Mobile Security Issues
Introduction

- So many cool mobile apps
- Access to web, personal information, social media, etc
- Security problems (not previously envisaged) have resulted
- Examples:
  - Malicious apps can steal your private information (credit card information, etc)
  - Smartphone sensors can leak sensitive information
  - Malware can lock your phone till you pay some money (ransomeware)
- Need deeper understanding of mobile security
Android Security Model
Android Security

● Security goals are to
  ● Protect user data, system resources (hardware, software)
  ● Provide application isolation

● Foundations of Android Security

1. Application Isolation:
   ● Application sandboxing: App 1 cannot interact directly with app 2
   ● Secure inter-process communication

2. Permission Requirement:
   ● System-built and user-defined permissions
   ● Application signing
Recall: Android Software Framework

- Each Android app runs in its own security sandbox (VM, minimizes complete system crashes)
- Android OS multi-user Linux system
- Each app is a different user (assigned unique Linux ID)
- Access control: only process with the app’s user ID can access its files
- Apps talk to each other only via intents, IPC or ContentProviders

Ref: Introduction to Android Programming, Annuzzi, Darcey & Conder
Recall: Android Software Framework

- Android software framework is layered
  - **OS:** Linux kernel, drivers
  - **Apps:** programmed & UI in Java
  - **Libraries:** OpenGL ES (graphics), SQLite (database), etc
- Each layer assumes layer below it is secure
Android Encryption

- Encryption encodes data so that unauthorized party cannot read it

- **Full-disk encryption:** Android 5.0+ provides full filesystem encryption
  - All user data can be encrypted in the kernel
  - User password needed to access files, even to boot device

- **File-based encryption:** Android 7.0+ allows specific files to be encrypted and unlocked independently
iPhone vs Android Encryption

- In earlier Androids, encryption was up to user
- iPhones encrypt automatically: almost all encrypted

Image credit: wall street journal
App Markets
App Markets & Distribution

- Major OS vendors manage their own markets for “certified” apps
  - Android: the Google Play Store
  - iOS: the App Store is the sole source of apps
Google Play App Scanning

- Important for app markets to check security of apps, prevent malware
- Most current markets include some form of scan or verification prior to accepting/certifying an app
  - Typically, static analysis of source code to check for known malware, best practices, app performance, etc.
  - Crowd-sourced reports after approval also useful (e.g. users report suspicious apps)

- Google Play app scanning (called Google Play Protect)
  - Antivirus system scans Google Play for threats, malware
  - New “peer grouping system:
    - similar apps (e.g. all calculators) are grouped on app market.
    - If one app requests more permissions than similar apps, human takes a look
App Markets: Android Vs iOS

● Apple App Store
  ● Highly regulated
  ● All applications are reviewed by human
  ● iOS devices can only obtain apps through here, unless jailbroken

● Google Play (Android Market)
  ● More automated scans
  ● Some applications may be reviewed
  ● Users may also install Android apps from 3rd party marketplaces (e.g. Pandaapp)

● Many malware developers target third-party markets
  ○ Weaker/no restrictions or analysis capabilities
Malware Evolution
Threat Types: Malware, Grayware & Personal Spyware

What’s the difference between??

1. Malware
2. Spyware
3. Grayware
Threat Types: Malware, Grayware & Personal Spyware

- **Malware:**
  - Gains access to a mobile device in order to steal data, damage device, or annoying the user, etc. *Malicious!!*

- **Personal Spyware:**
  - Collects user’s personal information over of time
  - Sends information to app *installer* instead of author
  - E.g. spouse may install personal spyware to get info

- **Grayware:**
  - Collect data on user, but with no intention to harm user
  - E.g. for marketing, user profiling by a company
Growth of Android Malware

Ref: Bochum, Author: Christian Lueg
8,400 new Android malware samples every day
https://www.gdatasoftware.com/blog/2017/04/29712-8-400-new-android-malware-samples-every-day
Mobile Malware Survey *(Felt et al)*
Mobile Malware Study?
A survey of mobile malware in the wild Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner in Proc SPSM 2011

