


Mobile Platform Security Models

John Mitchell

Acknowledgments: Lecture slides are from the Computer Security course thought by Dan Boneh and John Mitchell at Stanford University. When slides are obtained from other sources, a reference will be noted on the bottom of that slide. A full list of references is provided on the last slide.

Outline

- 
- ◆ Introduction
 - Platforms
 - App market
 - Threats
 - ◆ Android security model
 - ◆ Apple iOS security model
 - ~~◆ Windows 7, 8 Mobile security model~~

Change takes time



Apple Newton, 1987

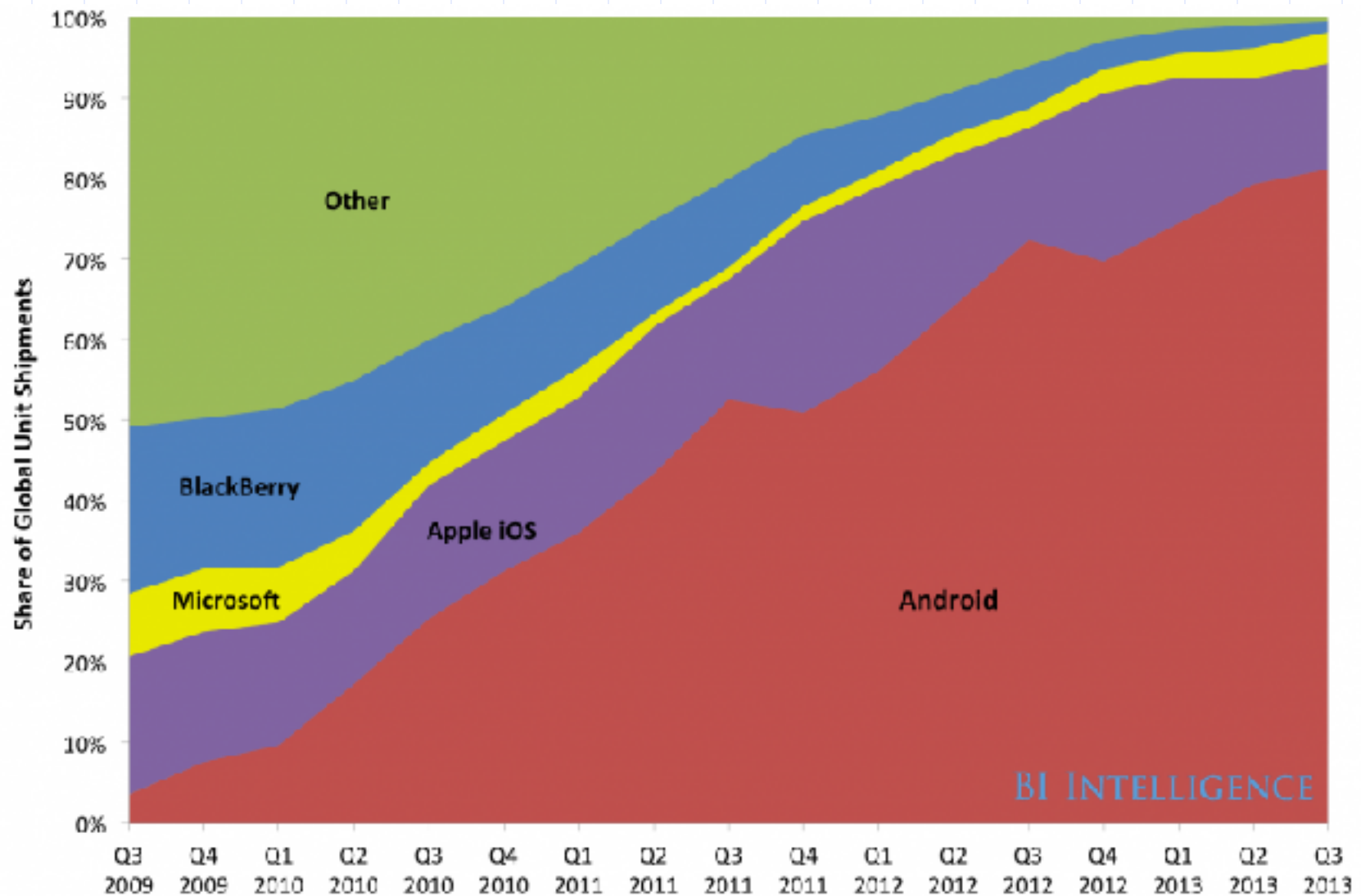


Palm Pilot, 1997

iPhone, 2007

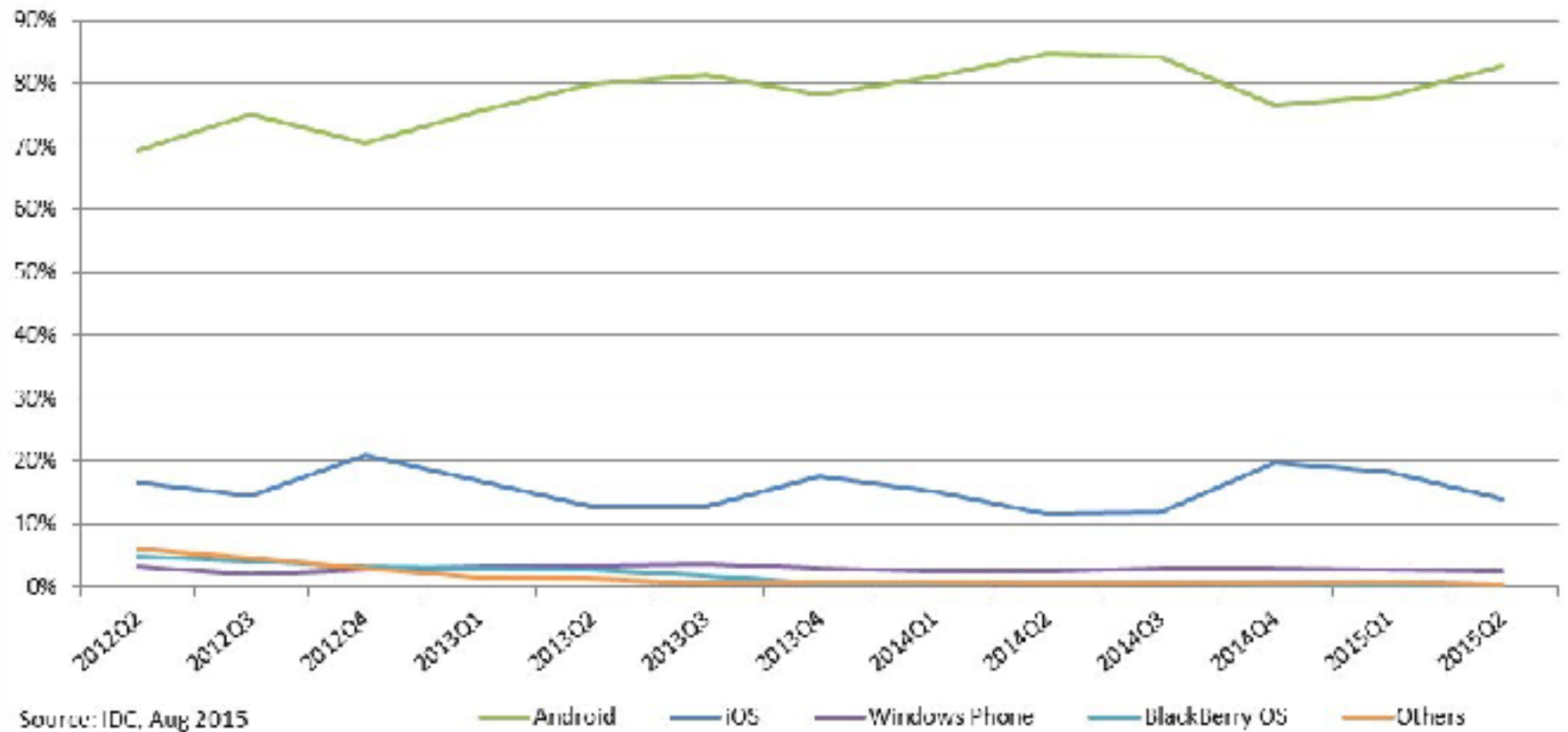


Global smartphone market share



Source: IDC, Strategy Analytics

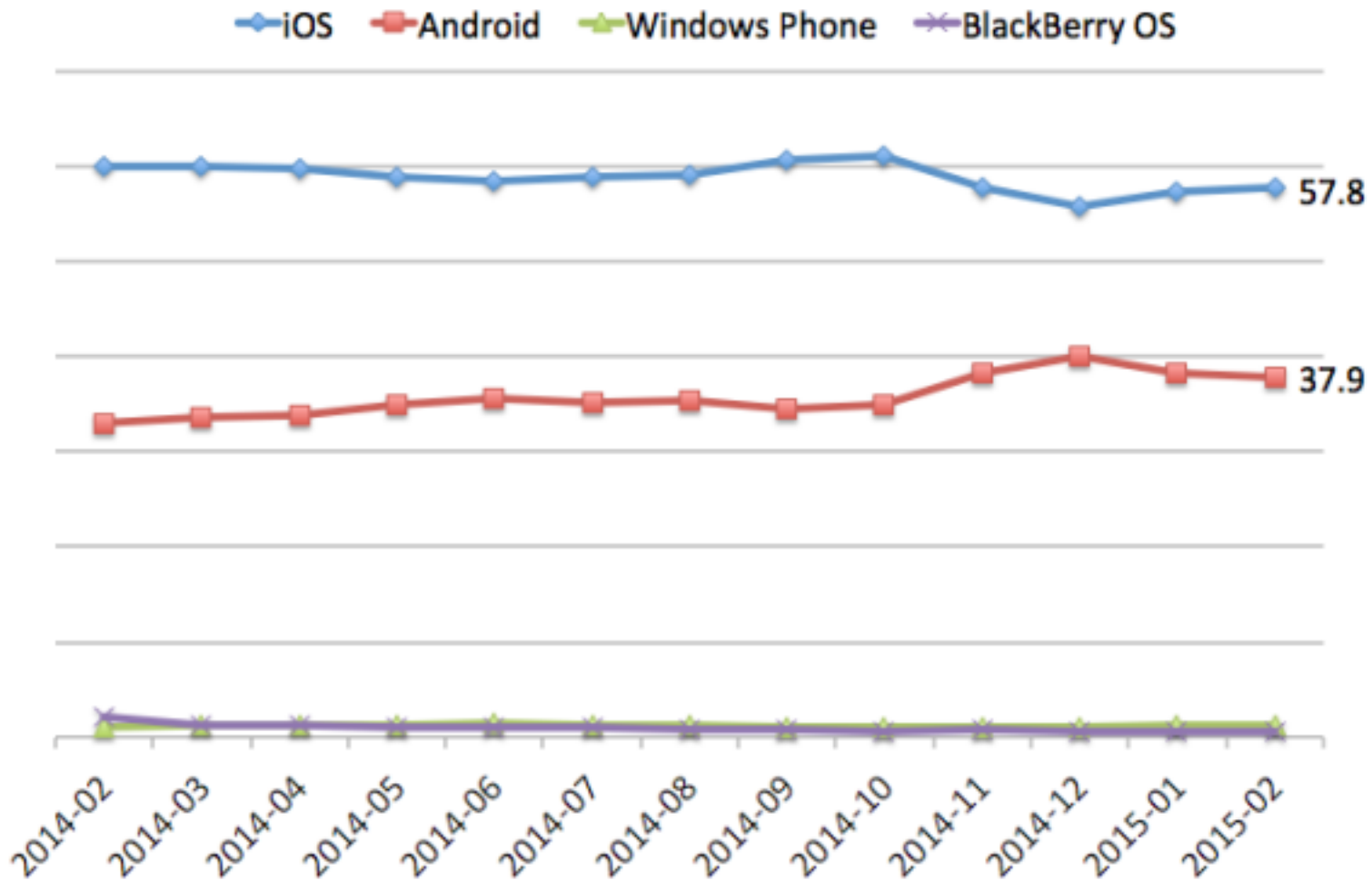
Worldwide Smartphone OS Market Share (Share in Unit Shipments)



Source: IDC, Aug 2015

— Android — iOS — Windows Phone — BlackBerry OS — Others

US Mobile App Traffic



Zillions of apps



App Marketplace

◆ App review before distribution

- iOS: Apple manual and automated vetting
- Android
 - ◆ Easier to get app placed on market
 - ◆ Transparent automated scanning, removal via Bouncer

◆ App isolation and protection

- Sandboxing and restricted permission
- Android
 - ◆ Permission model
 - ◆ Defense against circumvention

Threats to mobile applications

◆ Privacy

- Data leakage, identifier leakage, third-party tags and libraries, location privacy

◆ Security

- Phishing, malware & drive-bys, malicious intents on Android, Ikee/Zitmo and other mobile malware

