Computer Security

15. Mobile Device Security

Paul Krzyzanowski

Rutgers University

Spring 2019

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 an 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
  - Cameras – 6-axis Gyroscope and Accelerometer – Barometer
  - Magnetometer (compass) – Ambient light
- 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
  - 2.6 million Android apps and 2.1 million iOS apps*
- Most written by untrusted parties
  - We'd be wary of downloading these on our PCs
  - With mobile apps we rely on
    - Testing & approval by Google (automated) and Apple (automated + manual)
    - App 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

*As of the third quarter of 2018

Mobile Devices: Platform

- Mobile phones are comparable to desktop systems in complexity
  - The OS & libraries 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

https://www.skycure.com/pr/report-finds-rate-ios-malware-increasing-faster-android-malware-iphone-ten-year-anniversary/
https://www.theregister.co.uk/2017/07/20/ios_security_skycure/

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

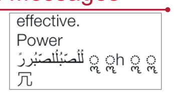
OWASP = Open Web Application Security Project

M1	Improper Platform Usage
M2	Insecure Data Storage
M3	Insecure Communication
M4	Insecure Authentication
M5	Insufficient Cryptography
M6	Insecure Authorization
M7	Client Code Quality
M8	Code Tampering
M9	Reverse Engineering
M10	Extraneous Functionality

The 2016 list is the latest as of April 2019

https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks
https://www.apriorit.com/dev-blog/435-owasp-mobile-top-10-2017#p1

Sample iOS input validation bugs in Messages



- **May 2015**: "Unicode of Death"
  - Single string in a text message could crash an iPhone
- **Again in Jan 2018**: "*ChaiOS*"
  - Receiving a link causes the messages app to go blank & crash instantly after opening; possible crashes
  - Malformatted characters in the message causes the Webkit HTML engine to crash.
  - The target file contains multiple such characters, so CoreText spends a lot of CPU time trying to match fonts for them
- **Again in Feb 2018**
  - A specific character in an Indian language (Telugu) causes Apple's iOS Springboard to crash when the message is received
  - Messages will no longer open as it fails to load the character
  - Affects third-party messaging apps too

- **Again in May 2018:** Black dot of death
  - Thousands-character-long string of invisible Unicode text causes iMessages to crash whenever the user launches the app

Sample iOS malware

- 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 iOS malware

- September 2018: bad CSS content crashes & restarts iOS
- WebKit rendering engine bug
  - Restarts iOS
- Exploited by loading an HTML page with the special CSS code
  - CSS tries to apply a backdrop filter to a series of nested page segments (<div> <div> …)
  - Weakness in the `-webkit-backdrop-filter` CSS property
    - Uses 3D acceleration to process the elements
    - Consumes all graphic resources and freezes or kernel panics the OS

Sample iOS malware: VoiceOver bug

- Lock screen bypass
  - Attacker calls victim's phone
  - Attacker then taps 'answer by SMS' and selects 'personalize/custom' option
  - Attacker then asks Siri to turn on VoiceOver
  - Then, at the same time:
    - select camera icon
    - double-tap the screen
    - Invoke Siri through side buttons
- Attack enables access to photos
- Attacker needs physical access to the device

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

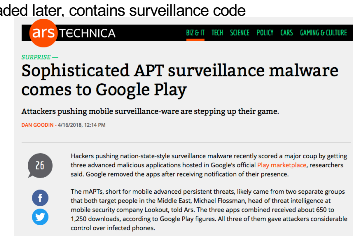
https://arstechnica.com/security/2017/04/found-quite-possibly-the-most-sophisticated-android-espionage-app-ever/

Android, iOS, Linux, others: Bluetooth

- Blueborne set of vulnerabilities
  - *Affects most devices using Bluetooth*
  - Can be used to hijack & control a device
  - Affects Android, Windows, Linux, iOS <10.0
- What it does
  - Poses as a device that wants to discover and connect over Bluetooth
  - Attacks portions of the software that establishes a Bluetooth connection
  - Hijacks the Bluetooth stack
  - Does this before the user needs to take any action
- Discovered in 2017
  - Affected practically every smart device in the world
  - Patched but two billion devices still estimated to be vulnerable

Mobile Advanced Persistent Threats

- 4/16/2018 report: Mobile Advanced Persistent Threats (mAPT)
  - Three mAPT apps were found in Google's Play marketplace
  - Target people in the Middle East
- Malicious functionality not part of initial downloaded version
  - Second stage, downloaded later, contains surveillance code



ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

SURPRISE —

Sophisticated APT surveillance malware comes to Google Play

Attackers pushing mobile surveillance-ware are stepping up their game.

