MOBILE DEVICE SECURITY

Mani Akella
MSIA, CISSP, CISM
February 14 2017

AGENDA

- The Problem
- Existing mobility solution
- Developing the new solution
- Explanation of solution

INTRODUCTION

"For many professionals, the mobile phone has become a mobile office,"

- Mike Jones, Symantec

"There is no question that mobile security will eventually equal – if not surpass – PC security as a threat to IT departments,"

- Denise Culver, Heavy Reading Mobile Networks Insider

"By 2014, 90 per cent of firms will support corporate applications on personal devices"

- The Economic Times, Nov. 30th, 2010

PROBLEM STATEMENT

With the explosive growth of smartphones, tablets and mobile devices, companies must find a means of providing access to their internal systems and information to their mobile workforce securely and seamlessly.

SMARTPHONES AND MOBILE DEVICES

What would we do without them?

- Phone service and text messaging
- WiFi and cellular Internet capabilities
- Document storage and productivity capabilities
- Different from computers:
  - Less likely to have up-to-date software and anti-virus software installed
  - Size
  - Functionality

COMMON USES

- Reading work and personal email
- Scheduling appointments & reminders
- Accessing social Websites
- Listening to music and watching videos
- Playing online games
- Online shopping, banking and bill paying

SMARTPHONE RISKS

- Increase mobility → Increased exposure
- Easily lost or stolen
  - device, content, identity
- Susceptible to threats and attacks
  - App-based, Web-based, SMS/Text message-based

KEY MOBILE DEVICE SECURITY CONCERNS

- Confidentiality
  - Commercial Data
    - Ex: Financial, IP, etc.
  - Personal Data
    - Ex: Customer, Employee records, PCI, etc.
    - User Personal Data
  - Diplomatic cables
- Accessibility
  - Resource uptime
  - High Availability / Recoverability
  - Archive

Maintain device flexibility while protecting against security risks

CURRENT NEEDS OF THE BUSINESS AND SOLUTION APPROACH

- Business users today are more mobile than ever before and are looking to access the enterprise from multiple devices:
  - Apple iOS
  - Android
  - Blackberry
  - Windows Mobile
- Users today are more technically skilled than before and are unfortunately able to develop "Business Managed Solutions" which may not meet the security requirements of the enterprise

High Level Requirements & Solution Approach

- Must securely support users on the 4 identified leading mobile platforms
- Must leverage the significant existing Exchange and Blackberry investment

The answer – A Mobile Device Management (MDM) Solution

BEST SECURITY PRACTICES

1. Pass**word** protect
   Pass**code** protect
   Pass **swipe** protect?

BEST SECURITY PRACTICES CONT.

2. Install Security Software
   - Anti-virus and anti-malware available for mobile devices
3. Keep your apps up-to-date
4. Install a phone finder app
5. Enroll in a backup program
6. Set device to wipe contents after specified number of failed login attempts

BEST SECURITY PRACTICES CONT.

7. When installing apps
   - Take time to read the small print
     - What information does the app require access to?
   - Where are you downloading the app from?
     - Is it the app store location set by default on the phone?

BEST SECURITY PRACTICES CONT.

8. Know where your device is at all times
9. Be mindful of how you use your device
   - Follow same guidelines as you do for your computer
   - Double check URLs for accuracy
   - Don't open suspicious links
   - Make sure the Website is secure before giving any personal data

BEST SECURITY PRACTICES CONT.

10. Limit your activities when using public WiFi
11. Your cellular network connection is more secure than WiFi
12. Check URL's before making a purchase https:// is secure; http:// is not

LOST OR STOLEN?

- Treat as if your purse or wallet
  - File a report with law enforcement
  - Contact your service vendor to cancel your service and report your device missing
  - If you have a backup/wipe program, contact your vendor to have them wipe the device

DEVELOPING THE SOLUTION

SOLUTION REQUIREMENTS

- MS Exchange
  - Exchange – current version
  - ActiveSync (EAS) enabled
  - Enterprise Certificate services / certificate based authentication
- Mobile Device support
  - Support latest Mobile OS's
  - Employee-provided device
  - Support for VPN, Wi-Fi, ActiveSync and encryption
  - Centralized IT management & control
  - Support for common file attachments

SOLUTION REQUIREMENTS (CONT'D)

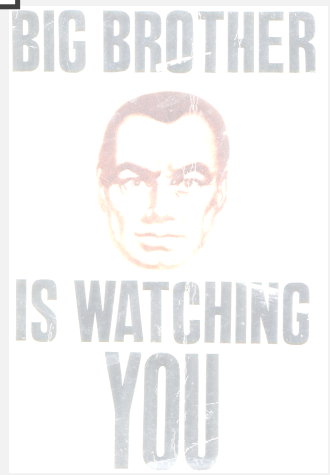
- Security
  - All devices should be enrolled into corporate network
  - Provisioning of mobile devices should be secure
  - Security policies should be targeted to right groups/employees
  - Restriction of some/all mobile applications
  - Complex/multi-character passwords required
  - Updates of mobile OS required
  - Encryption of all forms of corporate data
  - Tracking and inventory of all devices
  - Access control over corporate email system
  - Sanction and disconnect modified devices or rouge device
  - Selective/full remote wipe of device

BUSINESS/LEGAL CONSEQUENCES

- Financial Liability
  - May be required to pay stipend for device/usage
    - Additionally corporate data plans apply in some instances
  - Employee may be taxed for fringe benefit
  - Nonexempt employees create issues
- Legal Liability
  - Evidence of illegal activity must not go unreported
  - Archiving may be required

CONSEQUENCES TO PRIVACY

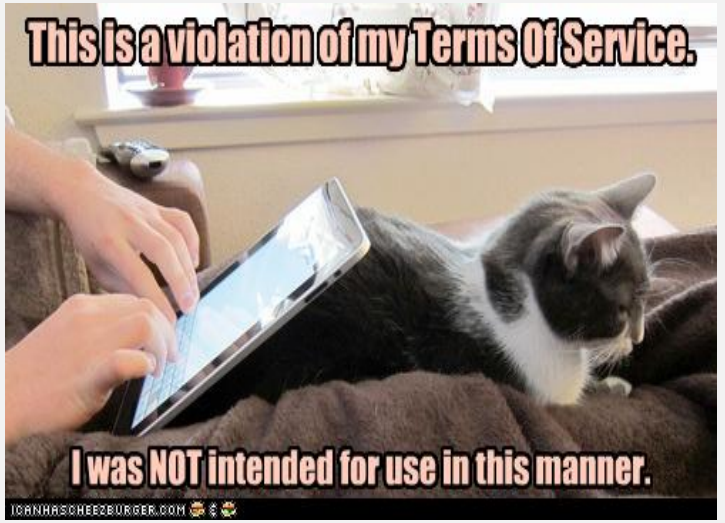
- While some employees will only need access to PIM-data, many will need full device management.
- In these cases, all data must be subject to review and/or archive by the company
  - Email, SMS/MMS, IM, music, etc.
- All activity (applications, browser, peripheral control, etc.) must be subject to audit and control at any time.
- How to handle all of this??