OWASP Mobile Top Ten

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

Mobile malware examples

◆ DroidDream (Android)

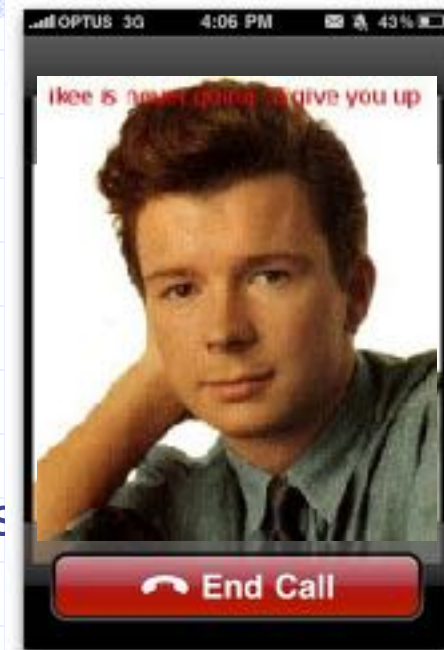
- Over 58 apps uploaded to Google app market
- Conducts data theft; send credentials to attackers

◆ Ikee (iOS)

- Worm capabilities (targeted default ssh pwd)
- Worked only on jailbroken phones with ssh installed

◆ Zitmo (Symbian, BlackBerry, Windows, Android)

- Propagates via SMS; claims to install a “security certificate”
- Captures info from SMS; aimed at defeating 2-factor auth
- Works with Zeus botnet; timed with user PC infection



Sample FTC concerns

- ◆ FTC To Study Mobile Device Industry's Security Update Practices (May 9, 2016)
- ◆ Federal Court Finds Amazon Liable for Billing Parents for Children's Unauthorized In-App Charges (April 27, 2016)
- ◆ Tech Company Settles FTC Charges It Unfairly Installed Apps on Android Mobile Devices Without Users' Permission (February 5, 2016)
- ◆ Defendants in Massive Spam Text Message, Robocalling and Mobile Cramming Scheme to Pay \$10 Million to Settle FTC Charges (October 22, 2014)
- ◆ Snapchat Settles FTC Charges That Promises of Disappearing Messages Were False (May 8, 2014)

Outline



◆ Introduction

- Platforms
- App market
- Threats



◆ Android security model

◆ Apple iOS security model

◆ ~~Windows 7, 8 Mobile security model~~

Android

◆ Platform outline:

- Linux kernel, browser, SQL-lite database
- Software for secure network communication
 - ◆ Open SSL, Bouncy Castle crypto API and Java library
- C language infrastructure
- Java platform for running applications
 - ◆ Dalvik bytecode, virtual machine

APPLICATIONS

Home

Contacts

Phone

Browser

...

APPLICATION FRAMEWORK

Activity Manager

Window Manager

Content Providers

View System

Package Manager

Telephony Manager

Resource Manager

Location Manager

Notification Manager

LIBRARIES

Surface Manager

Media Framework

SQLite

OpenGL | ES

FreeType

WebKit

SGL

SSL

libc

ANDROID RUNTIME

Core Libraries

Dalvik Virtual Machine

LINUX KERNEL

Display Driver

Camera Driver

Flash Memory Driver

Binder (IPC) Driver

Keypad Driver

WiFi Driver

Audio Drivers

Power Management

Android market

◆ Self-signed apps

◆ App permissions granted on user installation

◆ Open market

- Bad applications may show up on market
- Shifts focus from remote exploit to privilege escalation

Android permissions

◆ Example of permissions provided by Android

- "android.permission.INTERNET"
- "android.permission.READ_EXTERNAL_STORAGE"
- "android.permission.SEND_SMS"
- "android.permission.BLUETOOTH"

◆ Also possible to define custom permissions

Android permission model

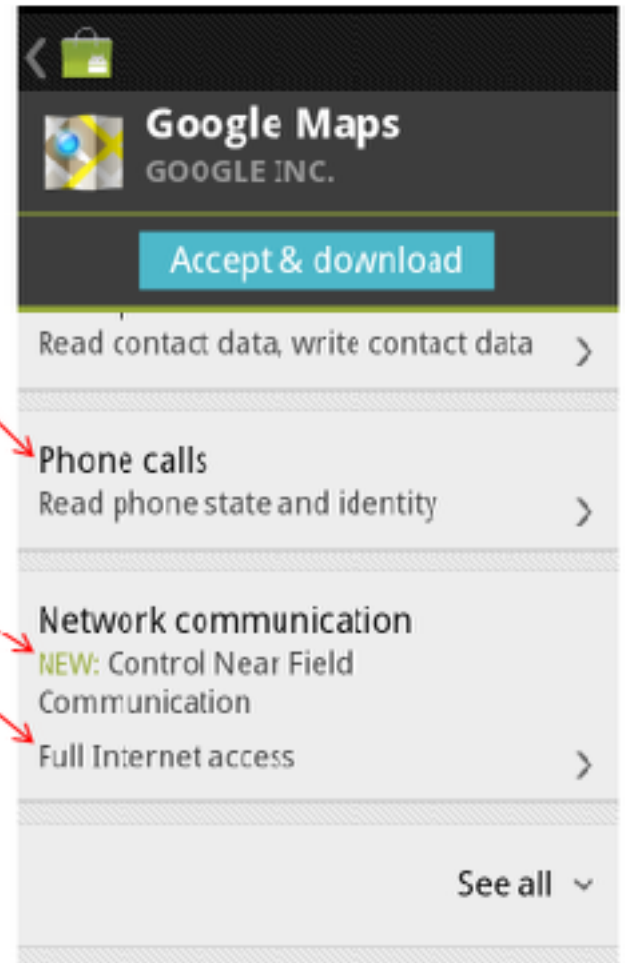
...