- First major mobile malware study in 2011 by Andrienne Porter Felt et al
  - Previously, studies mostly focused on PC malware
  - 18 – Android
  - 4 – iOS
  - 24 – Symbian (discontinued)
- Analyzed information in databases collected by:
  - information in databases maintained by anti-virus companies
    - E.g., Symantec, F-Secure, Fortiguard, Lookout, and Panda Security
  - Mentions of malware in news sources
- Did not analyze spyware and grayware
Categorized Apps based on Behaviors

- **Novelty and amusement:** Minor damage. E.g.
  - Change user’s wallpaper

- **Selling user information:**
  - Personal information obtained via API calls
    - User’s location, contacts, download + browser history/preferences
  - Information can be sold for advertisement
    - $1.90 to $9.50 per user per month
Categorized Apps based on Behaviors

- **Stealing user credentials:**
  - People use smartphones for shopping, banking, e-mail, and other activities that require passwords and payment information
  
  - Malwares can log keys typed by user (keylogging), scan their documents for username + password

- In 2008, black market price of:
  - Bank account credentials: $10 to $1,000,
  - Credit card numbers: $.10 to $25,
  - E-mail account passwords: $4 to $30
Categorized Apps based on Behaviors

- **Make premium-rate calls and SMS:**
  - Premium rate texts to specific numbers are expensive
  - Malware sends SMS to these numbers set up by attacker
  - Cell carrier (e.g. sprint) bills users
  - Attacker makes money

- **SMS spam:**
  - Used for commercial advertising and phishing
  - Sending spam email is illegal in most countries
  - Attacker uses malware app on user’s phone to send SPAM email
  - Harder to track down senders
Categorized Apps based on Behaviors

- **Search Engine Optimization (SEO):**
  - Malware makes HTTP requests for specific pages to increase its ranking (e.g. on Google)
  - Increases popularity of requested websites

- **Ransomeware**
  - Possess device, e.g. lock screen till money is paid
  - *Kenzero* – Japanese virus included in pornographic games distributed on the P2P network
    - Asked for Name, Address, Company Name for “registration” of software
    - Asked **5800 Yen** (~$60) to delete information from website (Paper information is wrong)
    - About 661 out of 5510 infections actually paid (12%)
Categorization of Malware Behaviors

Table 1: We classify 46 pieces of malware by behavior. Some samples exhibit more than one behavior, and every piece of malware exhibits at least one.
Malware Example: Toll Fraud

Source: Lookout State of Mobile Security 2012

**THE PLAYERS**
- **Users** are your everyday smartphone user.
- **Wireless providers** run the network and send you bills.
- **Aggregators** are "misdemen" for premium SMS transactions, who maintain the technical and service level requirements of each wireless network.
- **Malware** is software that performs malicious actions when installed.
- **Malware writers** are people who create malware.

**HOW**
- **TOLL FRAUD SMS MESSAGES**
- **WORK**

User inadvertently downloads malicious app

Malware sends premium SMS without the user's knowledge

Via Aggregator and Wireless provider, a message is sent to user to confirm the order

User is billed fraudulently

Malware blocks message and confirms charge

Malware writer makes money

* depending on country
Malware Example: Ad Jacking

Source: Lookout State of Mobile Security 2012
Malware Example: App Rating Manipulation

Source: Lookout State of Mobile Security 2012
Ransomware

**Ransomware:** Type of malware that prevents or limits users from accessing their system, by locking smartphone’s screen or by locking the users' files till a ransom is paid.
Application Repackaging

1a) Typical Download

1b) “Direct” Download

1a) Extract Mobile application

2) Add Malware & Repackage Application

3) Republish Application

Official Market

Alternative Market
Malware Detection based on Permissions

- Does malware request more permissions?
- Analyzed permissions of 11 Android malwares