EDUCATION!

- Most people will agree to any ToS without second thoughts.
- Acceptance of the restrictions rely completely on employees' understanding them
- Rewards are worth the

risks …



CONSEQUENCES

- Despite shared liability, employee-provided cell phones for business purposes are extremely popular.
  - Conveniences for employee
  - Savings for employer
- Trend will continue

WHAT ABOUT THE AVERAGE NON-CORPORATE USER?

- We share networks and application data
- Curiosity (ALWAYS) killed the cat !
- Will we learn though?
- How do we prevent the average non-technical user from making a security boo-boo?
- We forget easily
- And forgetfulness has consequences

SMARTPHONE NABBING & INFECTION

- Difficult to determine
- Decreased performance
  - Slow operation and decreased function
- Random action
  - Phone powers on by itself
  - Applications open on their own
  - Items are downloaded without permission
- Phone log shows calls you didn't make
- Emails are sent to addresses you don't recognize

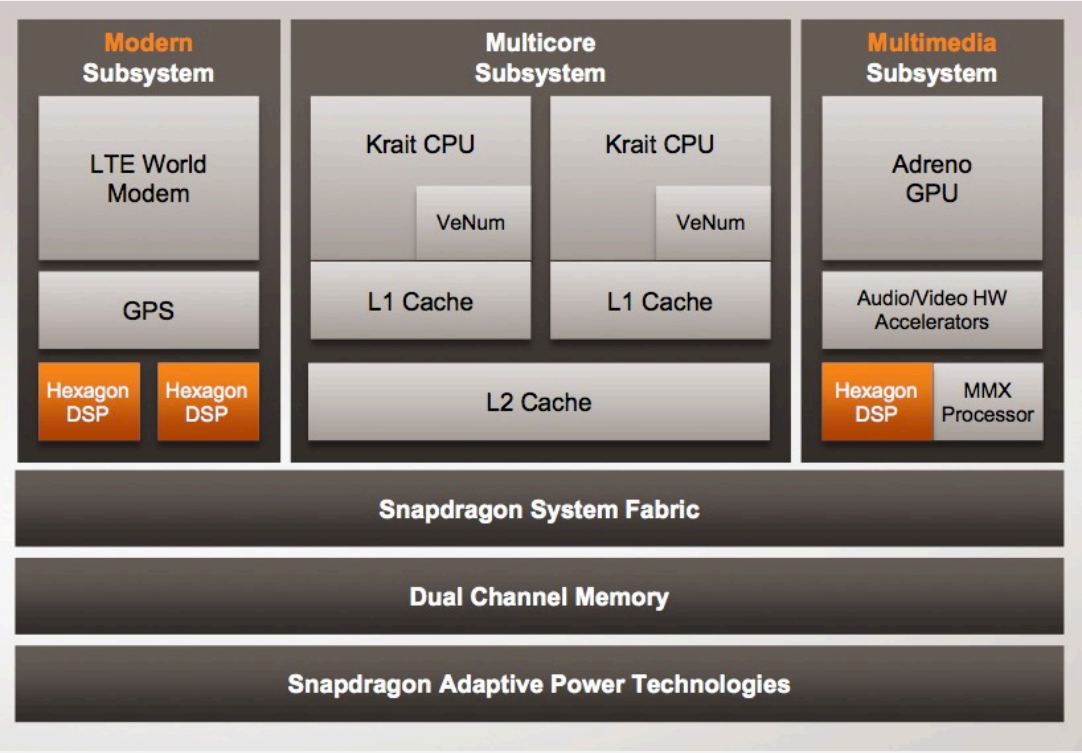
KEY POINTS

1. Password/Passcode protect your device
2. Lock your device
3. Use anti-virus software
4. Sync/back up your data
5. Install a phone finder app

GOALS

- What's different about mobile security?
  - Non-homogeneous hardware architecture
  - Device capabilities
  - Attacker goals
  - Software ecosystems