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

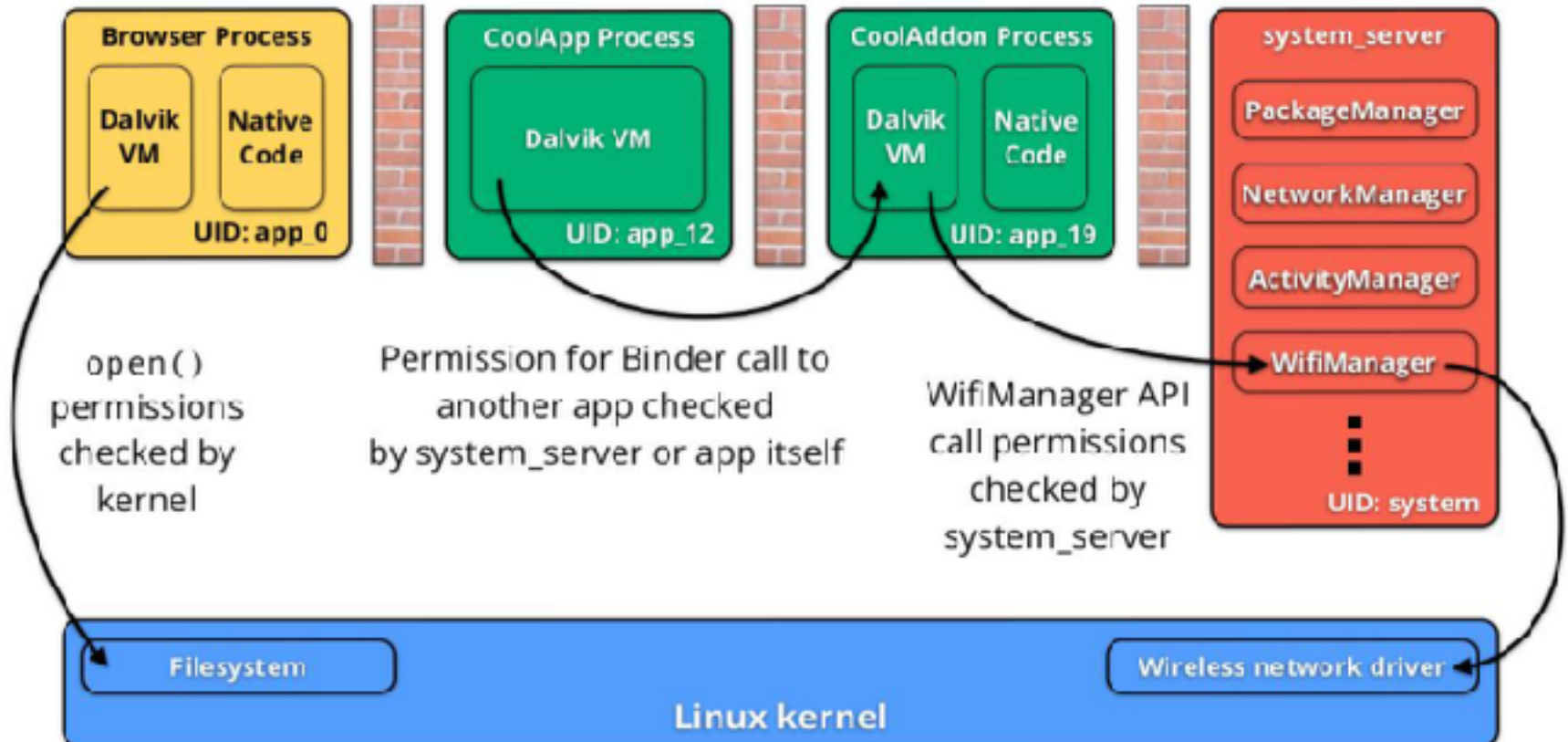
```
<uses-permission android:name="android.permission.NFC" />
```

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

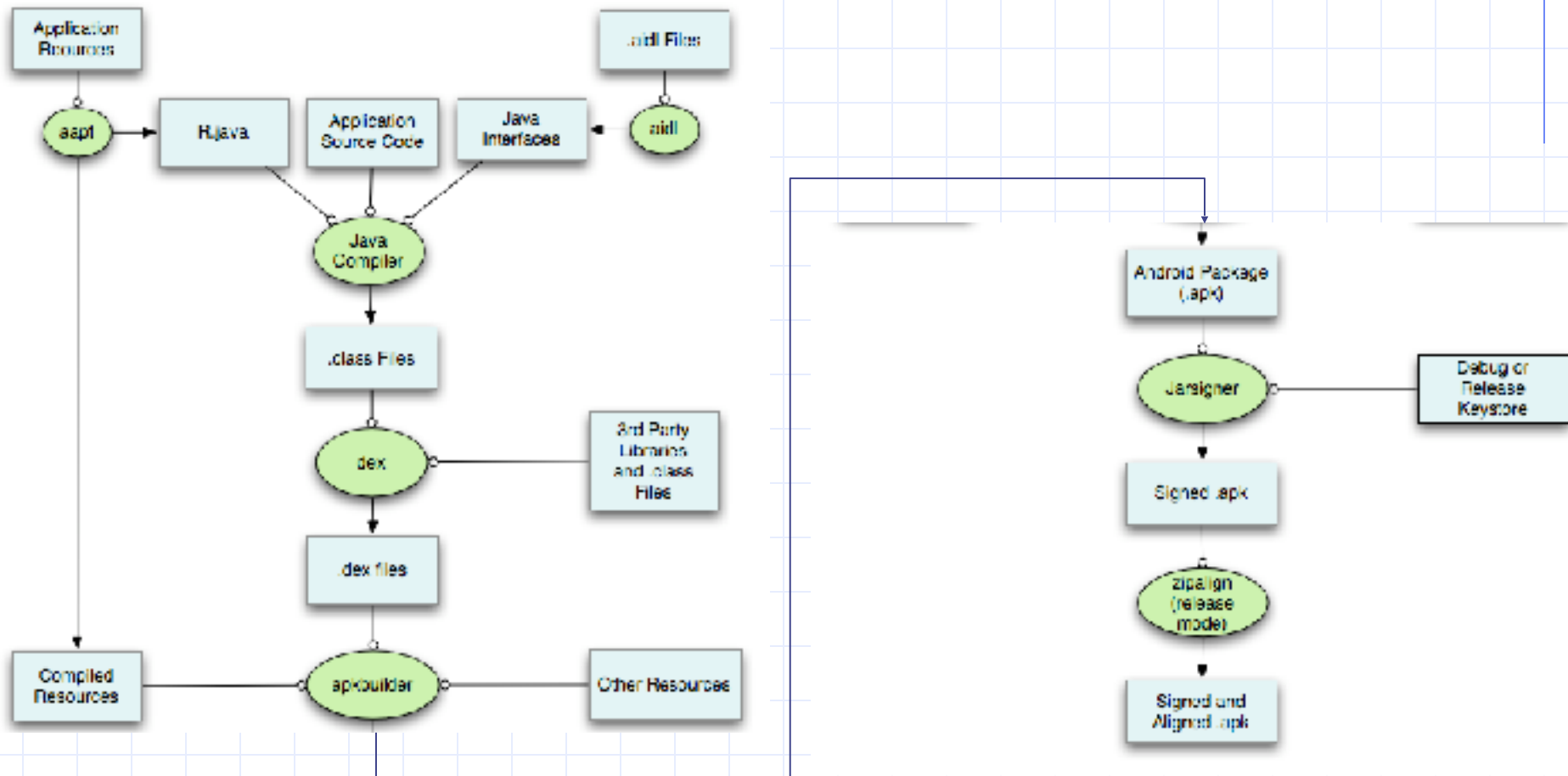
...



