

Lecture 08

Android Permissions Demystified

Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song,
David Wagner

Operating Systems Practical

20 November, 2013

Introduction

Android Permission System

Stowaway

Keywords

Questions

Introduction

Android Permission System

Stowaway

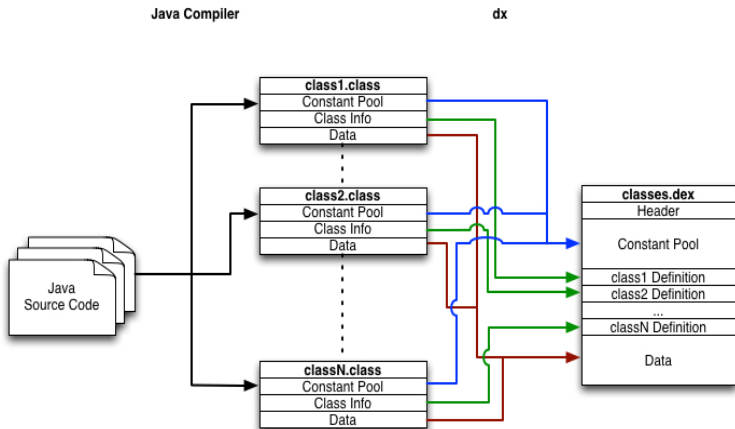
Keywords

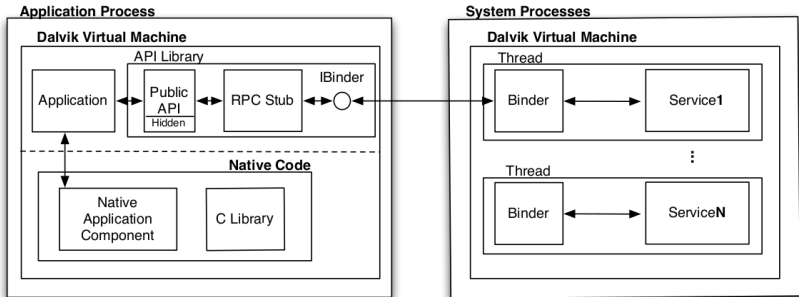
Questions

- ▶ Android OS security
- ▶ Coarse permission model
- ▶ A lot of research on Android permissions
- ▶ Applications with unnecessary permissions
- ▶ Paper doesn't focus on the malicious use of permissions

- ▶ Java source code → compiled into .dex byte-code file
- ▶ .dex file + *Manifest* file + resources = **.apk** archive
- ▶ Application isolation → **system level security**
 - ▶ Linux process, address space
 - ▶ VM (Dalvik Virtual Machine) for each application
 - ▶ unique Linux user ID
 - ▶ direct access only to its own data
 - ▶ API-based access to other apps' resources
- ▶ Not a single entry-point (no *main*)
- ▶ Applications can start each other
- ▶ Based on **Components** and **Intents**

- ▶ `.dex` - Dalvik Executable format
- ▶ Dalvik is optimised for mobile architectures
 - ▶ low memory consumption
 - ▶ Dex results in smaller binaries than JAR
- ▶ register-based architecture (JVM is stack-based)
- ▶ Java VM cannot execute Dalvik code
- ▶ 16-bit instructions
- ▶ copy-on-write memory sharing
- ▶ `dx` cross-compiler - works with `javac` output (oracle and openJDK, but not GCJ or other java compilers)





- ▶ Extends *Activity* base class
- ▶ User interfaces: UI elements(buttons, lists) and user input
- ▶ User interacts with one activity at a time
- ▶ Independent life-cycle, 4 states
 - ▶ active (running)
 - ▶ paused
 - ▶ stopped - still resides in memory
 - ▶ killed - removed from memory
- ▶ Activities stack
- ▶ Activities can launch other activities

- ▶ Extends *Service* base class
- ▶ Background processing
- ▶ It runs by default in the same process as the application
- ▶ Can provide functionality also for other applications

- ▶ Extends BroadcastReceiver base class
- ▶ Receive broadcast announcements, example: low battery, changed phone settings
- ▶ React to messages: start an activity or use NotificationManager
- ▶ Static registration - specified in the Manifest file
- ▶ Dynamic registration - Context.registerReceiver()
- ▶ Active only while it's responding to a broadcast message, no need to shut it down.

