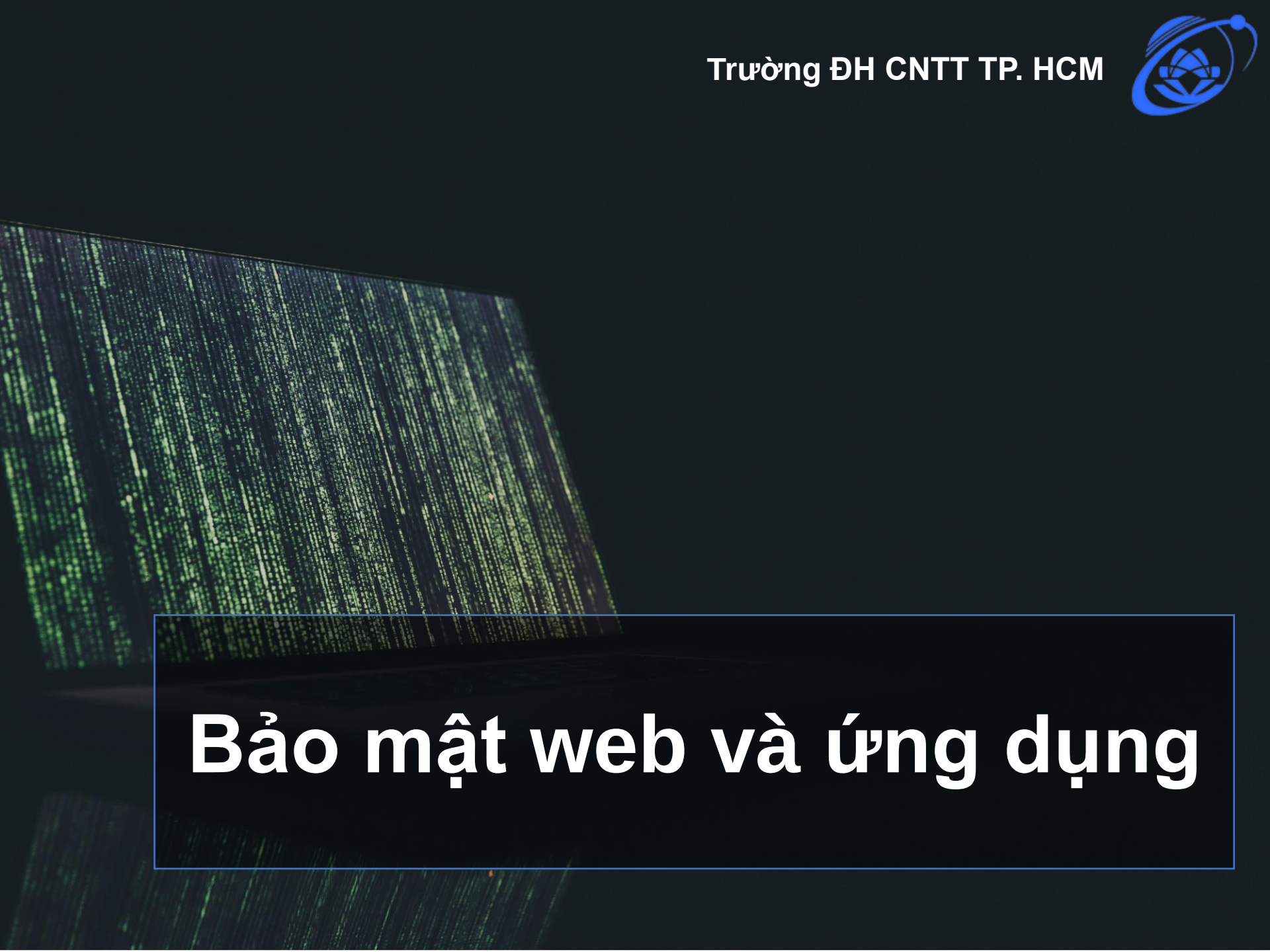
Trường ĐH CNTT TP. HCM



Bảo mật web và ứng dụng

Nội dung

- Mô hình an toàn ứng dụng Android
- Quyền hạn của ứng dụng
- Package Management

Mô hình an toàn trong Android

Android Security Model

- Application Sandboxing
- Permission
- IPC (Inter-Process Communication)
- Code Signing & Platform Key

Ứng dụng Android

- **System app**:
  - ✓ Đi kèm với OS, chỉ có thể đọc (read-only) trên thiết bị
  - ✓ Nằm trong thư mục ***/system***, *không thể bị xóa hay chỉnh sửa bởi người dùng*
- **User-installed app**
  - ✓ Nằm trên phân vùng read-write, tại thư mục ***/data***
  - ✓ Độc lập với nhau

Application Sandboxing

- Android gán một UID (thường được gọi là app ID), tại thời điểm cài đặt ứng dụng.
- Ứng dụng Android được cô lập (isolated) ở cả 2 mức: tiến trình (process) và tập tin (file).
  - Mỗi tiến trình của ứng dụng Android được thực thi như một user độc lập (với UID riêng biệt)
  - Mỗi ứng dụng có 1 thư mục mà ***chỉ có ứng dụng đó*** được quyền đọc và ghi (phân quyền dựa trên UID)
- Android không có file /etc/passwd, thông tin các UID được định nghĩa trong tập tin *android_filesystem_config.h*

https://android.googlesource.com/platform/system/core/+/master/libcutils/include/private/android_filesystem_config.h

Application Sandboxing

- Các daemon hệ thống chạy ngầm và ứng dụng được gắn với các UID, một vài daemons chạy dưới quyền root user (UID 0).
- **UID** của các **dịch vụ hệ thống** được bắt đầu **1000 (AID_SYSTEM)**
- **UID** của mỗi ứng dụng có giá trị từ **10000** trở lên **(AID_APP)**
- Username của mỗi app có quy tắc như sau:
  - **app_XXX**
  - Hoặc u**Y**_a**XXX**
  - Trong đó, XXX là **offset** được tính từ AID_APP, Y là Android user ID

Application Sandboxing

```
$ ps
--snip--
u0_a37    16973 182   941052  60800 ffffffff 400d073c S com.google.android.email❶
u0_a8     18788 182   925864  50236 ffffffff 400d073c S com.google.android.dialer
u0_a29    23128 182   875972  35120 ffffffff 400d073c S com.google.android.calendar
u0_a34    23264 182   868424  31980 ffffffff 400d073c S com.google.android.deskclock
--snip--
```

Tiến trình có **UID = 10037** tương ứng với username là **u0_a37**

```
# pm list packages
# dumpsys | grep -A18 "Package \[com.example.app1\]"
```

```
Package [com.example.[redacted]] (d13b03c):
  userId=10153
  pkg=Package{425ddb9 com.example.helloapplication}
  codePath=/data/app/~~rEI0o0VZQRxInCJa1S3u0g==/com.example
  resourcePath=/data/app/~~rEI0o0VZQRxInCJa1S3u0g==/com.exa
  legacyNativeLibraryDir=/data/app/~~rEI0o0VZQRxInCJa1S3u0g
```

