Introduction to Android

● Kernel layer
  ○ Linux
  ○ Binder for IPC
  ○ Low-level resource management

● Middleware layer
  ○ Unique to Android
  ○ Framework
    ■ high-level resource management
  ○ Runtime
    ■ Dalvik Virtual Machine
    ■ Android Runtime (ART)

● Application layer
  ○ System applications
  ○ User applications
Introduction to Android

- Android framework
  - Resources management such as location, telephony, etc
  - Largely in Java
  - Access control enforcement: permissions

```java
@RequiresPermission(anyOf = {ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION})
@NonNull
public Location getLastKnownLocation(@NonNull String provider) {
    checkProvider(provider);
    String packageName = mContext.getPackageName();
    LocationRequest request = LocationRequest.createFromDeprecAtedProvider(
            provider, 0, 0, true);

    try {
        return mService.getLastLocation(request, packageName);
    } catch (RemoteException e) {
        throw e.rethrowFromSystemServer();
    }
}
```
Introduction to Android

- Android framework

```java
/** @hide */
@SystemApi
@RequiresPermission(android.Manifest.permission.MODIFY_PHONE_STATE)
public boolean disableDataConnectivity() {
    try {
        ITelephony telephony = getITelephony();
        if (telephony != null)
            return telephony.disableDataConnectivity();
    } catch (RemoteException e) {
        Log.e(TAG, "Error calling ITelephony#disableDataConnectivity", e);
    }
    return false;
}
```
Introduction to Android

- **Runtime**
  - Where Android applications can be installed and executed
  - Dalvik virtual machine
    - Before Android 5.0
    - Bytecode interpreter
    - Similar to JVM but a register-based machine
    - Has performance issues
  - After Android 5.0
    - Android Runtime (ART)
    - Native code
    - Huge performance boost
Introduction to Android

The life of an APK

Source → ... → Dex File → Resources & Native Code → zip → APK

Resources & Native Code → install

Dex File → quickened dex → Dex File

Dex2oat → dex & native code

Odex File → Dalvik

ELF File → Art

Libraries

Dalvik | Native

JIT

Art | Native
Introduction to Android

- Android applications
  - Resource files
  - Dex files
  - Android Manifest file
Introduction to Android

- Android applications
  - Can be written in Java or C++ (most likely in Java)
  - Java compiled into Dex (Dalvik Executable) bytecode

```java
public boolean offer(E e) {
    checkNotNull(e);
    final ReentrantLock lock = this.lock;
    lock.lock();
    try {
        if (count == items.length)
            return false;
        else {
            enqueue(e);
            return true;
        }
    } finally {
        lock.unlock();
    }
}
```
Introduction to Android

- Android applications
  - Four major components
    - Activity
    - Service
    - Content provider
    - Broadcast receiver

- Intent
  - Used as inter-component signaling
  - Example:
    - Start an activity
    - Query a content provider
Introduction to Android

- **Activity**
  - Each activity is a ‘screen’ in app
  - One app can have multiple activities
  - **Intent** is used to launch an activity
  - Can be invisible/transparent
    - Security consequences!
Introduction to Android

- Activity
  - Lifecycle
  - Multiple entry points
  - No explicit control flow within Android apps
  - Make program analysis harder
Introduction to Android

- **Service Component**
  - Background processing
    - Download a file
    - Play music
  - Multiple interfaces
    - Control: start, stop
    - Method invocation: bind
  - Service lifecycle
    - Similar to activity lifecycle
Introduction to Android

- Permission system
  - Used for access control to sensitive APIs
  - Sensitive APIs:
    - Send text message
    - Retrieve location
    - Access your contacts
    - etc
  - Android apps need to request permissions at installation time

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

- Security:
  - Over-privilege issue
  - Hard to understand
  - Repackaged apps
Android Security Analysis

- Android application analysis
  - Vulnerability analysis
    - Component hijacking vulnerability
    - Information leakage
    - Collusion attacks
    - etc
  - Malware analysis
    - Packing techniques

- Android framework analysis
  - Inconsistent security policy
  - Implicit control flow transitions
Android Security Analysis

- Component hijacking vulnerability
  - Export components
    - Publicly available
    - Can be launched by other components from a different app
    - Accidentally share permissions

```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">
  <application tools:ignore="GoogleAppIndexingWarning">
    <activity
      android:name="com.samples.medium.SecondaryActivity"
      android:exported="true"
      android:screenOrientation="sensorLandscape"/>
  </application>
</manifest>
```
Android Security Analysis

- Component hijacking vulnerability
Android Security Analysis

- Collusion attack
  - Multiple apps work together
  - Communicate via intent
  - Stealthier
Android Security Analysis

- Information leakage
Packing techniques
Packing techniques

Android App → packed App 

encrypted code

decrypt.so

decrypt & load

decrypted code

Loading

Memory
Packing techniques

4.1 MB

After packing

1 KB
Framework analysis

- Framework
  - Sensitive resources protection
  - Even experts can make mistakes
  - Severe consequences
Framework analysis

- **Security protection inconsistency**
  - An app can use either of the two interfaces to update configs
  - Two interfaces enforce different permissions
Summary

- Understand Android system design
  - Uniqueness of Android
  - Framework
  - Android runtime
- Introduce basics of Android applications
  - Four components
  - Permission system
- Present Android security problems
  - Application vulnerabilities
  - Framework issues
  - Packing techniques
Thank you!!

Questions?