**Findings: Yes!**
- 8 of 11 malware request SMS permission (73%)
  - Only 4% of non-malicious apps ask for this
- Malware 6.18 dangerous permissions
  - 3.46 for Non-malicious apps
- Dangerous permissions: requests for personal info (e.g. contacts), etc

<table>
<thead>
<tr>
<th>Number of Dangerous permissions</th>
<th>Number of non-malicious applications</th>
<th>Number of malicious applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>75 (8%)</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>154 (16%)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>182 (19%)</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>152 (16%)</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>140 (15%)</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>82 (9%)</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>65 (7%)</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>28 (3%)</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>19 (2%)</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>21 (2%)</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>10 (1%)</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>6 (0.6%)</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>7 (0.7%)</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>4 (0.4%)</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>4 (0.4%)</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>2 (0.2%)</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>1 (0.1%)</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>1 (0.1%)</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>1 (0.1%)</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>1 (0.1%)</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>26</td>
<td>1 (0.1%)</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: The number of “Dangerous” Android permissions requested by 11 pieces of malware and 956 non-malicious applications [28].
iOS Malware Review

- iOS generally fewer vulnerabilities (even till date)
  - All 4 pieces of Apple malware were spread through jailbroken devices;
  - not found on App Store
  - Human review more effective but slow!!?
Using Hand Gestures to Curb Mobile Malware (Shrestha et al)
Real user will make certain natural hand gestures when:
- Making phone call
- Taking a picture
- Swiping to use NFC reader

These hand gestures will be missing if malware is requesting these services

Main idea: Check for these gestures to separate malware requests from valid user requests
Sensors used for Gesture Identification

- Gesture Identifier used sensors to detect natural hand movements associated with phone dialing, taking picture, NFC usage
  - **Motion Sensors**: Accelerometer and gyroscope
  - **Position Sensors**: Magnetometer and orientation sensors
  - **Environmental Sensors**: Temperature, pressure and illuminance

<table>
<thead>
<tr>
<th>Type</th>
<th>Sensor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion</td>
<td>Accelerometer (A)</td>
<td>The acceleration force including gravity</td>
</tr>
<tr>
<td>Motion</td>
<td>Gyroscope (Gy)</td>
<td>The rate of rotation</td>
</tr>
<tr>
<td>Motion</td>
<td>Linear Acceleration (LA)</td>
<td>The acceleration force excluding gravity</td>
</tr>
<tr>
<td>Motion</td>
<td>Rotation Vector (R)</td>
<td>The orientation of a device</td>
</tr>
<tr>
<td>Motion</td>
<td>Gravity (G)</td>
<td>The gravity force on the device</td>
</tr>
<tr>
<td>Position</td>
<td>Game Rotation (GR)</td>
<td>Uncalibrated rotation vector</td>
</tr>
<tr>
<td>Position</td>
<td>Magnetic Field (M)</td>
<td>The ambient magnetic field</td>
</tr>
<tr>
<td>Position</td>
<td>Orientation (O)</td>
<td>The device orientation</td>
</tr>
<tr>
<td>Environment</td>
<td>Pressure (P)</td>
<td>The ambient air pressure</td>
</tr>
</tbody>
</table>
System Architecture

- **3 Entities**
  - **Gesture Identifier**: classifier to identify gesture
  - **Gesture Manager**: communicates gets phone movement from motion sensors, provides gesture ID result to permission controller
  - **Permission Controller**: checks for the permission token

**Results**: Generally > 85% accuracy (user gesture detection)
Mobile Ad Vulnerabilities
Ad Services

- App developers make money from apps in 2 main ways:
  - Charge users for apps
  - Getting $$$ from advertisers to include ads in apps

- App maker integrates mobile app library in app

- Mobile ad company serves ads to device
AdMob

- AdMob: Most popular mobile ad company
  - Acquired by Google in 2009
Permissions Requested by Ad Services

- Each has 1 AndroidManifest.xml file
- Total permissions in AndroidManifest.xml
  = permissions requested by app + permissions requested by ad service
Rogue? Ad Services