Application Sandboxing

- Dữ liệu của ứng dụng được chứa trong thư mục ***/data/data*** trên single-user device.
- Các ứng dụng có thể tùy chỉnh tạo file với mong muốn cấp quyền truy cập cho các ứng dụng khác bằng cách sử dụng flag:
  - MODE_WORLD_READABLE (ứng với bits giá trị S_IROTH trên tập tin)
  - MODE_WORLD_WRITEABLE (ứng với bits giá trị S_IWOTH trên tập tin)

```
FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_WORLD_READABLE);
fos.write(string.getBytes());
fos.close();
```

```
# ls -l /data/data/com.google.android.email
drwxrwx--x u0_a37   u0_a37          app_webview
drwxrwx--x u0_a37   u0_a37          cache
drwxrwx--x u0_a37   u0_a37          databases
drwxrwx--x u0_a37   u0_a37          files
--snip--
```

Application Sandboxing

- UID của ứng dụng được quản lý trong tập tin:
  - **/data/system/packages.xml**
  - **/data/system/packages.list**

```
# grep 'com.google.android.email' /data/system/packages.list
com.google.android.email 10037 0 /data/data/com.google.android.email default 3003,1028,1015
```

- Trong đó, giá trị các trường như sau:
  - Thứ 1: package name
  - Thứ 2: UID của ứng dụng
  - Thứ 3: debug flag (1 là debug)
  - Thứ 4: đường dẫn thư mục của ứng dụng
  - Thứ 5: seinfo label (được sử dụng bởi SELinux)
  - Thứ 6: Danh sách các GID bổ sung mà app chạy. Mỗi GID sẽ gắn với một quyền (Android permission); GID được tạo ra dựa trên các quyền cấp cho ứng dụng.

Application Sandboxing

- Các ứng dụng có thể được cài đặt bằng cách sử dụng **cùng UID**, được gọi là **shared user ID**.

→ chạy cùng tiến trình, chia sẻ dữ liệu với nhau

- Để sử dụng cùng UID, các ứng dụng cần được ký bởi cùng code signing key
- Nếu thêm shared user ID vào một app đã được cài đặt → thay đổi UID của app → hệ thống vô hiệu hóa app này

  → Do đó, một ứng dụng cần được thiết kế với 1 shared ID từ đầu

```
# grep ' 10012 ' /data/system/packages.list
com.android.keyguard 10012 0 /data/data/com.android.keyguard platform 1028,1015,1035,3002,3001
com.android.systemui 10012 0 /data/data/com.android.systemui platform 1028,1015,1035,3002,3001
```

Quyền hạn (Permission)

- Một ứng dụng Android yêu cầu được cấp quyền bằng cách khai báo trong tập tin ***AndroidManifest.xml***.
- Khi được gán các quyền hạn nhất định, nó luôn tồn tại trong ứng dụng và không thể bị thu hồi.
- Tuy nhiên, đối với việc truy cập private key và tài khoản user sẽ bị hệ thống yêu cầu cấp phép tường minh từ người dùng cho mỗi đối tượng, cho dù đã được cấp quyền lúc cài đặt.

Quyền hạn (Permission)

- Các ứng dụng có thể định nghĩa:
  - Quyền hạn tùy chỉnh (Custom permission)
  - Hạn chế truy cập (protection levels)

  → giới hạn truy cập tới các tài nguyên và dịch vụ của những ứng dụng được tạo bởi cùng một tác giả.
- Quyền hạn có thể được thực thi ở nhiều mức khác nhau:
  - Tài nguyên hệ thống cấp thấp: Linux kernel kiểm tra thông tin UID và GID, với owner và bit truy cập của tài nguyên.
  - Tài nguyên hệ thống cấp cao: Android OS/ component hoặc cả 2.

Quyền hạn (Permission)

AndroidManifest.xml

```
<manifest ...>
    <uses-permission android:name="android.permission.CAMERA"/>
    <application ...>
        ...
    </application>
</manifest>
```

Inter-Process Communication - IPC

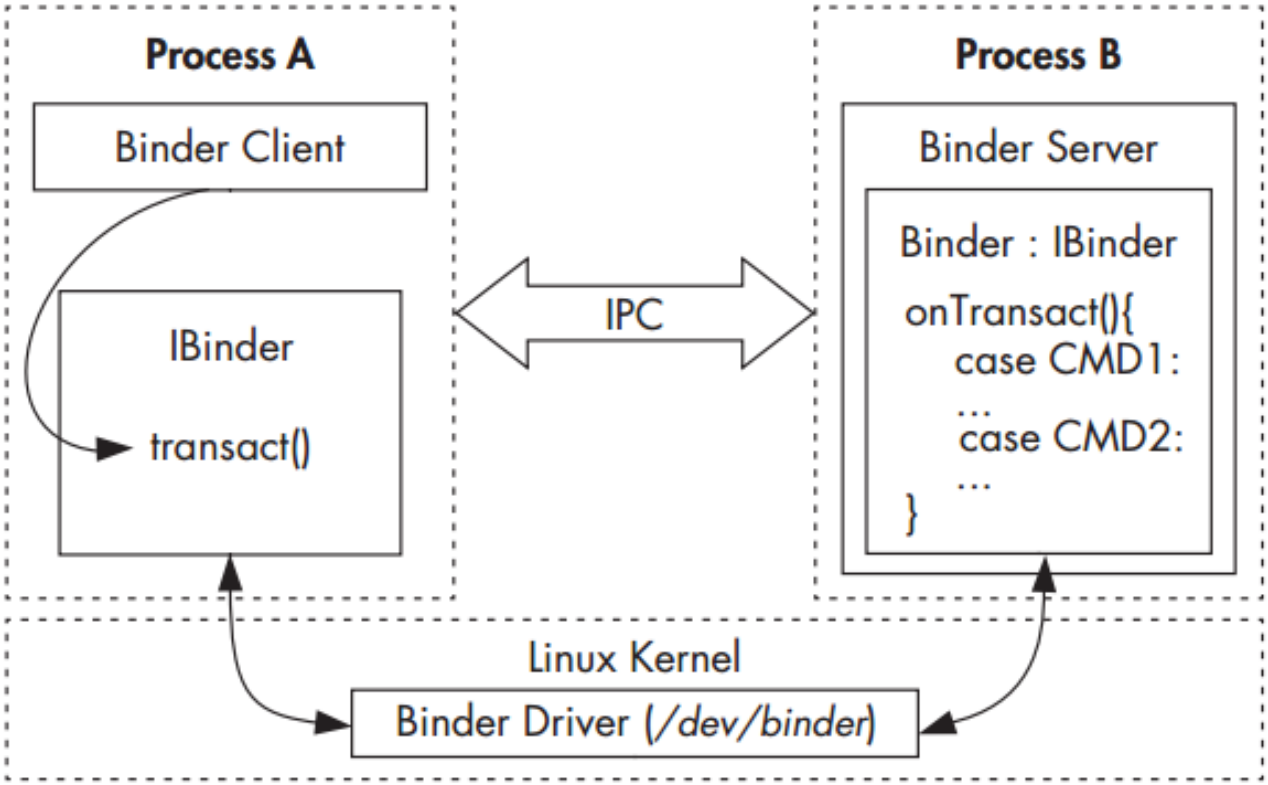
- Process Isolation: Các tiến trình Android độc lập với nhau.
- Một số dịch vụ chia sẻ giữa các tiến trình đã có:
  - File
  - Socket
  - Pipe
  - Semaphore
  - Shared memory
  - Message queue,…