Android permission model



Application development process



Security Features

◆ Isolation

- Multi-user Linux operating system
- Each application normally runs as a different user

◆ Communication between applications

- May share same Linux user ID
 - ◆ Access files from each other
 - ◆ May share same Linux process and Dalvik VM
- Communicate through application framework
 - ◆ “Intents,” based on Binder, discussed in a few slides

◆ Battery life

- Developers must conserve power
- Applications store state so they can be stopped (to save power) and restarted – helps with DoS

Application sandbox

◆ Application sandbox

- Each application runs with its UID in its own Dalvik virtual machine
 - ◆ Provides CPU protection, memory protection
 - ◆ Authenticated communication protection using Unix domain sockets
 - ◆ Only ping, zygote (spawn another process) run as root
- Applications announce permission requirement
 - ◆ Create a whitelist model – user grants access
 - Don't interrupt user – all questions asked as install time
 - ◆ Inter-component communication reference monitor checks permissions

Exploit prevention

◆ Open source: public review, no obscurity

◆ Goals

- Prevent remote attacks, privilege escalation
- Secure drivers, media codecs, new and custom features

◆ Overflow prevention

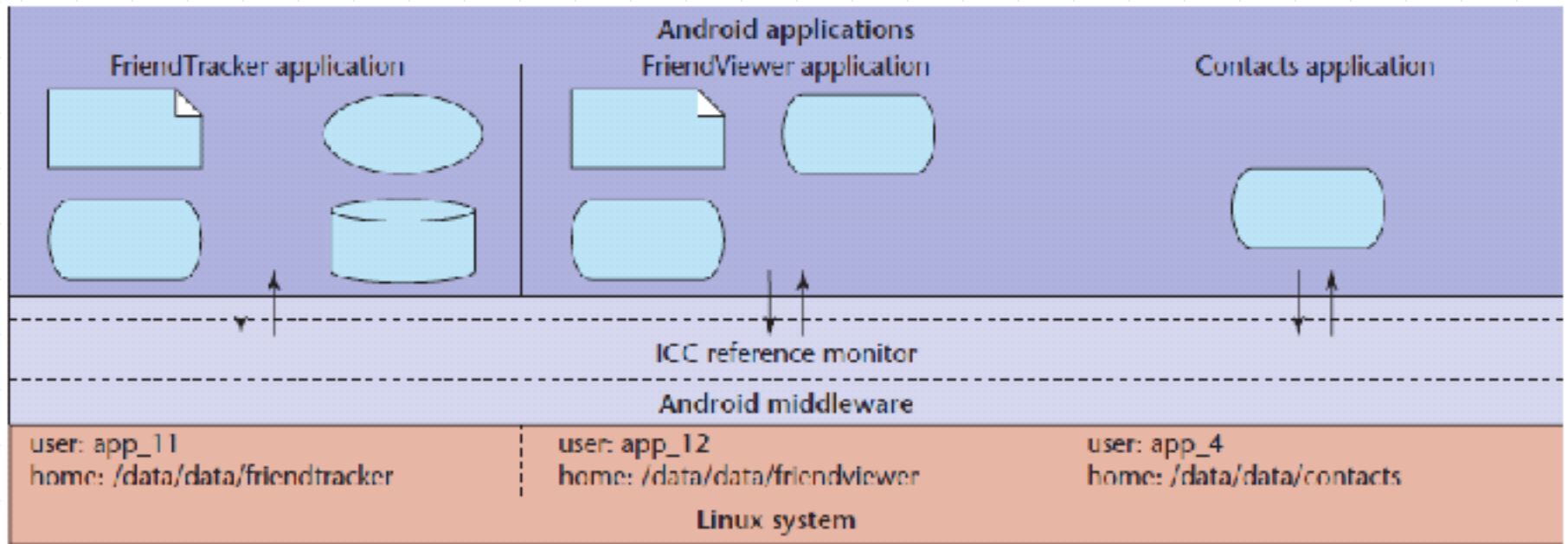
- ProPolice stack protection
 - ◆ First on the ARM architecture
- Some heap overflow protections
 - ◆ Chunk consolidation in DL malloc (from OpenBSD)

◆ ASLR

- Avoided in initial release due to performance concerns
- Later developed and contributed by Bojinov, Boneh

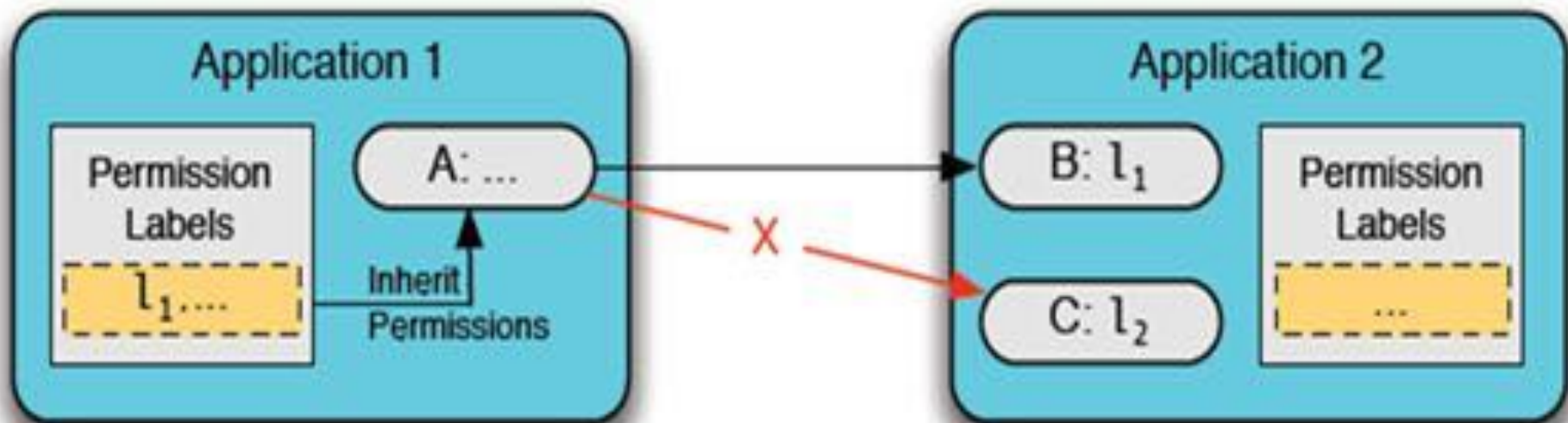
Android Intents

- ◆ Message between components in same or different app
- ◆ Intent is a bundle of information, e.g.,
 - action to be taken
 - data to act on
 - category of component to handle the intent
 - instructions on how to launch a target activity
- ◆ Routing can be
 - Explicit: delivered only to a specific receiver
 - Implicit: all components that have registered to receive that action will get the message