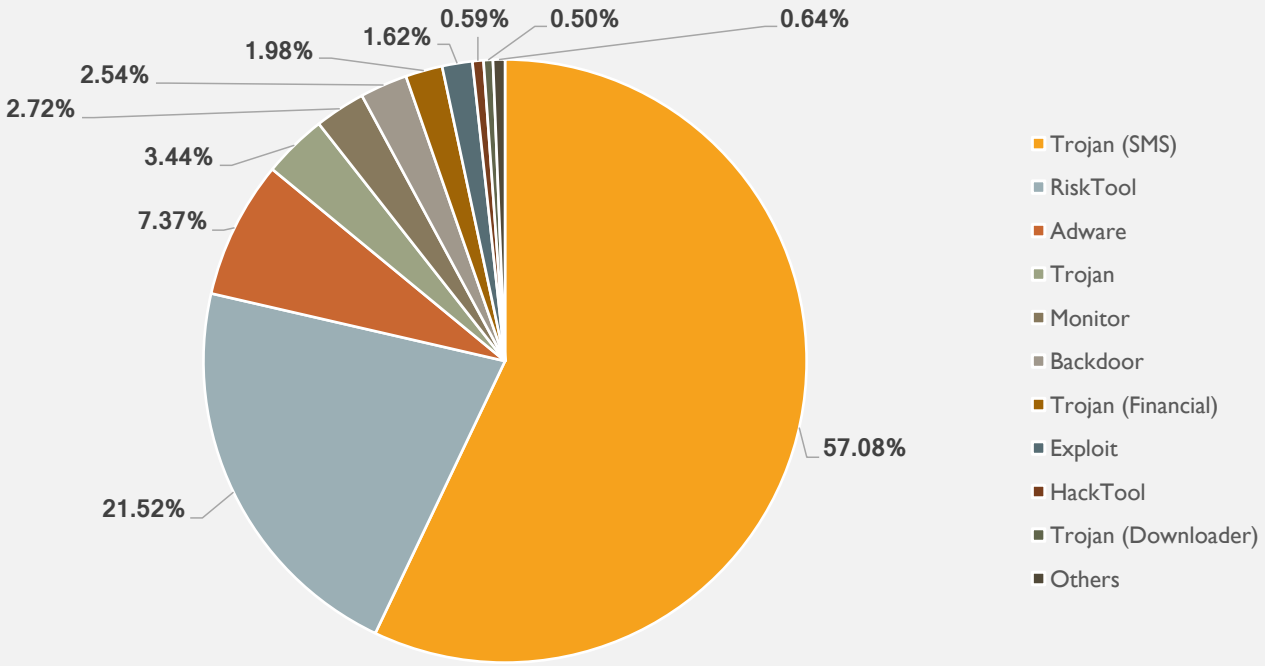
PROCESSOR ARCHITECTURE



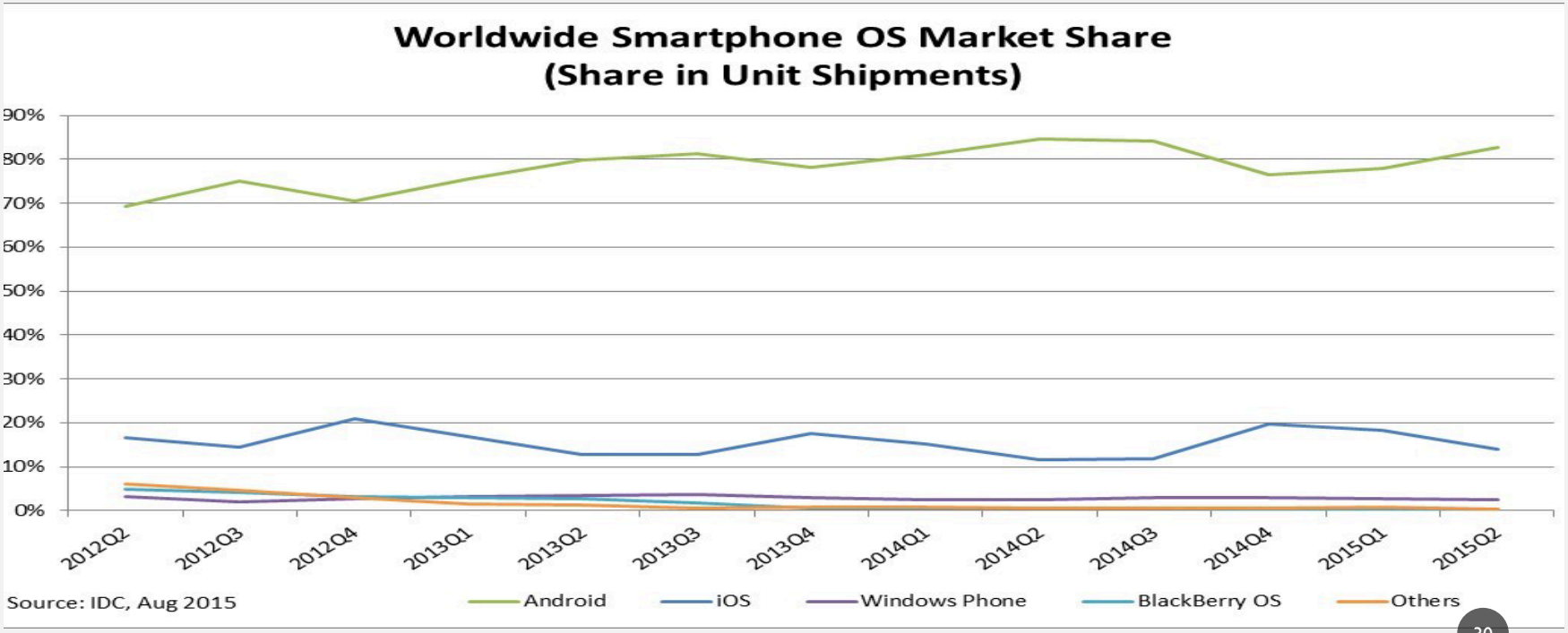
DEVICE CAPABILITIES

- Connectivity
  - GPS
  - Cellular Network (GSM/CDMA)
  - WiFi
  - NFC
- Sensors
  - Accelerometer
  - Gyroscope
  - Ambient Light
  - Compass
  - Barometer
  - Fingerprint sensor
- Battery-powered