  → **IPC cho Android?**

IPC: Binder (New IPC)

- Binder: phiên bản IPC mới được phát triển cho Android, dựa trên OpenBinder.
- Dựa trên kiến trúc thành phần phân tán (distributed component architecture):
  - Giống với Windows Component Object Model (COM) và Common Object Broker Request Architectures (CORBA trên Unix)
  - Khác với COM và CORBA, Binder chỉ chạy trên một thiết bị, không hỗ trợ RPC (remote procedure call) thông qua mạng

IPC: Binder (New IPC)



IPC: Binder Security

- Các đối tượng Binder luôn được duy trì trong một định danh – identity duy nhất giữa các tiến trình.
  - VD: tiến trình A tạo đối tượng Binder và chuyển cho tiến trình B, sau đó B chuyển cho tiến trình C. Khi đó các lời gọi từ tất cả 3 tiến trình này đều được thực hiện trên cùng 1 đối tượng Binder của tiến trình A (không gian bộ nhớ tiến trình A)
- Binder object có thể hoạt động như token bảo mật → hỗ trợ cơ chế **capability-based security model** của Android.

IPC: Capability-based Security

- Các ứng dụng được cấp phép truy cập tới những tài nguyên nhất định bằng cách cấp cho chúng ***năng lực không thể giả mạo*** - vừa tham chiếu tới các đối tượng tài nguyên vừa đóng gói các quyền truy cập tới nó.
- Android Binder không sử dụng cơ chế an ninh ACL (access control list)

IPC: Capability-based Security

- Đối tượng Binder hoạt động như một năng lực (capability), và được gọi là *Binder token*.
- Việc sở hữu Binder token cho phép tiến trình có toàn quyền truy cập tới Binder → cho phép nó thực thi các giao dịch Binder trên đối tượng đích (target).

IPC - Binder: Truy cập đối tượng Binder

- Binder sử dụng **context manager** (tham chiếu tới Binder) để cho phép xác định các dịch vụ (service discovery)
- Trong Android, context manager có tên là **service manager –** khởi chạy trong quá trình boot.
- Các dịch vụ được đăng ký với hệ thống bằng cách truyền vào service manager tên của dịch vụ và tham chiếu Binder.
- Sau khi đăng ký, các dịch vụ của hệ thống có thể được truy cập bằng tên thông qua tham chiếu Binder.
- Cơ chế kiểm tra quyền hạn qui định việc truy cập vào Binder chỉ giới hạn trên một số tiến trình hệ thống được đăng ký.
- VD: Chỉ cho phép tiến trình thực thi với UID 1002 (AID_BLUETOOTH) đăng ký dịch vụ hệ thống Bluetooth (Bluetooth system service).

IPC - Binder: Truy cập đối tượng Binder

- Danh sách các dịch vụ hệ thống đã đăng ký và hiện thực Ibinder interface có thể xem qua lệnh qua câu lệnh

  service list

  Ví dụ: trên các thiết bị Android 4.4

```
$ service list
service list
Found 79 services:
0       sip: [android.net.sip.ISipService]
1       phone: [com.android.internal.telephony.ITelephony]
2       iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
3       simphonebook: [com.android.internal.telephony.IIccPhoneBook]
4       isms: [com.android.internal.telephony.ISms]
5       nfc: [android.nfc.INfcAdapter]
6       media_router: [android.media.IMediaRouterService]
7       print: [android.print.IPrintManager]
8       assetatlas: [android.view.IAssetAtlas]
9       dreams: [android.service.dreams.IdreamManager]
--snip--
```

Binder

- Binder đảm bảo UID và PID của các tiến trình (caller) không thể bị giả mạo → nhiều dịch vụ hệ thống dùng binder để kiểm soát ứng dụng gọi tới API nhạy cảm thông qua IPC.
- VD: Kiểm tra ứng dụng có quyền bật tính năng Bluetooth nếu có system UID (UID = 1000)

```
public boolean enable() {
    if ((Binder.getCallingUid() != Process.SYSTEM_UID) &&
        (!checkIfCallerIsForegroundUser())) {
        Log.w(TAG,"enable(): not allowed for non-active and non-system user");
        return false;
     }
--snip--
}
```

Binder

- Các quyền hạn trên từng component của ứng dụng (AndroidManifest.xml) cũng được triển khai dựa trên việc kiểm tra UID của caller (thực hiện gọi) – lấy được từ Binder.
- Hệ thống sử dụng **package database** để xác định quyền hạn mà các thành phần được gọi (callee) yêu cầu; sau đó ánh xạ caller UID với package name và lấy danh sách các quyền hạn đã cấp cho caller.
- Nếu các quyền hạn được yêu cầu nằm trong danh sách, việc thực hiện truy cập thành công vào callee.
- Nếu không hợp lệ - thất bại, hệ thống sẽ đưa ra ngoại lệ SecurityException

Code Signing & Platform Keys

- Tất cả các ứng dụng đều được ký bởi nhà phát triển
- **Tập tin cài đặt APK** là một extension của định dạng file JAR → việc ký lên ứng dụng APK cũng dựa trên quy trình ký lên tập tin JAR.
- Android sử dụng chữ ký trên file APK để đảm bảo:
  - Việc cập nhật ứng dụng luôn đến từ một nhà phát triển (gọi là same origin policy).
  - Tạo mối quan hệ tin cậy giữa các ứng dụng
  - Thực hiện so sánh giữa phiên bản hiện tại với phiên bản cập nhật của 1 ứng dụng.

Code Signing & Platform Keys

- Các **ứng dụng hệ thống (System app)** được ký bởi một số các **platform key**.
- Các thành phần hệ thống có thể chia sẻ tài nguyên và chạy trong cùng một tiến trình khi nó được ký bởi cùng 1 platform key.
- Platform key được tạo ra và quản lý bởi những người cài đặt phiên bản hệ điều hành Android lên thiết bị: nhà sản xuất, nhà mạng, Google cho các thiết bị Nexus, hoặc những người dùng tự xây dựng một phiên bản Android tùy chỉnh,…

Multi-User Support

- Android ban đầu hướng đến thiết bị đơn người dùng → UID được gán cho từng ứng dụng thay vì gán cho nhiều user.
- Từ phiên bản Android 4.2, Android hỗ trợ tính năng multi-user (chỉ trên thiết bị tablet, ver 5.0 hỗ trợ nhiều thiết bị).
- Có thể vô hiệu hóa tính năng multi-user trên thiết bị cầm tay bằng cách thiết lập số lượng tối đa user là 1.
- Mỗi user có một mã định danh user (ID) duy nhất, bắt đầu bằng số 0, được cấp một thư mục riêng biệt cho nó tại đường dẫn: **/data/system/users/<user ID>**

SEAndroid

- **Discretionary access control (DAC):** user có quyền truy cập đến file có thể chuyển quyền cho user khác tuỳ ý
- **Mandatory access control (MAC)**: dựa trên policy, chỉ có thể sửa bởi quản trị viên
- **Security Enhanced Linux (SELinux)** là một phiên bản của MAC cho Linux kernel.
- Từ phiên bản Android 4.3, **SEAndroid** - bản sửa đổi của SELinux dành cho Android được tích hợp