◆ Layers of security

- Each application executes as its own user identity
- Android middleware has reference monitor that mediates the establishment of inter-component communication (ICC)



MAC Policy Enforcement in Android. This is how applications access components of other applications via the reference monitor. Component A can access components B and C if permission labels of application 1 are equal or dominate labels of application 2.

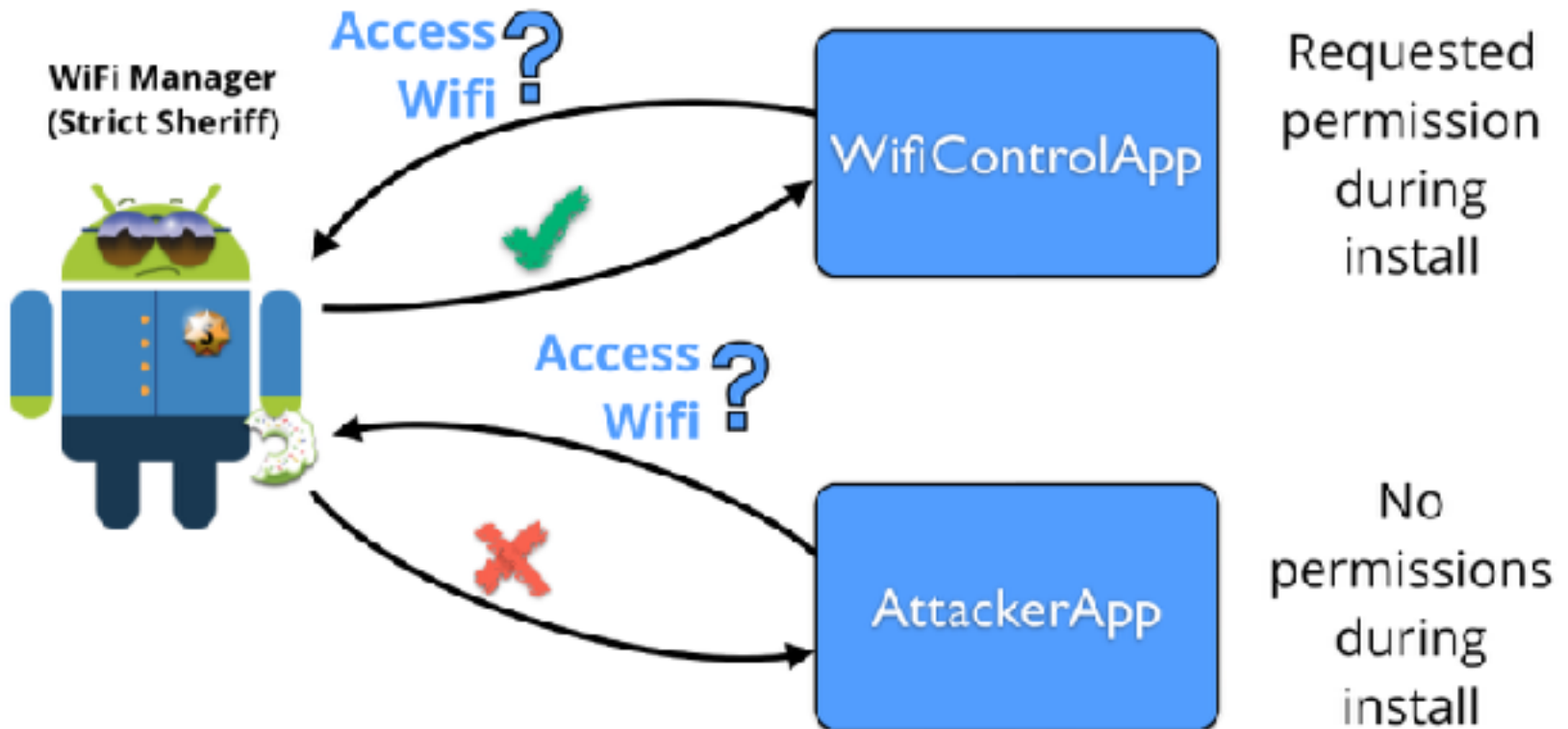
Security issues with intents

- ◆ Sender of an intent can verify that the recipient has a permission by specifying a permission with the method call
- ◆ Senders can use explicit intents to send the message to a single component (avoiding broadcasting)
- ◆ Receivers have to handle malicious intents