MALWARE TYPES



OPERATING SYSTEMS

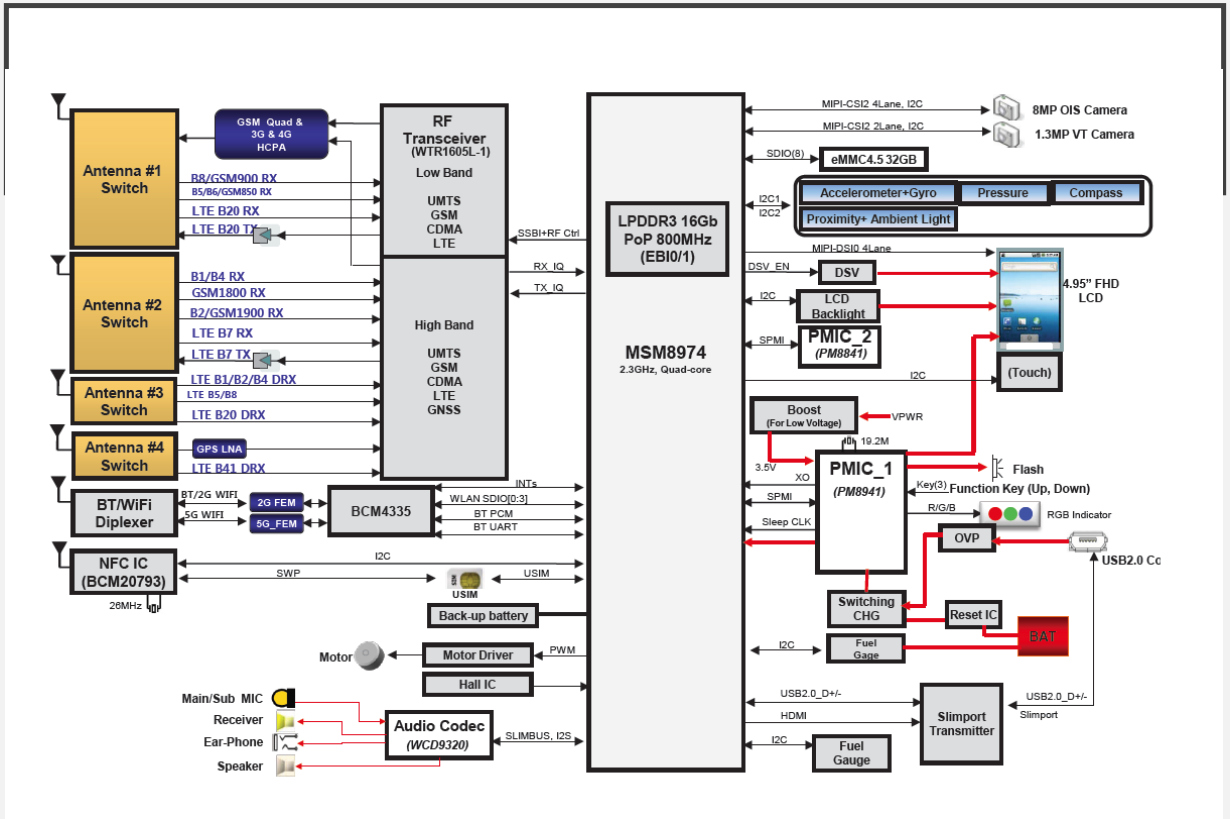


SOFTWARE ECOSYSTEM

- Resource-limited devices
  - Compute
  - Power
- Event-driven programming
  - No `main()` method
  - State transitions via callbacks
- Well-defined interfaces
  - Application lifecycle
  - Access to user data
- Centralized software distribution
  - Can only download applications from a single source
  - Vendor takes responsibility for filtering content

OVERVIEW

- **Architectural complexity**
  - New attack vectors
- Mobile operating systems
  - Operating system safety protections
  - Software development and distribution model
- Common problems with real-world software
  - Cryptographic misuse
  - Personal information leakage
- Current research techniques


Antenna #1 Switch
Antenna #2 Switch
Antenna #3 Switch
Antenna #4 Switch
BT/WiFi Diplexer
NFC IC (BCM20793)
26MHz
GSM Quad & 3G & 4G HCPA
GPS LNA
B8/GSM900 RX
B5/B6/GSM850 RX
LTE B20 RX
LTE B20 TX
B1/B4 RX
GSM1800 RX
B2/GSM1900 RX
LTE B7 RX
LTE B7 TX
LTE B1/B2/B4 DRX
LTE B5/B8
LTE B20 DRX
LTE B41 DRX
BT/2G WIFI
5G WIFI
2G FEM
5G_FEM
BCM4335
I2C
SWP
USIM
RF Transceiver (WTR1605L-1)
Low Band
UMTS GSM CDMA LTE
High Band
UMTS GSM CDMA LTE GNSS
SSBI+RF Ctrl
RX_IQ
TX_IQ
INTs
WLAN SDIO[0:3]
BT PCM
BT UART
Back-up battery
Motor
Motor Driver
PWM
Hall IC
Main/Sub MIC
Receiver
Ear-Phone
Speaker
Audio Codec (WCD9320)
SLIMBUS, I2S
LPDDR3 16Gb PoP 800MHz (EBI0/1)
MSM8974
2.3GHz, Quad-core
MIPI-CSI2 4Lane, I2C
MIPI-CSI2 2Lane, I2C
8MP OIS Camera
1.3MP VT Camera
SDIO(8)
eMMC4.5 32GB
I2C1
I2C2
Accelerometer+Gyro
Pressure
Compass
Proximity+ Ambient Light
MIPI-DSI0 4Lane
DSV_EN
DSV
I2C
LCD Backlight
SPMI
PMIC_2 (PM8841)
4.95" FHD LCD
(Touch)
Boost (For Low Voltage)
VPWR
19.2M
3.5V
XO
PMIC_1 (PM8941)
Flash
Key(3)
Function Key (Up, Down)
R/G/B
RGB Indicator
Sleep CLK
OVP
USB2.0 Cc
Switching CHG
Reset IC
BAT
Fuel Gage
USB2.0_D+/-
HDMI
Slimport Transmitter
Slimport
Fuel Gauge

BLOCK DIAGRAM (BASEBAND)

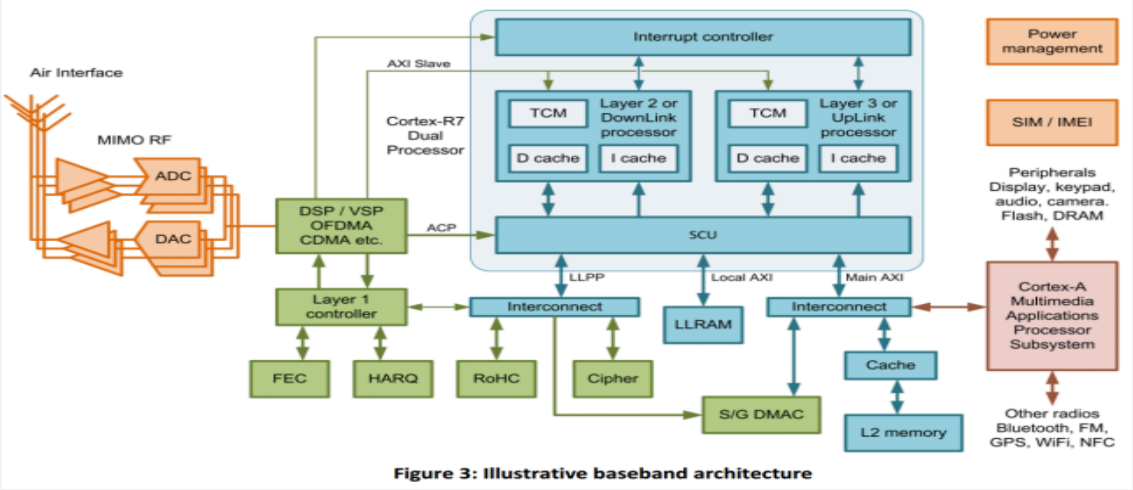
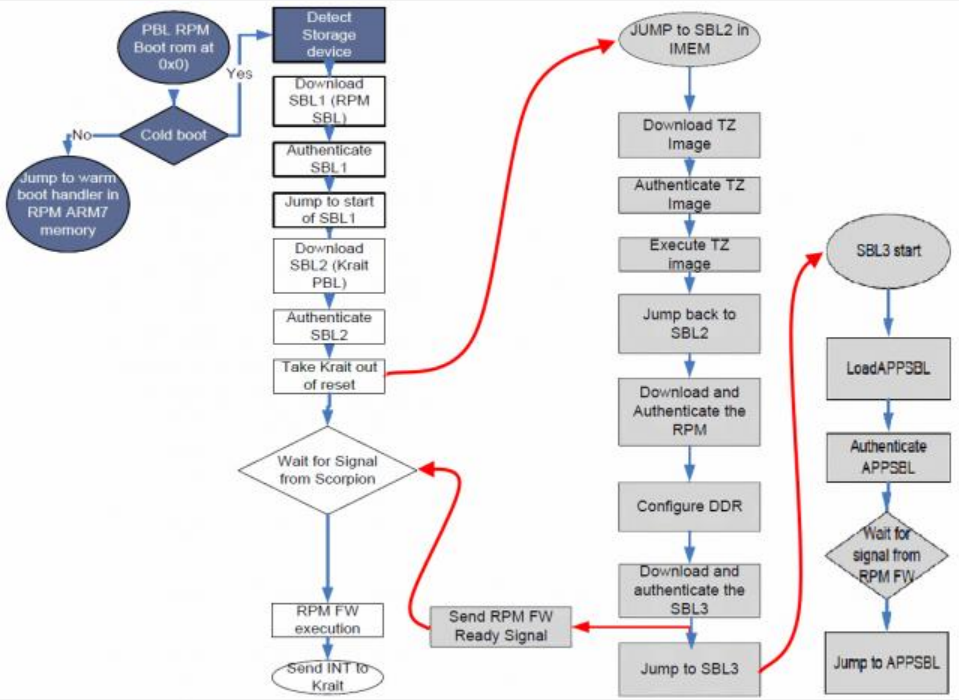