System Updates

- Over-the-air (OTA): tải file ZIP với code signature, sau đó được biên dịch bởi recovery (còn gọi là recovery OS)
  - Sự cần thiết phải truy cập đến tất cả các phần cứng trên thiết bị khi cập nhật hệ thống: firmware, bootloader,… -> Android đưa ra một minimal OS với các đặc quyền truy cập tới tất cả các phần cứng của thiết bị, gọi là **recovery OS**.
  - Trên các thiết bị Android được sản xuất, recovery chỉ chấp nhận các cập nhật được ký bởi nhà SX thiết bị.
  - File cập nhật được ký bằng cách tích hợp chữ ký vào phần mở rộng của file ZIP, trong phần Comment.
  - Recovery trích xuất và xác nhận thông tin chữ ký trước khi cài đặt phiên bản cập nhật cho hệ thống.
  - Boot loader unlocking: thay thế recovery và system image → cài đặt phiên bản cập nhật từ các bên thứ 3. (dữ liệu user sẽ bị xóa)
- Kết nối với PC và cập nhật hệ điều hành bằng cách sử dụng ADB (Android debug bridge)

Verify Boot

- Phiên bản Android 4.4 hỗ trợ xác thực boot bằng **verity** (của Linux's Device-Mapper).
- Kiểm tra tính toàn vẹn của các block trên thiết bị dựa vào cây mã hóa:
  - Nút lá: giá trị băm của một khối dữ liệu (data block)
  - Nút trung gian: chứa giá trị băm của các nút con của nó
  - Nút gốc: chứa giá trị băm của tất cả các nút trực tiếp thuộc về nó.

  →chỉ cần dựa vào **nút gốc**
- Quá trình xác thực được thực hiện bởi một khóa công cộng RSA trong mỗi phân vùng boot.
- Từng block của thiết bị sẽ được kiểm tra lúc runtime bằng cách tính giá trị băm của block với giá trị được ghi trong cây băm.
- Bởi vì tất cả các kiểm tra được thực hiện bởi kernel, do đó qui trình boot cần xác thực tính toàn vẹn của kernel nhằm cho các verified boot hoạt động.

Quyền hạn – Android Permissions

Quyền hạn – Permission

- Permission là ***một chuỗi (string)*** qui định khả năng thực hiện một tác vụ nào đó:
  - Truy cập vào tài nguyên vật lý (VD: Truy cập thẻ nhớ SD của máy)
  - Chia sẻ dữ liệu (danh sách user đăng ký)
  - Cho phép ứng dụng bên thứ 3 khởi động/truy cập vào các thành phần của ứng dụng

Quyền hạn – Permission

- Danh sách các quyền hạn của hệ thống:

```
$ pm list permissions
All Permissions:

permission:android.permission.REBOOT❶
permission:android.permission.BIND_VPN_SERVICE❷
permission:com.google.android.gallery3d.permission.GALLERY_PROVIDER❸
permission:com.android.launcher3.permission.RECEIVE_LAUNCH_BROADCASTS❹
--snip--
```

Quản lý Permissions

- Quyền hạn ***được gán*** cho từng ứng dụng ***tại thời điểm cài đặt ứng dụng*** bằng dịch vụ ***Package manager***.
- Package manager được lưu trữ trong file */data/system/packages.xml*
- Package manager quản lý 1 CSDL tập trung về các package đã cài (pre-installed và user-installed) với các thông tin:
  - Đường dẫn cài đặt
  - Phiên bản, Chứng chỉ ký
  - Danh sách các quyền hạn của từng package
  - Danh sách tất cả các quyền hạn được định nghĩa trên thiết bị

Quyền hạn – Quản lý Permission

Ví dụ: Entry của 1 application trong packages.xml

```
<package name="com.google.android.apps.translate"
         codePath="/data/app/com.google.android.apps.translate-2.apk"
         nativeLibraryPath="/data/app-lib/com.google.android.apps.translate-2"
         flags="4767300" ft="1430dfab9e0" it="142cdf04d67" ut="1430dfabd8d"
         version="30000028"
         userId="10204"❶
         installer="com.android.vending">

    <sigs count="1">
        <cert index="7" />❷
    </sigs>

    <perms>❸
        <item name="android.permission.READ_EXTERNAL_STORAGE" />
        <item name="android.permission.USE_CREDENTIALS" />
        <item name="android.permission.READ_SMS" />
        <item name="android.permission.CAMERA" />
        <item name="android.permission.WRITE_EXTERNAL_STORAGE" />
        <item name="android.permission.INTERNET" />
        <item name="android.permission.MANAGE_ACCOUNTS" />
        <item name="android.permission.GET_ACCOUNTS" />
        <item name="android.permission.ACCESS_NETWORK_STATE" />
        <item name="android.permission.RECORD_AUDIO" />
    </perms>

    <signing-keyset identifier="17" />
    <signing-keyset identifier="6" />
</package>
```

Thông tin chung: tên package, đường dẫn, version…

Permissions

Permission Protection Level

Protection level của permission đặc trưng cho những rủi ro có thể có bao hàm trong permission đó, và hệ thống nên làm gì để quyết định có cấp permission được yêu cầu hay không.

- **normal**: rủi ro thấp → permission sẽ tự động được cấp mà không đòi hỏi xác nhận từ người dùng.
  - VD: ACCESS_NETWORK_STATE, GET_ACCOUNTS
- **dangerous**: các permission có thể cho phép truy cập dữ liệu người dùng hay các hình thức kiểm soát thiết bị → Trước khi cấp phép, Android hiển thị một thông báo cho người dùng chấp thuận.
  - VD: READ_SMS, CAMERA