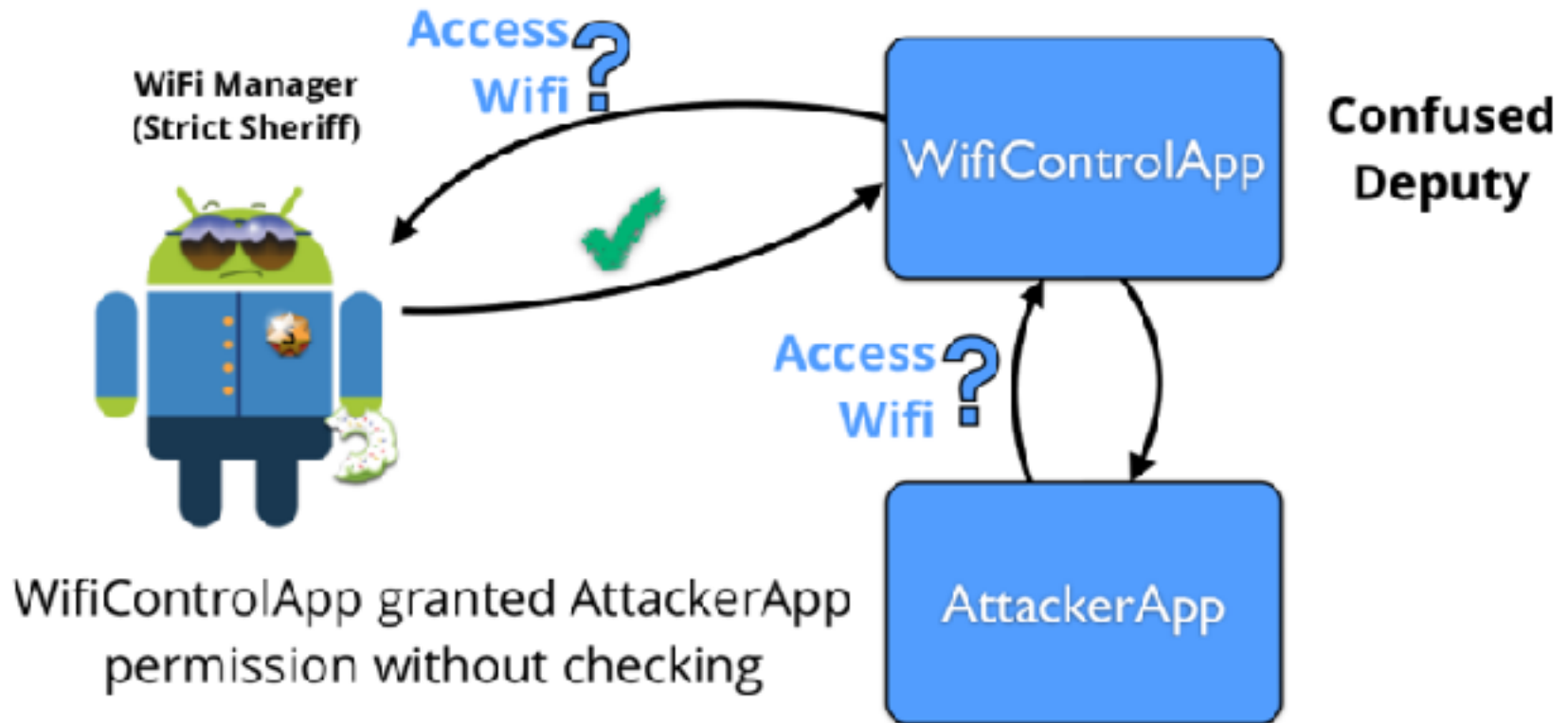
Attack: Permission redelegation

- ◆ Definition: an application without a permission gains additional privileges through another application
- ◆ Example of the “confused deputy” problem

Permission redelegation



Permission redelegation



How could this happen?

◆ App w/ permissions exposes a public interface

◆ Study in 2011

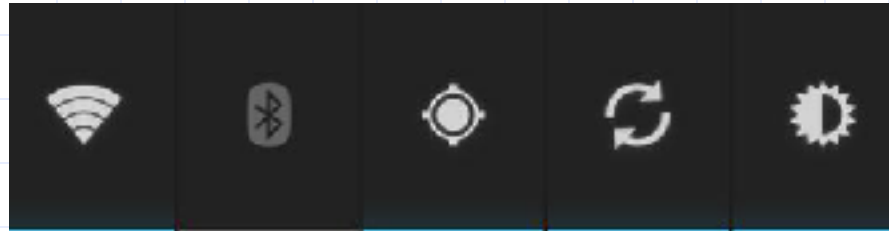
- Examine 872 apps
- 320 of these (37%) have permissions and at least one type of public component
- Construct attacks using 15 vulnerabilities in 5 apps

◆ Reference

- Permission Re-Delegation: Attacks and Defenses, Adrienne Felt, Helen Wang, Alexander Moshchuk, Steven Hanna, Erika Chin, Usenix 2011

Example: power control widget

- ◆ Default widgets provided by Android, present on all devices



- ◆ Can change Wi-fi, BT, GPS, Data Sync, Screen Brightness with only one click
- ◆ Uses Intent to communicate the event of switching settings
- ◆ A malicious app without permissions can send a fake Intent to the Power Control Widget, simulating click to switch settings

Vulnerable versions (in red)

Version	Codename	API	Distribution
1.6	Donut	4	0.10%
2.1	Eclair	7	1.50%
2.2	Froyo	8	3.20%
2.3 - 2.3.2	Gingerbread	9	0.10%
2.3.3 - 2.3.7		10	36.40%
3.2	Honeycomb	13	0.10%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	25.60%
4.1.x	Jelly Bean	16	29.00%
4.2.x		17	4.00%