Figure 3: Illustrative baseband architecture

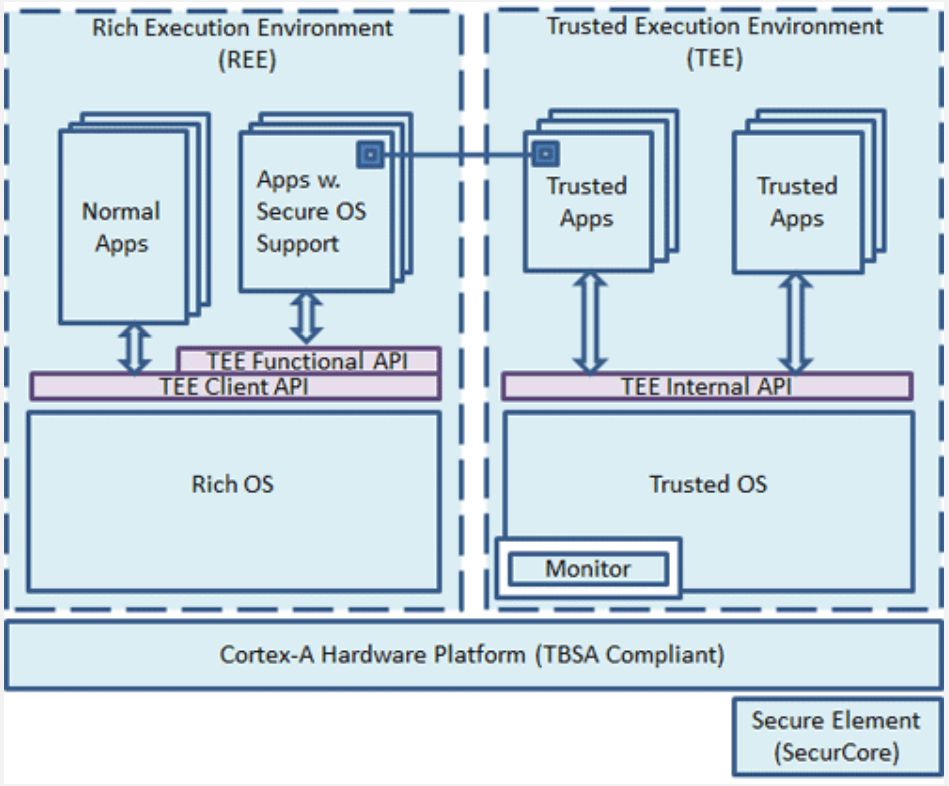
BASEBAND PROCESSOR

- Separate processor or core that manages radio functionality (why?)
- Typically runs a proprietary **real-time** operating system
  - Apple iPhone: Nucleus RTOS, ThreadX
  - Qualcomm: Advanced Mobile Subscriber Software (AMSS/REX OS)
    - L4A Pistachio microkernel

BOOT PROCESS



ARM TRUSTZONE



ARM TRUSTZONE

- Provides a separate hardware-enforced execution environment
  - x86 protection rings (0, 3)
- Applications
  - Digital rights management
  - Secure key storage
  - Mobile payments
  - Secure boot management (Q-Fuses)
  - Kernel integrity monitoring

ARM TRUSTZONE

- Qualcomm Secure Execution Environment (QSEE)
  - Contains separate kernel with separate memory space
  - Has privileged access to all hardware and the non-secure world
  - Interfaces with the non-secure world via the privileged *Secure Monitor Call* (SMC) instruction

CASE STUDIES

- *Baseband Attacks: Remote Exploitation of Memory Corruptions in Cellular Protocol Stacks*, Ralf-Philpp Weinmann (WOOT 2012)
  - Memory corruption in various baseband stacks led to injection/execution of arbitrary code
- *Reflections on Trusting TrustZone*, Dan Rosenberg (BlackHat 2014)
  - Integer overflow vulnerability led to arbitrary write of secure memory
- *TrustNone*, Sean Beaupre (11/28/15)
  - Signed comparison on unsigned user input led to arbitrary read/write of secure memory

