ANDROID SECURITY REFINEMENTS
Lecture 12a

COMPSCI 702
Security for Smart-Devices

Nalin Asanka Gamagedara Arachchilage

Slides from Muhammad **Rizwan** Asghar

March 25, 2021

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

SECURITY REFINEMENTS

- Android's security framework is based on MAC and DAC
- Out of necessity and for convenience, Android offers several security refinements to the basic security model
- These refinements can be considered as exceptions
- Some of these refinements have subtle side effects
  - Which makes the overall security difficult to understand

PUBLIC VS PRIVATE COMPONENTS

- Apps often contain components that other apps should never access
  - For example, an activity returning a user password
- The developer can declare this component private
  - Set the exported attribute to false in the manifest file
- Private components can only be accessed by other components in the same app
- Private components simplify security specification
  - Developers do not need to worry about assignment of permission labels

PUBLIC VS PRIVATE COMPONENTS

- Best practice
  - Always declare the component private to avoid unknowingly access by other components

IMPLICITLY OPEN COMPONENTS

- Developers frequently define intent filters on activities
  - E.g., the system finds an image viewer when an intent is with a VIEW action
- The caller cannot know beforehand what access permission is required
- The developer of the target activity can declare it open by not assigning any access permission to it
  - That is, a public component without any permission

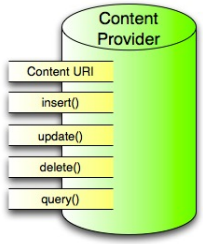
IMPLICITLY OPEN COMPONENTS

- Advantage
  - This enables richer functionality and ease of development
- Issue
  - Any app can have access
  - It can lead to poor security practices
- Best practice
  - Components must be declared open in exceptional cases
  - Consider splitting components to sub-components to specify fine-grained control

BROADCAST INTENT PERMISSIONS

- A broadcast intent is read by all apps
- It can lead to leaking sensitive information
- Using a broadcast intent permission, the developer can protect the intent
- A broadcast intent permission can be declared programmatically
  - sendBroadcast(intent, COMPSCI702.OurPermission)
  - Does the manifest file provide a complete view of the app security?
- Best practice
  - Always use a broadcast intent permission

CONTENT PROVIDER PERMISSIONS



- Recall that content providers provide interfaces for reading (select) or writing (insert, update, and delete) the data
- Instead of using one permission label, Android allows developers to assign both read and write permissions
- Best practice
  - Always define both read and write permissions

SERVICE HOOKS

- If a component has the permission, it can start, stop, or bind the service at anytime
- To specify more flexible and fine-grained access control, Android allows components to invoke the *checkPermission()* method
- This extra check is performed at the code level
- It intermingles code and security policies
- Best practice
  - Use *checkPermission()*
  - Create sub-services

PENDING INTENTS

- Pending Intents delegate actions to another app
  - E.g., passing Pending Intent to other apps enables them to invoke services on behalf of the requesting app
- Pending Intents provide better integration with the third party apps
- Pending Intents enable delegation, which is deviation from the MAC model

URI PERMISSIONS

- Android uses a special content URI to deal with content providers
  - It can also specify a record within a table
- An app that does not have a read permission to access the content provider, it can get access using a URI permission
- The developer can pass a URI in an intent filter
- Like Pending Intents, URI permissions also enable delegation, which is deviation from the MAC model

RESOURCES

- **Chapter 2** of
  Android Security Internals: An In-Depth Guide to Android's Security Architecture
  Elenkov, Nikolay
  First Edition
  No Starch Press 2014
  ISBN:1593275811 9781593275815

- Enck, William, Machigar Ongtang, and Patrick McDaniel
  Understanding Android Security
  IEEE Security & Privacy 1 (2009): 50-57

- **SELinux concepts**
  https://source.android.com/security/selinux/concepts.html

Questions?

Thanks for your attention!