Permission Protection Level

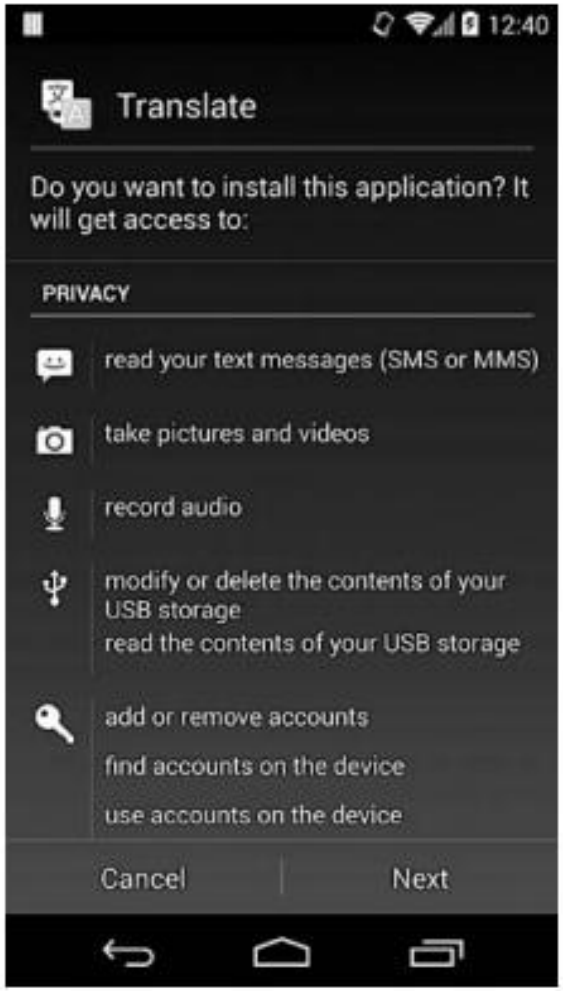


Figure 2-1: Default Android application install confirmation dialog

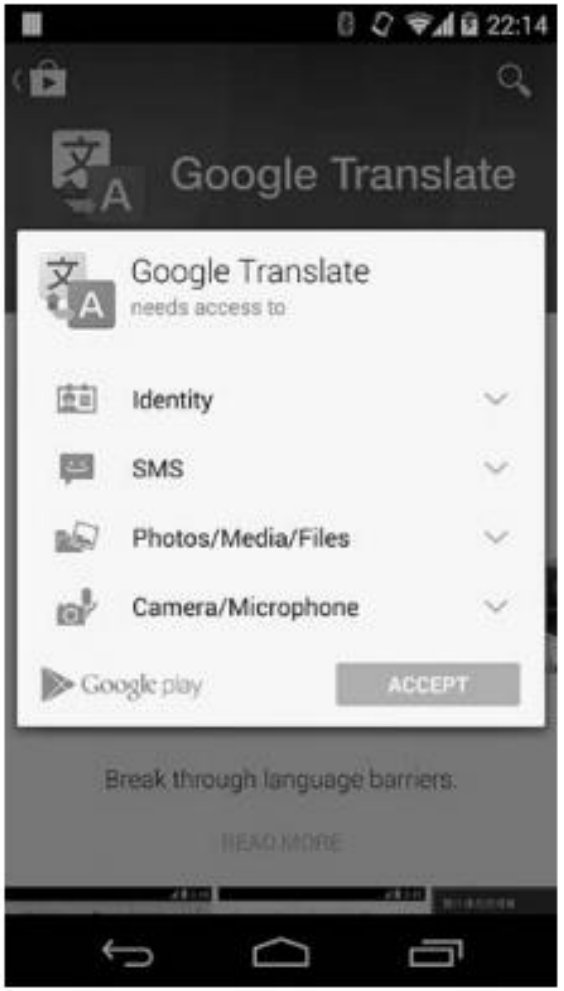


Figure 2-2: Google Play Store client application install confirmation dialog

Permission Protection Level

- **signature**: chỉ được cấp các quyền hạn ở mức bảo vệ là signature cho các ứng dụng được ký bởi cùng một khóa bởi ứng dụng khai báo quyền đó.
  - VD: NET_ADMIN, ACCESS_ALL_EXTERNAL_STORAGE (multi-user external storage)

  → Protection level **mạnh nhất**!
- **signatureOrSystem**: được cấp cho các ứng dụng hoặc một phần của system image, hoặc được ký với cùng một khóa với ứng dụng khai báo quyền hạn.
  - Giúp các vendor không cần chia sẻ khóa trên các ứng dụng cài đặt sẵn trên thiết bị.
  - Android phiên bản 4.3 trở về trước: tự động gán quyền hạn ở mức signatureOrSystem cho các ứng dụng hệ thống được cài
  - Android 4.4 trở đi, ứng dụng cần được cài ở */system/priv-app/*

Custom Permission

- Định nghĩa bởi **third-party applications**, có thể được gán các **protection level** như các system permission.
- Lưu ý: Android chỉ có thể cấp các permission đã biết → ứng dụng định nghĩa permission cần được cài đặt trước ứng dụng sử dụng permission.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.app"
    android:versionCode="1"
    android:versionName="1.0" >
    --snip--
<permission
    android:name="jcom.example.app.permission.PERMISSION1"
    android:label="@string/permission1_label"
    android:description="@string/permission1_desc"
    android:permissionGroup="com.example.app.permission-group.TEST_GROUP"
    android:protectionLevel="signature" />❸
```

Quyền hạn – Yêu cầu Permission

- Ứng dụng có thể yêu cầu được cấp quyền hạn bằng cách thêm vào thẻ **<uses-permission>** trong file ***AndroidManifest.xml***

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.app"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    --snip--
    <application android:name="SampleApp" ...>
    --snip--
    </application>
</manifest>
```

Quyền hạn – Gán Permission

- Quyền hạn và thuộc tính của tiến trình:
  - Mỗi tiến trình trong Android gắn liền với các thuộc tính như UID, GID, và tập hợp các GID bổ sung.
  - Các quyền hạn được cấp cho ứng dụng được ánh xạ tới GID và gán cho danh sách GID bổ sung của tiến trình.
- Ánh xạ giữa GID và quyền hạn được qui định trong ***/etc/permission/platform.xml***

Quyền hạn – Gán Permission

Ví dụ trong **platform.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<permissions>
    --snip--
    <permission name="android.permission.INTERNET" >❶
        <group gid="inet" />
    </permission>

    <permission name="android.permission.WRITE_EXTERNAL_STORAGE" >❷
        <group gid="sdcard_r" />
        <group gid="sdcard_rw" />
    </permission>

    <assign-permission name="android.permission.MODIFY_AUDIO_SETTINGS"
                                                uid="media" />❸
    <assign-permission name="android.permission.ACCESS_SURFACE_FLINGER"
                                                uid="media" />❹
    --snip--
</permissions>
```

→ Các tiến trình của các ứng dụng nào được cấp phép quyền INTERNET hay WRITE_EXTERNAL_STORAGE sẽ có GID tương ứng được thêm vào phần GID bổ sung của tiến trình

Quyền hạn – Gán Permission

- Thẻ **<assign-permission>** có nhiệm vụ ngược lại – gán quyền hạn cho các tiến trình đang chạy với UID tương ứng

```
<?xml version="1.0" encoding="utf-8"?>
<permissions>
    --snip--
    <permission name="android.permission.INTERNET" >❶
        <group gid="inet" />
    </permission>

    <permission name="android.permission.WRITE_EXTERNAL_STORAGE" >❷
        <group gid="sdcard_r" />
        <group gid="sdcard_rw" />
    </permission>

    <assign-permission name="android.permission.MODIFY_AUDIO_SETTINGS"
        uid="media" />❸
    <assign-permission name="android.permission.ACCESS_SURFACE_FLINGER"
        uid="media" />❹

    --snip--
</permissions>
```

Quyền hạn – Gán Permission

- Do Android không có tập tin **/etc/group** nên việc ánh xạ giữa group name và GID được định nghĩa trong file *android_filesystem_config.h*

```
--snip--
#define AID_ROOT             0  /* traditional unix root user */
#define AID_SYSTEM        1000  /* system server */
--snip--