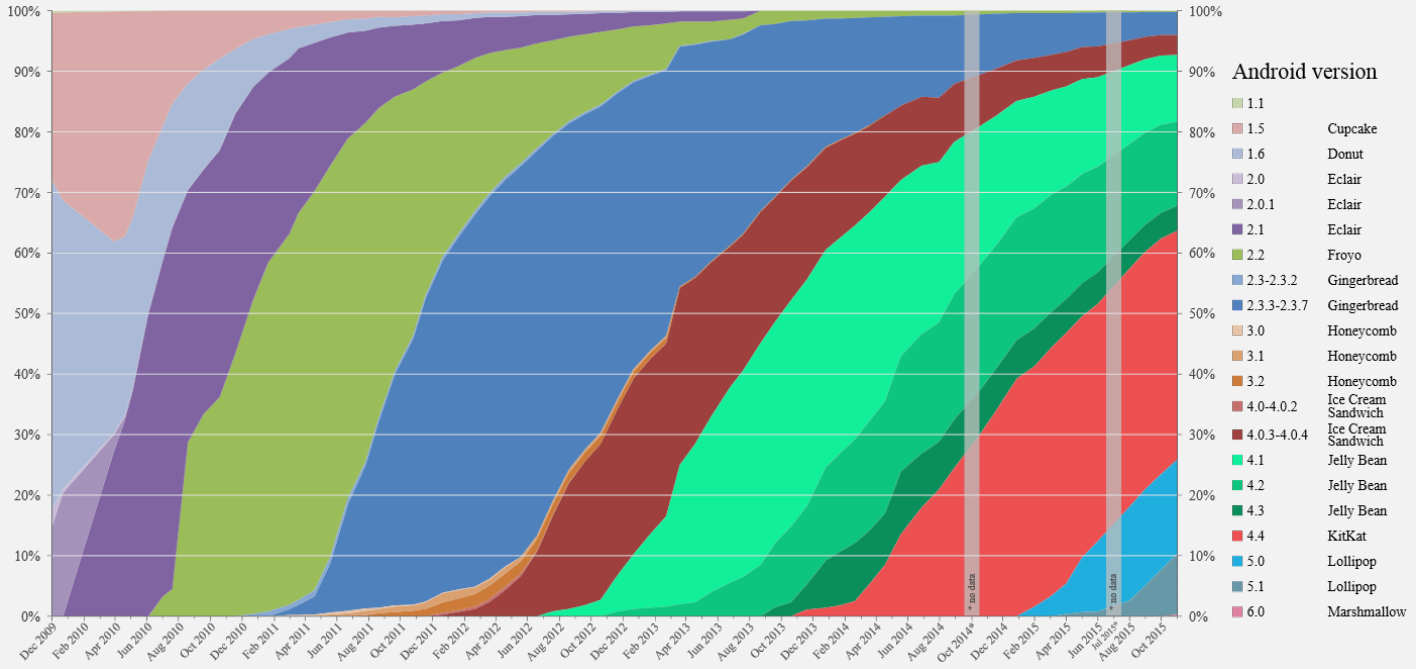
OVERVIEW

- Architectural complexity
  - New attack vectors
- **Mobile operating systems**
  - Operating system safety protections
  - Software development and distribution model
- Common problems with real-world software
  - Cryptographic misuse
  - Personal information leakage
- Current research techniques

INTRODUCTION: ANDROID

- Originally developed by startup in 2003
  - Bought out by Google in 2005
  - Publicly released in 2007
- Mostly released under open source license
  - Proprietary device-specific drivers distributed in binary form
  - Access to Play Store and Google applications requires licensing agreement
    - Fire OS, Baidu, Yandex.Store, etc

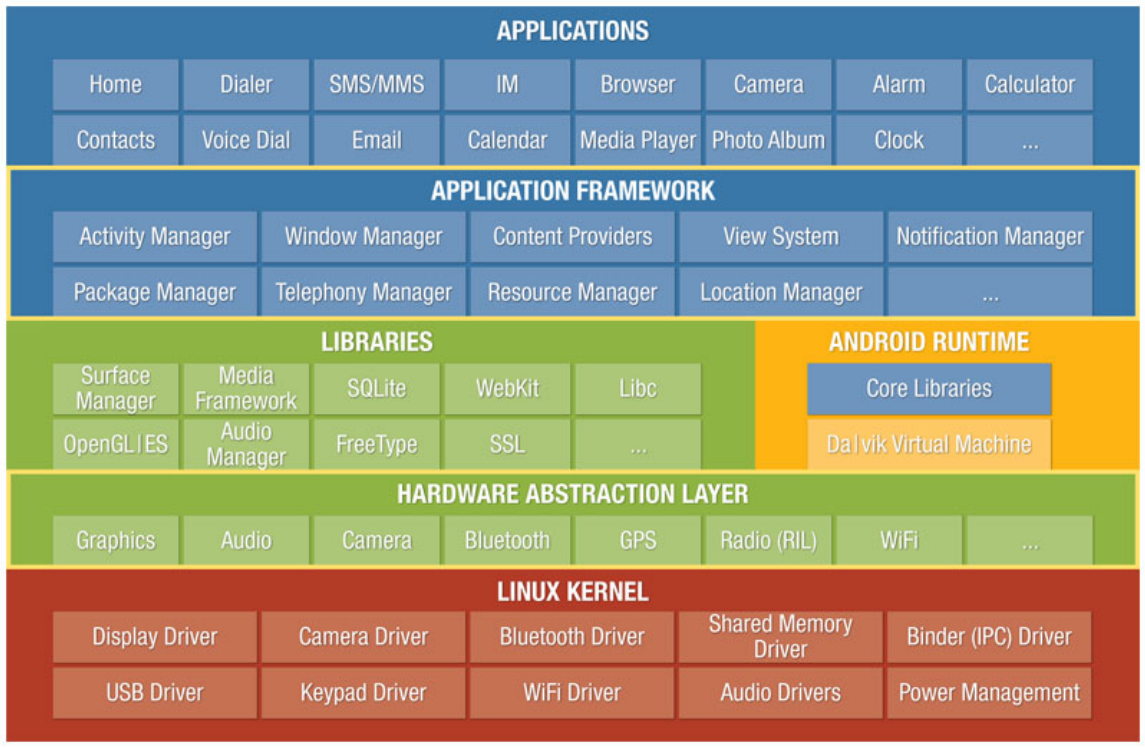
VERSION HISTORY



SECURITY MODEL

- Utilizes a modified version of the Linux kernel
  - Changes are slowly being merged back upstream
- UNIX permission model for applications
  - Mandatory sandbox as separate users (distinct UID)
- Limited interface for inter-process communication
- Applications are cryptographically signed and verified

ARCHITECTURE: ANDROID



SAFETY ENHANCEMENTS

- Android 1.5+
  - Stack overflow protection (-fstack-protector)
  - Safe integer operations (-lsafe_iop)
  - Double free protection
  - Memory allocation integer overflow protection
- Android 2.3+
  - Format-string protections (-Wformat-security)
  - Data execution protection (DEP)
  - NULL pointer dereference protection (vm.mmap_min_addr)
- Android 4.0+
  - Address Space Layout Randomization (ASLR)

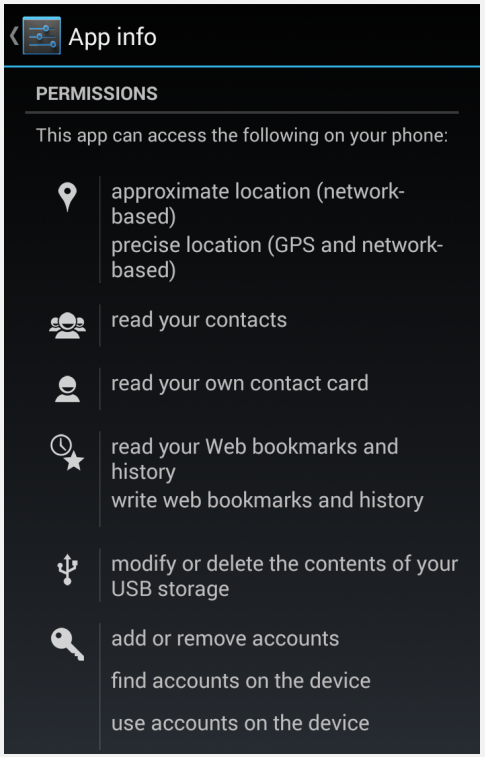
SAFETY ENHANCEMENTS

- Android 4.1+
  - Position Independent Executables (PIE)
  - Read-only relocations (-Wl,-z,relro –Wl,-z,now)
- Android 5.0+
  - Default full disk encryption
  - Mandatory PIE
  - SELinux
- Android 6.0+
  - Verified boot
  - USB access control
  - Monthly security patches