DAN GOODIN - 4/16/2018, 12:14 PM

26

Hackers pushing nation-state-style surveillance malware recently scored a major coup by getting three advanced malicious applications hosted in Google's official Play marketplace, researchers said. Google removed the apps after receiving notification of their presence.

The mAPTs, short for mobile advanced persistent threats, likely came from two separate groups that both target people in the Middle East, Michael Flossman, head of threat intelligence at mobile security company Lookout, told Ars. The three apps combined received about 650 to 1,250 downloads, according to Google Play figures. All three of them gave attackers considerable control over infected phones.

Mobile Advanced Persistent Threats

- Upload attacker-specified files to command and control servers
- Record surrounding audio, calls, and video
- Retrieve account information such as email addresses
- Retrieve contacts
- Remove copies of itself if any additional APKs are downloaded to external storage
- Call an attacker-specified number
- Uninstall apps
- Hide its icon
- Retrieve list of files on external storage
- Encrypt some exfiltrated data
- Obtain a list of installed applications
- Get device metadata
- Inspect itself to get a list of launchable activities
- Retrieve PDF, txt, doc, xls, xlsx, ppt, and pptx files found in external storage
- Send SMS messages
- Retrieve text messages
- Track device location
- Handle limited attacker commands via out-of-band text messages
- Check if a device is rooted
- If running on a Huawei device, it will attempt to add itself to the protected list of apps able to run with the screen off

https://arstechnica.com/information-technology/2018/04/malicious-apps-in-google-play-gave-attackers-considerable-control-of-phones/

Android Security

Android Security Features

- All app code runs under Dalvik (a variant of a JVM)
  - But native code was needed too
- **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}
    - Apps must be granted explicit permission to access input devices & personal data
      - Camera, microphone, GPS

Android Security Features

- Signed applications
  - Apps must be signed. Signature validated by Google Play & package manager on the device
- App verification
  - Users can enable "verify apps" to have apps evaluated by an app verifier prior to installation
  - Will scan app against Google's database of apps
- 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
  - No-execute (NX) hardware protection to prevent code execution on the heap or stack

Some security issues

- Inter-app communication: **intents**
  - Messaging system used to request actions from another app component
    - Intents are used to invoke system services as well as 3rd party apps
    - Examples: *add a calendar event, set an alarm, take a photo & return it, view a contact, add a contact, show a location on a map, retrieve a file, initiate a phone call*
  - 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

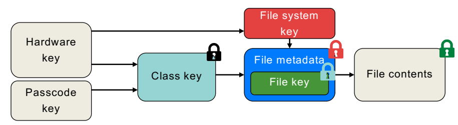
iOS Security

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

Reading iOS files

- Metadata decrypted with the **file system key**
  - File system key = random key created when iOS is installed
- This reveals the encrypted **per-file key** & identifies which **class** protects it (class = user or group)
- The per-file key is unwrapped with the class key
  - AES engine decrypts file as it is read from flash memory
  - Per-extent keys: portions of a file can be given different keys



Masque Attack

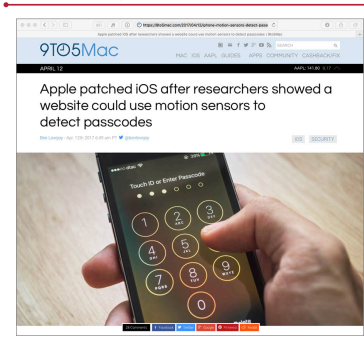
iOS app can be installed using enterprise ad-hoc provisioning

- Designed to bypass the App Store & allow developers to install apps for deployment within an enterprise
- 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
- The user gets a warning "untrusted app developer"
  - But users often ignore these.

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
  - XSS attacks, …

Web page access to sensors



"a malicious webpage could use iPhone sensors to detect a passcode.

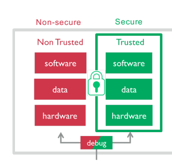
The technique was so accurate that the team had a 100% success rate at working out 4-digit PINs within five attempt …

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-isolated secure & non-secure worlds
  - Non-secure world cannot access secure resources directly
  - Each CPU core has two virtual cores: secure & non-secure
- Processor executes in one world at any given time
- Each world has its own OS & applications
- Software resides in the secure or non-secure world
  - Non-secure (non-trusted) applications cannot access secure resources directly



- 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: Similar to TrustZone but a *separate processor*

- 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
  - Neural network for Face ID
- 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, but also native): 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 far better than on normal computers
    … but often not good enough!

The end