#define AID_SDCARD_RW     1015  /* external storage write access */
#define AID_SDCARD_R      1028  /* external storage read access */
#define AID_SDCARD_ALL    1035  /* access all users external storage */
--snip--
#define AID_INET          3003  /* can create AF_INET and AF_INET6 sockets */
--snip--

struct android_id_info {
    const char *name;
    unsigned aid;
};

static const struct android_id_info android_ids[] = {
    { "root",          AID_ROOT, },
    { "system",        AID_SYSTEM, },
    --snip--
    { "sdcard_rw",     AID_SDCARD_RW, },❶
    { "sdcard_r",      AID_SDCARD_R, },❷
    { "sdcard_all",    AID_SDCARD_ALL, },
    --snip--
    { "inet",          AID_INET, },❸
};
```

Quyền hạn – Gán Permission

- Process Attribute Assignment: **zygote** và tiến trình của ứng dụng
  - Tất cả các ứng dụng khởi chạy đều có tiến trình cha là **zygote**
  - Giúp khởi động nhanh tiến trình ứng dụng

```
$ ps
USER      PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
root      1     0     680    540     ffffffff 00000000 S /init❶
--snip--
root      181   1     858808 38280   ffffffff 00000000 S zygote❷
--snip--
radio     1139  181   926888 46512   ffffffff 00000000 S com.android.phone
nfc       1154  181   888516 36976   ffffffff 00000000 S com.android.nfc
u0_a7     1219  181   956836 48012   ffffffff 00000000 S com.google.android.gms
```

Component Security & Permission: Component trong **Ứng dụng Android?**

- Activity
- Service
- Content Provider
- Broadcast Receiver

Component Security & Permission: Activity

- Để bảo vệ truy cập vào **Activity**, cần thêm thuộc tính **permission** vào trong dòng khai báo Activity đó trong tập tin ***AndroidManifest.xml***
- Khi đó, một ứng dụng X khác muốn khởi động/truy cập Activity thông qua **Intent** phải có permission mà Activity yêu cầu → định nghĩa thẻ <user-permission> trong file manifest của X.
- Ví dụ: Để gọi **activity1** của app **test1** bên dưới, app **X** cần có quyền ***START_ACTIVITY1***.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.testapps.test1">
 ...
    <activity android:name=".Activity1"

 android:permission="com.example.testapps.test1.permission.START_ACTIVITY1">
                <intent-filter>
                   ...
                </intent-filter>
    </activity>
  ...
</manifest>
```

Component Security & Permission: Service

- Để bảo vệ truy cập vào component **Service,** thực hiện tương tự như trên Activity.
- VD:
  - Khai báo **permission** cho Service
  - Giá trị trường **exported = true**
  - **Không** khai báo Intent Filter

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.testapps.test1">
 ...
    <service android:name=".MailListenerService"
             android:permission=
                "com.example.testapps.test1.permission.BIND_TO_MAIL_LISTENER"
             android:enabled="true"
             android:exported="true">
             <intent-filter></intent-filter>
    </service>
  ...
</manifest>
```

Component Security & Permission: Content Provider

- Để bảo vệ truy cập vào component **Content Provider,** thực hiện tương tự như trên Activity/Service bằng cách khai báo quyền hạn trong AndroidManifest.xml
- VD: Khai báo 2 quyền Read/Write trong Content Provider có tên **MailProvider:**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.testapps.test1">
 ...
     <provider android.name="com.example.testapps.test1.MailProvider"
               android.authorities="com.example.testapps.test1.mailprovider"
               android.readPermission="com.example.testapps.test1.permission.DB_READ"
              android.writePermission="com.example.testapps.test1.permission.DB_WRITE">
     </provider>
  ...
</manifest>
```

Component Security & Permission: Content Provider

- **URI permission** trên Content Provider: chỉ định một số thành phần trong dữ liệu app được cung cấp bởi Content Provider có thể được truy cập bởi các ứng dụng khác.
- Hình thức khai báo **global configuration:** gán giá trị thuộc tính **grantUriPermissions** của Provider thành **true**.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.testapps.test1">
 ...
     <provider android.name="com.example.testapps.test1.MailProvider"
               android.authorities="com.example.testapps.test1.mailprovider"
               android.readPermission="com.example.testapps.test1.permission.DB_READ"

              android.writePermission="com.example.testapps.test1.permission.DB_WRITE"
               android:grantUriPermissions="true">
      </provider>
   ...
 </manifest>
```

Component Security & Permission: Content Provider

- content://com.example.testapps.test1.mailprovider/messages/inbox/155
- content://com.example.testapps.test1.mailprovider/attachments/42

  → Chỉ định theo tập tài nguyên dữ liệu URI nhất định bằng cách thêm thẻ **<grant-uri-permission>** với thuộc tính **path** vào Content Provider.
- Lưu ý: Chỉ có tác dụng đối với thư mục URI hiện tại, không thể truy cập vào thư mục URI con của nó.
- VD: chỉ cho phép truy cập vào URI: /attachments/

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.testapps.test1">
 ...
    <provider android.name="com.example.testapps.test1.MailProvider"
              android.authorities="com.example.testapps.test1.mailprovider"
              android.readPermission="com.example.testapps.test1.permission.DB_READ"
            android.writePermission="com.example.testapps.test1.permission.DB_WRITE">
       <grant-uri-permission android:path="/attachments/" />
    </provider>
  ...
</manifest>
```

Component Security & Permission: Content Provider

- Chỉ định theo tập tài nguyên dữ liệu URI nhất định bằng cách thêm thẻ **<grant-uri-permission>** với thuộc tính **pathPrefix** vào Content Provider.
- VD: cho phép truy cập vào các URI có path bắt đầu bằng ***/messages/*** trong CSDL của app

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.testapps.test1">
 ...
    <provider android.name="com.example.testapps.test1.MailProvider"
              android.authorities="com.example.testapps.test1.mailprovider"
              android.readPermission="com.example.testapps.test1.permission.DB_READ"
            android.writePermission="com.example.testapps.test1.permission.DB_WRITE">
        <grant-uri-permission android:pathPrefix="/messages/" />
    </provider>
  ...
</manifest>
```

Component Security & Permission: Content Provider

- Chỉ định theo tập tài nguyên dữ liệu URI nhất định bằng cách thêm thẻ **<grant-uri-permission>** với thuộc tính **pathPattern** vào Content Provider.
- VD: cho phép truy cập vào các URI có path có chưa từ ***public*** trong URI dẫn đến CSDL của app

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.testapps.test1">
 ...
     <provider android.name="com.example.testapps.test1.MailProvider"
               android.authorities="com.example.testapps.test1.mailprovider"
               android.readPermission="com.example.testapps.test1.permission.DB_READ"
           android.writePermission="com.example.testapps.test1.permission.DB_WRITE">
         <grant-uri-permission android:pathPattern=".*public.*" />
     </provider>
  ...
</manifest>
```

Component Security & Permission: Broadcast Receiver

- **Giới hạn ứng dụng có thể gửi thông tin cho 1 receiver**:
  - Chỉ định **permission** tương ứng trong khai báo Receiver ở tập tin *AndroidManifest.xml*
  - Hoặc chỉ định trong mã lập trình:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.testapps.test1">
 ...
    <receiver android:name=".UIMailBroadcastReceiver"
              android:permission=
                 "com.example.testapps.test1.permission.MSG_NOTIFY_SEND">
        <intent-filter>
            <action android:name="com.example.testapps.test1.action.MESSAGE_RECEVIED">
        </intent-filet>
    </receiver>
  ...
</manifest>
```