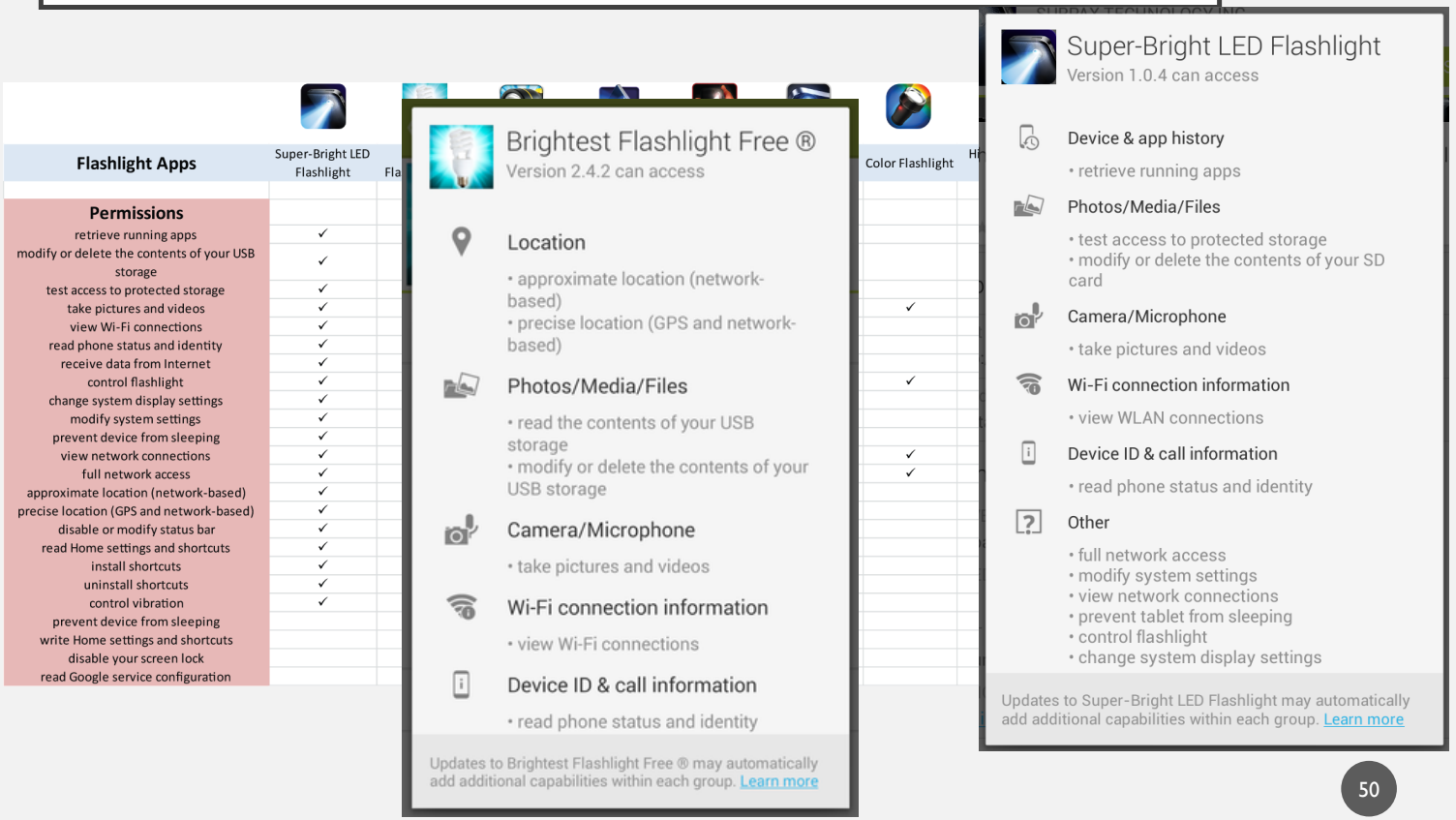
PERMISSION MODEL

- Capability-based access control model
- Categorized into various functional groups
  - Bluetooth
  - Camera
  - Location (fine/coarse-grained)
  - Network/data connection
  - SMS/MMS
  - Telephony
- User receives permission prompt at install-time
  - All-or-nothing

PERMISSION MODEL



PERMISSION MODEL



Flashlight Apps	Super-Bright LED Flashlight	Color Flashlight
Permissions		
retrieve running apps	✓	
modify or delete the contents of your USB storage	✓	
test access to protected storage	✓	
take pictures and videos	✓	✓
view Wi-Fi connections	✓	
read phone status and identity	✓	
receive data from Internet	✓	
control flashlight	✓	✓
change system display settings	✓	
modify system settings	✓	
prevent device from sleeping	✓	
view network connections	✓	✓
full network access	✓	✓
approximate location (network-based)	✓	
precise location (GPS and network-based)	✓	
disable or modify status bar	✓	
read Home settings and shortcuts	✓	
install shortcuts	✓	
uninstall shortcuts	✓	
control vibration	✓	
prevent device from sleeping		
write Home settings and shortcuts		
disable your screen lock		
read Google service configuration		

PERMISSION MODEL

- Starting with Android 6.0 (Marshmallow), permissions are queried at run-time
  - Allows users to deny individual permissions
  - Was briefly available for Android 4.4.0 – 4.4.2
- 3rd party solutions
  - Xposed Framework (requires root)

APPLICATION STRUCTURE

- Written in Java
  - Interpreted by Dalvik bytecode virtual machine
    - Uses just-in-time (JIT) techniques to compile native code
  - Replaced with Android Runtime (ART) in 5.0+
    - Introduces ahead-of-time (AOT) compilation instead of JIT
- Can also call into native code
  - Uses Java Native Interface (JNI) to interface with C/C++ libraries

APPLICATION STRUCTURE

- Activity
  - Portions of the application's user interface
    - Login window, registration interface, etc.
- Service
  - Performs background processing
    - Download a file, play music, etc.
- Broadcast Receiver
  - Handlers for global messages
    - Boot completed, power disconnected, etc.
- Content Provider
  - Manages access to structured data
    - User calendar, contacts, etc.

