Computer Security

13. Mobile Device Security

Paul Krzyzanowski
Rutgers University
Spring 2017
Mobile Devices: Users

• Users don't think of phones as computers
  – Social engineering may work more easily on phones

• Small form factor
  – Users may miss security indicators (such as EV cert indicator)
  – Easy to lose/steal a device

• Users tend to pick bad PINs/passwords

• Users may grant app permission requests without thinking
Mobile Devices: Interfaces

• Phones have lots of sensors
  – GSM – Wi-Fi – Bluetooth – GPS – NFC – Microphone
  – Camera – 6-axis Gyroscope and Accelerometer – Barometer

• Sensors enable attackers to monitor the world around you
  – Where you are & whether you are moving
  – Conversations
  – Video
  – Sensing vibrations due to neighboring keyboard activity led to a word recovery rate of 80%
Mobile Devices: Apps

• Lots of apps
  – 2.8 million Android apps and 2.2 million iOS apps

• Most written by untrusted parties
  – We'd be wary of downloading these on our PCs
  – Rely on
    • Testing & approval by Google (automated) and Apple (automated + manual)
    • Sandboxing
    • Explicit granting of permissions for resource access

• Apps often ask for more permissions than they use
  – Most users ignore permission screens

• Most apps do not get security updates
Mobile Devices: Platform

• Mobile phones are comparable to desktop systems in complexity
  – They will have bugs

• Single user environment

• Malicious apps may be able to get root privileges
  – Attacker can install rootkits, enabling long-term control while concealing their presence
Threats

- Privacy
  - Data leakage
  - Identifier leakage
  - Location privacy
  - Microphone/camera access

- Security
  - Phishing
  - Malware
  - Malicious Android intents
  - Broad access to resources (more than the app needs)
## OWASP Top 10 Mobile Risks – 2016

**OWASP = Open Web Application Security Project**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>Improper Platform Usage</td>
</tr>
<tr>
<td>M2</td>
<td>Insecure Data Storage</td>
</tr>
<tr>
<td>M3</td>
<td>Insecure Communication</td>
</tr>
<tr>
<td>M4</td>
<td>Insecure Authentication</td>
</tr>
<tr>
<td>M5</td>
<td>Insufficient Cryptography</td>
</tr>
<tr>
<td>M6</td>
<td>Insecure Authorization</td>
</tr>
<tr>
<td>M7</td>
<td>Client Code Quality</td>
</tr>
<tr>
<td>M8</td>
<td>Code Tampering</td>
</tr>
<tr>
<td>M9</td>
<td>Reverse Engineering</td>
</tr>
<tr>
<td>M10</td>
<td>Extraneous Functionality</td>
</tr>
</tbody>
</table>

https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks
Sample iOS malware

• May 2015: "Unicode of Death"
  – Single string in a text message could crash an iPhone

• 2015: XcodeGhost: affected over 4000 apps
  – Infected Xcode developer software hosted on the Baidu file sharing service
  – Developers who downloaded this version of Xcode would create apps with malware
    • Remote control via commands from a command web server
    • Send information: time, app's name/ID, network time
    • Ability to hijack apps that support iOS's Inter-App Communication URL mechanism
      – Whatsapp, Facebook, iTunes
    • Access clipboard
Sample Android malware

• 2016: **HummingBad** – affected over 10 million devices
  – Developed by a Chinese advertising company
  – Can take control of devices, forcing users to click ads and download apps

• 2016: **Stagefright** – latest version called Metaphor
  – Tricks user into visiting a hacker’s web page
  – Page contains a malicious multimedia file that infects the phone
  – Hacker can take control of the device to
    • Gain access to personal information
    • Copy data
    • Use microphone & camera
Android & iOS

Pegasus espionage app
- 2016: iOS espionage found infecting phone of a political dissident in the UAE
- 2017: Companion app on Android

"example of the common feature-set that we see from nation states and nation state-like groups"

Functions include
- Keylogging
- Screenshot capture
- Live audio & video capture
- Remote control of the malware via SMS
- Messaging data exfiltration from common apps, including WhatsApp, Skype, Facebook, Twitter, Viber, and Kakao
- Browser history, email, contacts, and text message exfiltration

App can self-destruct when it's at risk of being discovered or compromised

Android Security
Android Security Features

• All app code runs under Dalvik (a variant of a JVM)
  – But native code was needed

• Isolation
  – Android based on Linux, which is multi-user
  – Each app normally runs as a different user