- Google is careful about permissions requested by AdMob
- Some other mobile ad libraries require permissions to:
  - Access location data, camera, account details, calendar, call logs, browser bookmarks, contact lists, phone information, phone number, SMS, etc
  - Make phone calls, send SMS messages, vibrate
  - Change calendar and contacts

Ref: Unsafe exposure analysis of mobile in-app advertisements
Final Words: Mobile Ad Services

- Many apps use multiple ad services
  - Angry Birds app (a game) includes 7+ ad services

- One version of the Dictionary.com app requests permissions to monitor phone calls and access location
Run-Time Permissions Changed in Marshmallow (Android 6.0)

- “Normal” permissions don’t require user consent
  - Normal permissions can do very little to harm app
  - E.g. change timezone
  - Automatically granted
  - Can be used freely by ad networks

- Run-time permissions required for “more dangerous” access
- **Dangerous?** contacts, etc
Android Analysis Tools
Analyzing Android Apps

- Analysis tools give attacker more information about Android app

  - **Source code recovery:** generate app source code from executable

  - **Static analysis (binaries or source code):** Understand app design without running it.
    - Examine application logic, APIs used

  - **Dynamic analysis:** Observe how app executes
    - App memory usage, network usage, response time, performance, etc

- Many available (open source?) tools for all of the above!
Android Analysis Tools

- APKinspector
- Androguard
- AndroBugs
- Qark
- Epicc / IC3
- FlowDroid
- DidFail
- DroidBox
- MobSF

- Scary!!
Android Pay using NFC
Android Pay

- Google Wallet → Android Pay (Sept 2015 initial release)
- **Vision:** Use smartphone to pay in stores
- E.g. Pay for donuts at Dunkin Donuts
- Easier way to track expenses, get rewards
  - Integrates with financial apps (banking, personal finance, etc)
How Android Pay Works

- First need to download Android Pay app, add credit cards

1. Download
2. Add
3. Pay

- To pay, place smartphone near Android pay terminal
Mobile Pay Uses NFC

- Mobile payment (e.g. Android Pay) typically uses NFC for transaction
- NFC: Near Field Communication: short-range, low-rate wireless
- Enables communication between devices in close proximity
- Utilized by many smartphone mobile pay systems (e.g. Google Pay)
- E.g. pay at Dunkin donuts
Wireless Comparison

- NFC: Short range, low bitrate
Why use NFC?

- Proximity makes it easier to verify payee

- **Convenient:** store all credentials inside the phone

- Integrates with other mobile services: eBooks, music downloads, barcodes, etc. (easier payments)
Types of NFC Devices

- **Active Device:**
  - Can read targets information and also send information to target
  - **2-way** communication possible
  - E.g. 2 smartphones

- **Passive Device:**
  - Information on passive device can only be read.
  - Cannot initiate communication
  - E.g. NFC tag
NFC Modes of Interaction

● **Reader/Writer:**
  ○ Active NFC device reads/writes from/to passive NFC tag (One way)

● **Peer-to-Peer:**
  ○ Active NFC devices interact with each other bi-directionally
  ○ Take turns being active vs passive

● **Card Emulation:**
  ○ An NFC device emulates a passive NFC tag that is read by an active NFC device
NFC Security / Threats

- NFC has similar threats as other wireless communications
  - Eavesdropping
  - Data corruption / modification / insertion
  - Man-in-the-middle attacks

- Eavesdropping: Another device listening to transaction
  - NFC itself provides no explicit protection against eavesdropping
  - Active-vs-Passive:
    - Harder to eavesdrop on passive exchange
    - Mainly because of shorter range (<1m passive, <10m active),
Data Modification & Injection

- Attacker modifies bits in flight based on standardized encoding, e.g., flip 0s to 1s

**Data Injection:**
- Attacker responds faster than intended target
- Possible defenses:
  - Secure handshake w/ verifiable response

**MitM is difficult in NFC due to:**
- Close proximity requirement (MitM needs to be closer than tag)
- Attacker can use sheet of Aluminum to block legitimate sender