- ▶ Store and Share applications' data
- ▶ Required when sharing data between multiple applications
- ▶ Must be declared in the Manifest file
- ▶ Accessed with ContentResolver using URIs
- ▶ Uses relational databases
- ▶ Active only while it's responding to a request from a ContentResolver, no need to shut it down explicitly

► Intents

- Extend Intent class
 - Used for inter-component signaling
 - Used for starting activities, services and sending broadcast messages
 - IntentFilters specified in the Manifest file
 - Contain actions to be performed and data for these actions
 - Example: action = make a phone call, data = phone number
- ContentProviders do not use intents

- ▶ XML configuration file
- ▶ Every application must have it
- ▶ Contains:
 - ▶ application's name, icon, labels
 - ▶ linked libraries
 - ▶ **application components**: `<activity>`, `<service>`, `<receiver>`, `<provider>` tags
 - ▶ Activity shown at launch time
 - ▶ Intent filters
 - ▶ **Permissions**

Panoramio App:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.panoramio">
    <application android:icon="@drawable/icon">
        <activity android:name=".Panoramio" android:label="@string/app_name"
            android:theme="@style/Theme.Panoramio">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name=".ImageList" android:label="@string/app_name"
            android:theme="@android:style/Theme.Light"/>

        <activity android:name=".ViewImage" android:label="@string/app_name"
            android:theme="@style/Theme.Panoramio"/>