CASE STUDIES

- *Stagefright*, Zimperium (2015)
  - Integer overflow vulnerabilities in system multimedia library leads to remote code execution
    - Fixed in November monthly security patch
- *Master Key*, Bluebox Security (2013)
  - Structure of Android application packages allows manipulation of contents without invalidating digital signatures

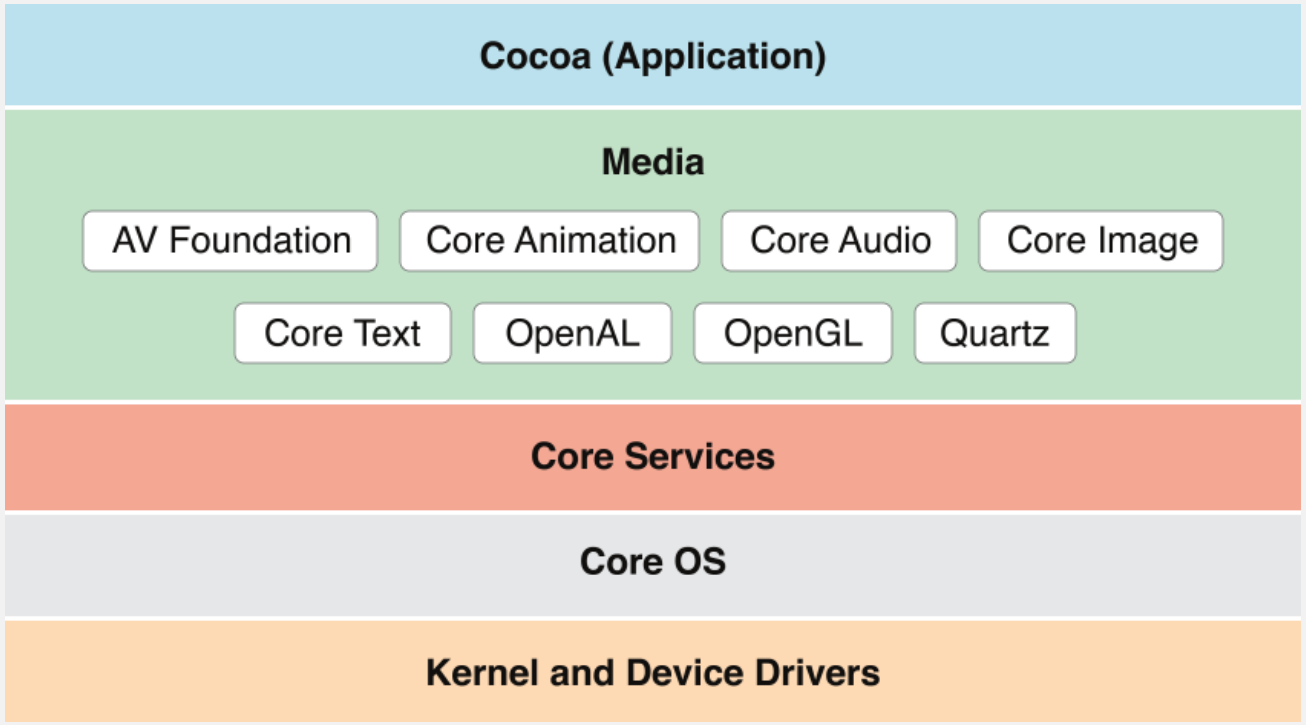
INTRODUCTION: IOS

- Originally developed in 2005
  - Publicly released in 2007
- Based off of the Macintosh XNU kernel
  - Supports memory-protection features
    - ASLR, DEP, etc.
  - UNIX-like

SECURITY MODEL

- All applications must be signed by Apple
  - Unless system is jailbroken to remove checks
- Individual applications are encrypted and sandboxed from one another
- Code integrity is verified during execution
  - Makes injection of shellcode difficult

ARCHITECTURE: IOS



APPLICATION STRUCTURE

- Written in Objective-C or Swift
  - Compiled by Clang/LLVM into native code
  - Adds automatic reference counting for garbage collection in Swift
    - Transitioning to open source later this year
- Uses Model-View-Controller (MVC) design paradigm
  - Applications objects are model, view, or controller
  - Abstracts data from logic and presentation

APPLICATION APPROVALS

- Applications are typically submitted by developers to App Store for inclusion
- These undergo a review process for unwanted behavior or policy violations
  - Objectionable content
  - Game emulators
  - Internal API's
- Techniques
  - Static analysis
  - Manual review

ENTERPRISE PROVISIONING

- Enterprise developer certificates allow bypass of the App Store
  - Designed for deployment of internal applications to employees
- Historically, have also been used to bypass platform security
  - Game emulators
  - Jailbreaking
  - Malware

CASE STUDIES

- *XcodeGhost*, Alibaba (2015)
  - Modified version of Xcode uploaded to a Chinese file sharing service inserted malicious code into binaries
- *Pangu8*, Pangu Team (2015)
  - Heap overflow in kernel battery gauge service for iOS 8 led to arbitrary writes of kernel memory
- *limera1n*, George Hotz (2010)
  - Heap overflow in bootloader USB protocol implementation led to arbitrary writes of memory

OVERVIEW

- Architectural complexity
  - New attack vectors
- Mobile operating systems
  - Operating system safety protections
  - Software development and distribution model
- **Common problems with real-world software**
  - Cryptographic misuse
  - Personal information leakage
- Current research techniques

COMMON PROBLEMS

- Developers are not experts in implementing or using cryptography
  - Tendency to copy-paste "template" code
  - Need to disable certain cryptographic features for ease of debugging
- Developers tend to inadvertently or maliciously request extraneous permissions
  - Can use user information for advertising or analytics

CRYPTOGRAPHIC MISUSE

1. Usage of ECB mode for encryption
2. Usage of static IV's in CBC mode
3. Usage of hardcoded symmetric encryption keys
4. Usage of low iterations for password-based encryption
5. Bad seeding of random-number generators

CRYPTOGRAPHIC MISUSE

- *CryptoLint*, Manuel Egele et al. (CCS 2013)
  1. Extract a control flow graph of an application
  2. Identify calls to sensitive cryptographic API's
  3. Perform static backward slicing to evaluate security rules
- Allows for automatic detection of cryptographic misuse

CONCLUSION

- Architectural complexity
  - New attack vectors
- Mobile operating systems
  - Operating system safety protections
  - Software development and distribution model
- Common problems with real-world software
  - Cryptographic misuse
  - Personal information leakage
- Current research techniques

Questions?

END