accessed on: June 25, 2024. [Online]. Available: <https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-200901-202405>
- Bùi Đức Phú, "App\_Coffee", version 1.0.0. [Online]. Accessed: Jul. 25, 2024 Available: [https://github.com/Bui-Duc-Phu/App\\_Coffee](https://github.com/Bui-Duc-Phu/App_Coffee).