        <activity android:name=".ViewMap" android:label="@string/app_name"/>

        <uses-library android:name="com.google.android.maps" />
    </application>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
</manifest>
```

Introduction

Android Permission System

Stowaway

Keywords

Questions

- ▶ Android Framework Security
- ▶ Mandatory Access Control(MAC) enforced by middleware
- ▶ Components protected using **access permission labels**
 - ▶ declared in the AndroidManifest file
 - ▶ can not be changed after installation
 - ▶ 4 protection levels
 - ▶ normal - always granted
 - ▶ dangerous - requires user approval
 - ▶ signature - matching certificate
 - ▶ signature or system - matching certificate with system image

- ▶ At install-time each application requests a list of permissions
- ▶ All permissions must be granted at install time - *all or nothing*
- ▶ Protect access to Android components, services and APIs
 - ▶ e.g API for access to phone's hardware
- ▶ ~130 API-defined permissions in *Manifest.Permissions* class ¹
- ▶ Custom-defined permissions by developers
 - ▶ name conflicts may appear
 - ▶ current research on Android permissions doesn't take them into consideration

¹<http://developer.android.com/reference/android/Manifest.permission.html>

▶ **activity**

- ▶ restricts access to the activity
- ▶ checked when starting activity
- ▶ throw `SecurityException` if caller does not have required permission

▶ **service**

- ▶ restricts who can start, stop or bind to the service

▶ **receiver**

- ▶ restricts who can send broadcasts to the `BroadcastReceiver`
- ▶ checked at delivery, after broadcast was sent
- ▶ does not throw exception in case of permission failure

▶ **provider**

- ▶ restrict who can access the data
- ▶ read and write permissions
- ▶ checked when performing operations(e.g. query, insert)

- ▶ **Broadcast permissions**
 - ▶ permission label as parameter to the sending method (sendBroadcast)
- ▶ **Direct permission check**
 - ▶ *checkPermission* methods
 - ▶ check against PID, package name
- ▶ **URI Permissions**
 - ▶ Provide finer control over content sharing
 - ▶ Record level delegation
 - ▶ Set flags in the Intent that allow access (e.g. Intent.FLAG_GRANT_READ_URI_PERMISSION)
 - ▶ example: view mail attachments

- ▶ Usability study by the same authors
- ▶ Are users paying attention to the permissions?
- ▶ Do users understand the permissions?
- ▶ Can users make correct security decisions?
- ▶ Results: too few users comprehend or pay attention
- ▶ \Rightarrow security risks

Introduction

Android Permission System

Stowaway

Keywords

Questions

- ▶ The problem: unnecessary use of permissions
- ▶ The proposed solution: static analysis of API calls
- ▶ **Permission map** - identifies permissions for Intents, Content Provides, API calls
- ▶ **Stowaway tool** - determines if an app is overprivileged or not
- ▶ 2011 paper → research performed on Android 2.2 SDK

- ▶ Map of permissions for each method in the Android API
- ▶ Log permission checks → modified middleware
- ▶ Test cases for API calls, Intents, Content Providers

- ▶ Feedback-Directed Testing
 - ▶ *Randoop* unit test generator
 - ▶ receives a list of classes as input
 - ▶ tries to cover all possible combinations of calls
 - ▶ use return values as parameters for other methods
 - ▶ limitations
 - ▶ find an object of the correct type needed to invoke a method
 - ▶ object created through API calls with specific parameters
 - ▶ methods precede each other in a very specific order
 - ▶ native code generate segmentation faults if called out of order

- ▶ Customizable Test Case Generation
 - ▶ custom tool for building methods unit tests
 - ▶ list of method signatures as input
 - ▶ outputs at least one unit test for each method
 - ▶ allows manual adjustments of test sequences - order, parameters

- ▶ Manual Verification
 - ▶ solves inconsistencies
 - ▶ argument-dependent permission requirement
 - ▶ API calls order-dependent
 - ▶ test cases with and without permissions
 - ▶ identified methods that require INTERNET permission
 - ▶ tests run until no security exceptions appeared

- ▶ Content Providers
 - ▶ collected all URIs
 - ▶ test operations: query, insert, update, delete
 - ▶ run test with and without permissions
 - ▶ tests run until no security exceptions appeared
- ▶ Intents
 - ▶ send/receive between a pair of applications
 - ▶ searched API for all Intent action strings
 - ▶ tested all Intent action on the pair of apps
 - ▶ triggered system broadcasts

- ▶ 85% coverage of Android 2.2 API
- ▶ Proves the limitation of Android documentation of permissions
 - ▶ 1259 API calls with permission checks
 - ▶ only 78 methods with permission requirements in the documentation
 - ▶ documentation for 6 API calls is incorrect
- ▶ Characterized how permissions are distributed in the API
 - ▶ system permissions, hierarchical permissions, unused permissions
 - ▶ number of checks, permissions granularity
- ▶ Distribution of permissions per classes

- ▶ Available online for testing overprivileged applications
- ▶ Parses applications' API calls
- ▶ Identifies which declared permissions are actually needed

- ▶ Dissassembles Dexfiles - *Dedexer* tool
 - ▶ easy to parse method calls
- ▶ Identifies API calls
- ▶ Identifies Content Provider URIs
- ▶ Uses *ComDroid* for Intents

- ▶ Dex files parsing
- ▶ Identifies calls to API methods
- ▶ Problems
 - ▶ Java Reflection
 - ▶ use heuristics
 - ▶ Internet and External Storage permissions
 - ▶ enforced by the kernel not the middleware checker
 - ▶ Stowaway parses the app's XML files

- ▶ Parses URI strings
 - ▶ detects strings with "content:/"
 - ▶ detects URI API constants
- ▶ Cannot know the exact database operation from the URI

- ▶ Uses *ComDroid* static analysis tool²
- ▶ ComDroid tracks Intents
- ▶ For each Intent Stowaway checks
 - ▶ permission to send Intent
 - ▶ permission to receive Intent

²developed by the same authors - <http://www.comdroid.org/>

- ▶ Testbed of 940 applications
 - ▶ 40 apps - Stowaway vs manual analysis
 - ▶ 900 apps - automated analysis
- ▶ 7% false positives rate
- ▶ 35% applications were found to be overprivileged
 - ▶ 56% declare one extra permission
 - ▶ 94% have 4 or fewer extra permissions

Most common unnecessary permissions:

Permission	Usage
ACCESS_NETWORK_STATE	16%
READ_PHONE_STATE	13%
ACCESS_WIFI_STATE	8%
WRITE_EXTERNAL_STORAGE	7%
CALL_PHONE	6%
ACCESS_COARSE_LOCATION	6%
CAMERA	6%
WRITE_SETTINGS	5%
ACCESS_mock_LOCATION	5%
GET_TASKS	5%

- *Usage* - the percentage of applications that request the permission.

- ▶ Confusing permission names
 - ▶ request permissions in pairs when only one is required
- ▶ Deputies - app sends Intent to another app
 - ▶ the deputy app requires the permission
 - ▶ the sender app doesn't need to declare the permission
 - ▶ e.g. `INSTALL_PACKAGES` - Google Play app installs packages
 - ▶ camera, browser, phone dialer

- ▶ Related Methods - getters and setters (read/write permissions)
 - ▶ app uses only getters but declares the WRITE.... permission
- ▶ Copy and Paste - copying incorrect examples
- ▶ Deprecated Permissions
- ▶ Testing Artifacts - used when developing and testing the app
 - ▶ ACCESS_MOCK_LOCATION
- ▶ Declared intentionally - for automatic updates

Introduction

Android Permission System

Stowaway

Keywords

Questions

- ▶ Android
 - ▶ operating system security
 - ▶ permission system
 - ▶ overprivileged
 - ▶ permission map
- ▶ API Calls
 - ▶ Intents
 - ▶ Content providers
 - ▶ Randoop automated testing

- ▶ Stowaway <http://android-permissions.org/>
- ▶ Research on Android permissions:
<http://www.cs.berkeley.edu/~afelt/>
- ▶ *Understanding Android Security*, William Enck, Machigar Ongtang, and Patrick McDaniel IEEE Security & Privacy Magazine, 7(1):50–57, January/February, 2009
- ▶ *Android Permissions: User Attention, Comprehension, and Behavior*, Adrienne Porter Felt et al, Symposium on Usable Privacy and Security (SOUPS) 2012
- ▶ Android Permissions documentation:
<http://developer.android.com/guide/topics/security/permissions.html>

Introduction

Android Permission System

Stowaway

Keywords

Questions