• Communication between apps
  – Related apps may share the same Linux user ID
    • Can share files and may share the same Linux process & Dalvik VM
  – Communication via app framework
    • "Intents": message with {action, data to act on, component to handle the intent}

• Battery life
  – Developers must conserve power
  – Apps store state so they can be stopped and restarted
    • Helps with DoS
App Sandbox

• Each app runs with its own UID in its own Dalvik virtual machine
  – CPU protection, memory protection
  – Authenticated communication with UNIX domain sockets

• Permission model
  – Apps announce permission requirements
  – **Whitelist access**: user grants access
  – All questions asked at install time

• Exploit prevention
  – Stack canaries
  – Some heap overflow protections (check backward & forward pointers)
  – ASLR
Some security issues

• **Intents**
  – Sender can verify recipient has a permission by specifying a permission with the intent method call
  – Receivers have to handle malicious intents

• **Permissions re-delegation**
  – An app, without a permission, may gain privileges through another app
  – If a public component does not explicitly have an access permission listed in its manifest definition, Android permits any app to access it
  – Example
    • Power Control Widget (a default Android widget) – allows 3rd party apps to change protected system settings without requesting permissions
    • Malicious app can send a fake Intent to the Power Control Widget, simulating the pressure of the widget button to switch settings
Some security issues

• Permissions avoidance
  – By default, all apps have access to read from external storage
    • Lots of apps store data in external storage without protection
  – Android intents allow opening some system apps without requiring permissions
    • Camera, SMS, contact list, browser
    • Opening a browser via an intent can be dangerous since it enables
      – Data transmission, receiving remote commands, downloading files
iOS Security
iOS App Security

• Runtime protection
  – System resources & kernel shielded from user apps
  – App sandbox restricts access to other app's data & resources
    • Each app has its own sandbox directory
    • Limit access to files, preferences, network, other resources
  – Inter-app communication only through iOS APIs
  – Code generation prevented – memory pages cannot be made executable

• Mandatory code signing
  – Must be signed using an Apple Developer certificate

• App data protection
  – Apps can use built-in hardware encryption
iOS File Encryption

- File contents are encrypted with a per-file key
- Per-file key is encrypted with a class key & stored in a file's metadata
- File's metadata is encrypted with the file system key
- Hardware AES engine encrypts/decrypts the file as it is written/read on flash memory
Masque Attack

iOS app can be installed using enterprise ad-hoc provisioning

• Can replace genuine app from App Store if they have the same bundle identifier

• iOS didn't enforce matching certificates for apps with the same bundle identifier

• But … user gets a warning "untrusted app developer"
Web apps

• Both iOS & Android support web apps
  – Fully functional web browser incorporated as an app to a specific site

• This makes web client issues relevant
  – Loading untrusted content
  – Leaking URLs to foreign apps
"a malicious webpage could use iPhone sensors to detect a passcode.

The technique was so accurate that the team had a 100% success rate at working out 4-digit PINs within five attempt …

A neural network was used to identify correlations between motion sensor data and tapped PINs, and a browser JavaScript exploit was used to run the malware.

https://9to5mac.com/2017/04/12/iphone-motion-sensors-detect-passcodes-pins/
Hardware aids to security: ARM TrustZone

- Hardware-separated secure & non-secure worlds
  - Non-secure world cannot access secure resources directly
- Software resides in the secure or non-secure world
- Processor executes in one world at any given time
- Each world has its own OS & applications
- Applications
  - Secure key management & key generation
  - Secure boot, digital rights management, secure payment

http://www.arm.com/products/security-on-arm/trustzone
Hardware aids to security

**Apple Secure Enclave**: Apple's customized TrustZone

- Coprocessor in Apple A7 and later processors
- Runs its own OS (modified L4 microkernel)
- Has its own secure boot & custom software update
- Provides
  - All cryptographic operations for data protection & key management
  - Random number generation
  - Secure key store, including Touch ID (fingerprint) data
- Maintains integrity of data protection even if kernel has been compromised
- Uses encrypted memory
- Communicates with the main processor by an interrupt-driven mailbox and shared memory buffers
Summary

• **Mobile devices are attractive targets**
  – Huge adoption, simple app installation by users, always with the user

• **Android security model**
  – Isolated processes with separate UID and separate VM
  – Java code (mostly): managed, no buffer overflows
  – Permission model & communication via intents

• **iOS security model**
  – App sandbox based on file isolation
  – File encryption
  – Apps written in Objective C and Swift
  – Vendor-signed code, closed marketplace (App Store only)

• **Protection efforts have generally been good**
  – Usually better than on normal computers
  … but often not good enough!
The end