```
IntentFilter intentFilter = new IntentFilter(MESSAGE_RECEIVED);
UIMailBroadcastReceiver rcv = new UIMailBroadcastReceiver();
myContext.registerReceiver(rcv, intentFilter,
    "com.example.testapps.test1.permission.MSG_NOTIFY_SEND", null);
```

Component Security & Permission: Broadcast Receiver

- **Giới hạn receiver có thể nhận thông tin từ ứng dụng**:
  - Chỉ định **permission** mà receiver cần có để nhận được thông báo:

```
sendBroadcast(new Intent("com.example.NOTIFY"),
              Manifest.permission.SEND_SMS);
```

  - Để nhận được thông tin broadcast, receiver cần được cấp permission tương ứng:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Package Management

Package Management

- Cấu trúc tập tin APK

```
apk/
|-- AndroidManifest.xml❶
|-- classes.dex❷
|-- resources.arsc❸
|-- assets/❹
|-- lib/❺
|   |-- armeabi/
|   |   `-- libapp.so
|   `-- armeabi-v7a/
|       `-- libapp.so
|-- META-INF/❻
|   |-- CERT.RSA
|   |-- CERT.SF
|   `-- MANIFEST.MF
`-- res/❼
    |-- anim/
    |-- color/
    |-- drawable/
    |-- layout/
    |-- menu/
    |-- raw/
    `-- xml/
```

Package Management

- Code Signing
  - Java Code Signing (JAR file level)
  - Android Code Signing

Package Management

- Code Signing
  - **Java Code Signing (JAR file level)**
    - Cấu trúc file MANIFEST.MF trong file APK: chứa nhiều entry gồm tên và giá trị digest của mỗi file trong file nén

```
Manifest-Version: 1.0
Created-By: 1.0 (Android)

Name: res/drawable-xhdpi/ic_launcher.png
SHA1-Digest: K/0Rd/ltOqSlgDD/9DY7aCNlBvU=

Name: res/menu/main.xml
SHA1-Digest: kG8WDil9urOf+F2AxgcSSKDhjnO=

Name: ...
```

Package Management

- **Java Code Signing (JAR file level)**
  - Thêm signature file (.SF) chứa:
    - Dữ liệu được ký
    - Chữ ký

```
Signature-Version: 1.0
SHA1-Digest-Manifest-Main-Attributes: ZKXxNW/3Rg7JA1rO+RlbJIP6IMA=
Created-By: 1.7.0_51 (Sun Microsystems Inc.)
SHA1-Digest-Manifest: zbOXjEhVBxEOz2ZC+B4OW25WBxo=❶

Name: res/drawable-xhdpi/ic_launcher.png
SHA1-Digest: jTeE2Y5L3uBdQ2g4OPB2n72L3dE=❷

Name: res/menu/main.xml
SHA1-Digest: kSQDLtTEO7cLhTH/cY54UjbbNBo=❸

Name: ...
```

Package Management

- **Java Code Signing (JAR file level)**
  - Kiểm tra tập tin chữ ký JAR sử dụng chuẩn mã hóa nào bằng cách sử dụng thư viện OpenSSL.

```
$ openssl sha1 -binary MANIFEST.MF |openssl base64❶
zbOXjEhVBxEOz2ZC+B4OW25WBxo=
$ echo -en "Name: res/drawable-xhdpi/ic_launcher.png\r\nSHA1-Digest: \
K/ORd/ltOqSlgDD/9DY7aCNlBvU=\r\n\r\n"|openssl sha1 -binary |openssl base64❷
jTeE2Y5L3uBdQ2g4OPB2n72L3dE=
```

Package Management

- **Java Code Signing (JAR file level)**
  - Quá trình Ký lên tập tin JAR, sử dụng ***jarsigner***:

```
$ jarsigner -keystore debug.keystore -sigalg SHA1withRSA test.apk androiddebugkey❶
$ jarsigner -keystore debug.keystore -verify -verbose -certs test.apk❷
--snip--

smk      965 Sat Mar 08 23:55:34 JST 2014 res/drawable-xxhdpi/ic_launcher.png

      X.509, CN=Android Debug, O=Android, C=US (androiddebugkey)❸
      [certificate is valid from 6/18/11 7:31 PM to 6/10/41 7:31 PM]

smk   458072 Sun Mar 09 01:16:18 JST 2013 classes.dex

      X.509, CN=Android Debug, O=Android, C=US (androiddebugkey)❹
      [certificate is valid from 6/18/11 7:31 PM to 6/10/41 7:31 PM]

         903 Sun Mar 09 01:16:18 JST 2014 META-INF/MANIFEST.MF
         956 Sun Mar 09 01:16:18 JST 2014 META-INF/CERT.SF
         776 Sun Mar 09 01:16:18 JST 2014 META-INF/CERT.RSA

  s = signature was verified
  m = entry is listed in manifest
  k = at least one certificate was found in keystore
  i = at least one certificate was found in identity scope

jar verified.
```

Package Management

- **Java Code Signing (JAR file level)**
  - Xem, trích xuất thông tin người ký (Signer)

```
$ keytool -list -printcert -jarfile test.apk
Signer #1:
Signature:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 4dfc7e9a
Valid from: Sat Jun 18 19:31:54 JST 2011 until: Mon Jun 10 19:31:54 JST 2041
Certificate fingerprints:
         MD5:  E8:93:6E:43:99:61:C8:37:E1:30:36:14:CF:71:C2:32
         SHA1: 08:53:74:41:50:26:07:E7:8F:A5:5F:56:4B:11:62:52:06:54:83:BE
         Signature algorithm name: SHA1withRSA
         Version: 3
```

Package Management

- **Java Code Signing (JAR file level)**
  - Sử dụng lệnh unzip với OpenSSL để trích xuất chứng chỉ đã ký lên file:

```
$ unzip -q -c test.apk META-INF/CERT.RSA|openssl pkcs7 -inform DER -print_certs -out cert.pem
```

Package Management

- **Android Code Signing:** dựa trên Java JAR signing
  - Sử dụng khóa công khai và chứng chỉ X.509
  - Chứng chỉ dùng trong Android có thể là tự ký (self-signed)
  - Các entry trong file APK phải được ký bởi cùng một signer (cùng certificate) thay vì một số entry trong JAR signing có thể không được ký hoặc được ký bởi một signer khác.

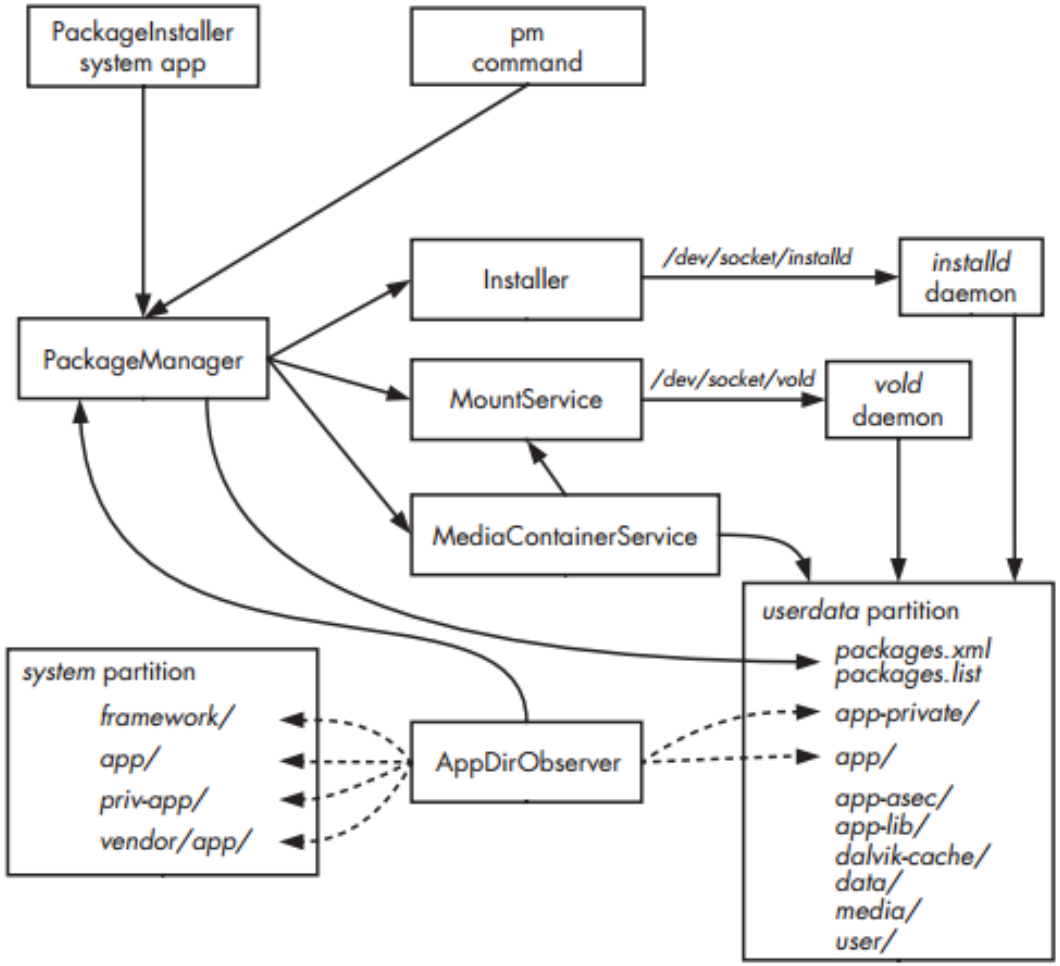
- **Công cụ**: ***signapk***

```
$ echo "keypwd"|openssl pkcs8 -in mykey.pem -topk8 -outform DER -out key.pk8 -passout stdin
```

```
$ java -jar signapk.jar cert.cer key.pk8 test.apk test-signed.apk
```

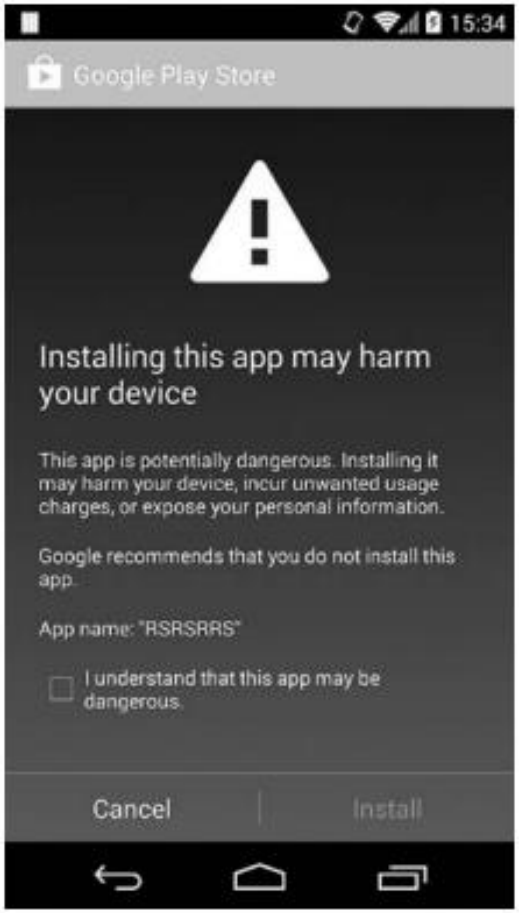
Package Management: Cài đặt APK

- Các thành phần của package management



Package Management: Package Verification

- Thông báo cảnh báo xác nhận ứng dụng trên Android:



Package Management: Package Verification

- Android hỗ trợ Package Verification:
  - ***PackageManagerService***
  - Một ứng dụng có thể tự đăng ký tính năng xác thực bằng cách khai báo 1 Broadcast Receiver có intent filter khớp với action PACKAGE_NEEDS_VERIFICATION và MIME type của tập tin APK (*application/vnd.android.package-archive*)

```
<receiver android:name=".MyPackageVerificationReceiver"
          android:permission="android.permission.BIND_PACKAGE_VERIFIER">
    <intent-filter>
        <action
            android:name="android.intent.action.PACKAGE_NEEDS_VERIFICATION" />
        <action android:name="android.intent.action.PACKAGE_VERIFIED" />
        <data android:mimeType="application/vnd.android.package-archive" />
    </intent-filter>
</receiver>
```

Package Management: Package Verification

- Android hỗ trợ Package Verification:
  - Thêm quyền ***PACKAGE_VERIFICATION_AGENT*** ứng dụng khai báo làm verifier.
  - Thêm thẻ ***<package-verifier>*** vào file manifest và liệt kê danh sách tên các package và khóa công khai của verifier.
  - PackageManagerService thực hiện quá trình xác thực khi có một ứng dụng được yêu cầu xác thực và thiết lập ***Settings.Global.PACKAGE_VERIFIER_ENABLE*** là ***true*** trong hệ thống.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.app">
        <package-verifier android:name="com.example.verifier"
                          android:publicKey="MIIB..." />
    <application ...>
     --snip--
    </application>
</manifest>
```

Package Management: Package Verification

- ***Google Play Implementation***
  - ✓ *PackageInstaller*
  - ✓ *Hoặc adb install*

Bài tập

*Chọn **2 ứng dụng Android** cùng một công ty phát triển, hãy phân tích:*

- Các quyền hạn được cấp của ứng dụng, các quyền hạn tùy chỉnh trên các thành phần của ứng dụng

- Xem các thông tin về chữ ký trên các file APK của ứng dụng.

Bài tập 2

Viết 2 ứng dụng Android cơ bản có các tính năng kiểm soát truy cập lẫn nhau bằng Permission vào các thành phần của nó.

Tài liệu tham khảo

- Chương 1-2-3 Sách **“Android Security Internals : An In-Depth Guide to Android's Security Architecture”**. Nikolay Elenkov. No Starch Press. 2015.

- Sách **“Application Security for the Android Platform: Processes, Permissions, and Other”.** Jeff Six. 2011.

Tài liệu tham khảo

- *https://securelist.com/crimeware-report-android-malware/112121/*
- https://www.tomsguide.com/news/this-android-malware-installs-backdoor-on-your-phone-delete-these-malicious-apps-now
- https://www.bleepingcomputer.com/news/security/anatsa-android-malware-downloaded-150-000-times-via-google-play/

Bảo mật web và ứng dụng

Trường ĐH CNTT TP. HCM