◆ Apps with permissions need to manage security

Outline



Introduction

- Platforms
- App market
- Threats



Android security model



Apple iOS security model



~~Windows 7, 8 Mobile security model~~

Apple iOS

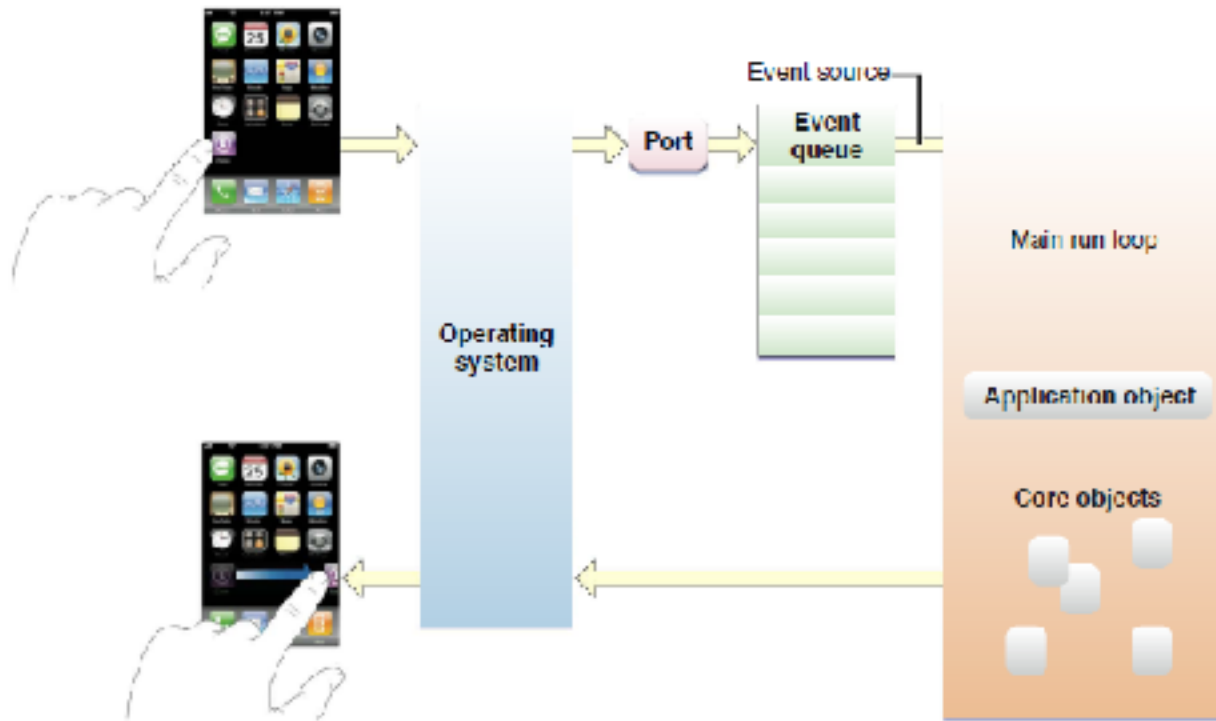


From: iOS App Programming Guide

Reference

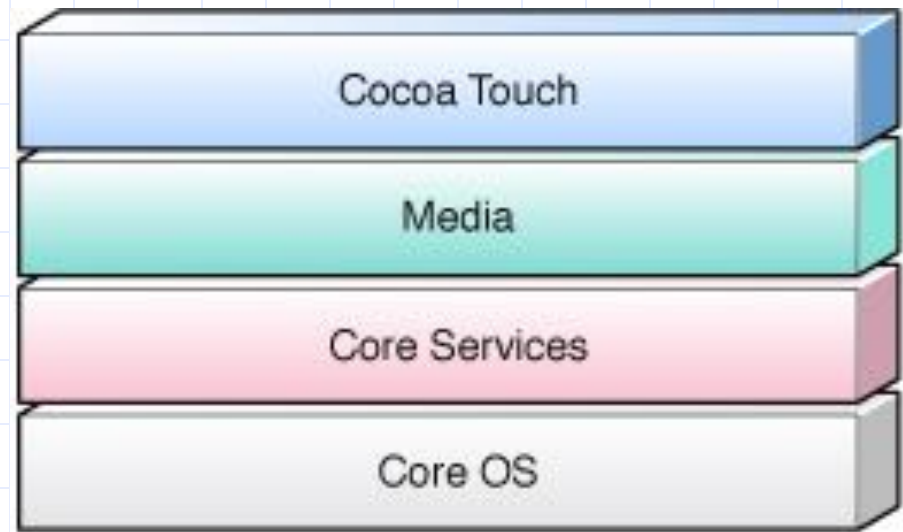
iOS Security (9.3), May 2016

iOS Application Development



- ◆ Apps developed in Objective-C using Apple SDK
- ◆ Event-handling model based on touch events
- ◆ Foundation and UIKit frameworks provide the key services used by all iOS applications

iOS Platform



- ◆ Cocoa Touch: Foundation framework, OO support for collections, file management, network operations; UIKit
- ◆ Media layer: supports 2D and 3D drawing, audio, video
- ◆ Core OS and Core Services: APIs for files, network, ... includes SQLite, POSIX threads, UNIX sockets
- ◆ Kernel: based on Mach kernel like Mac OS X

Implemented in C and Objective-C

Apple iOS Security

◆ Device security

- Prevent unauthorized use of device

◆ Data security

- Protect data at rest; device may be lost or stolen

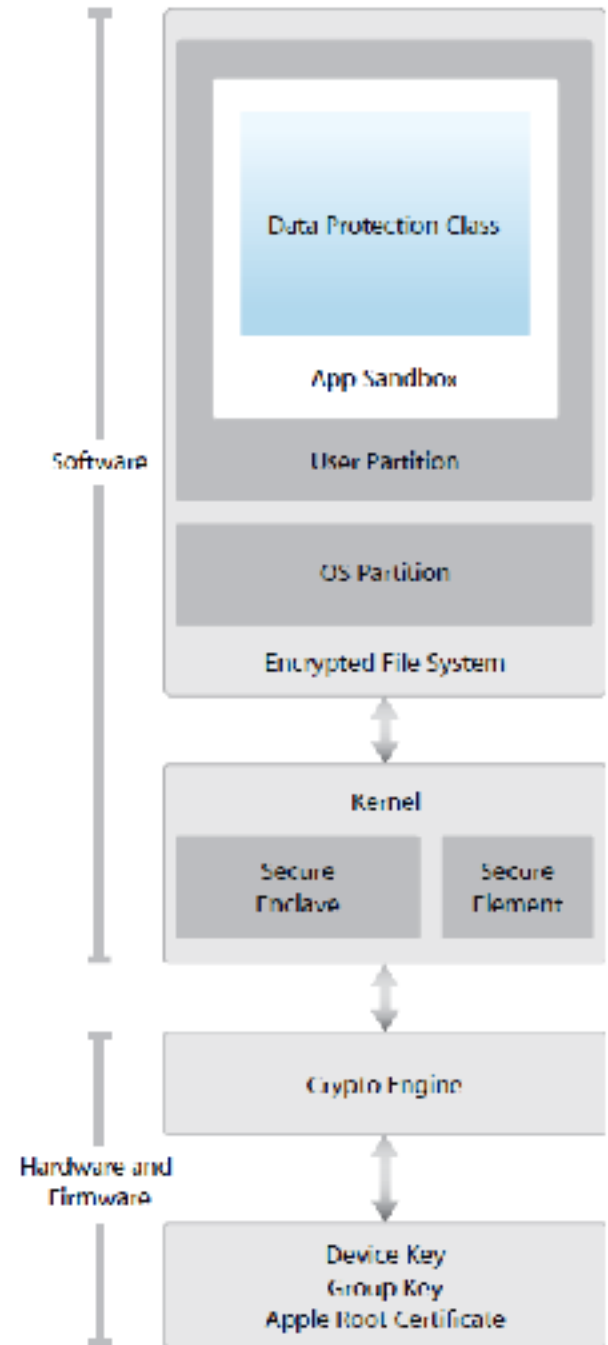
◆ Network security

- Networking protocols and encryption of data in transmission

◆ App security

- Secure platform foundation

https://www.apple.com/business/docs/iOS_Security_Guide.pdf





Runtime protection

- System resources, kernel shielded from user apps
- App “sandbox” prevents access to other app’s data
- Inter-app communication only through iOS APIs
- Code generation prevented



Mandatory code signing

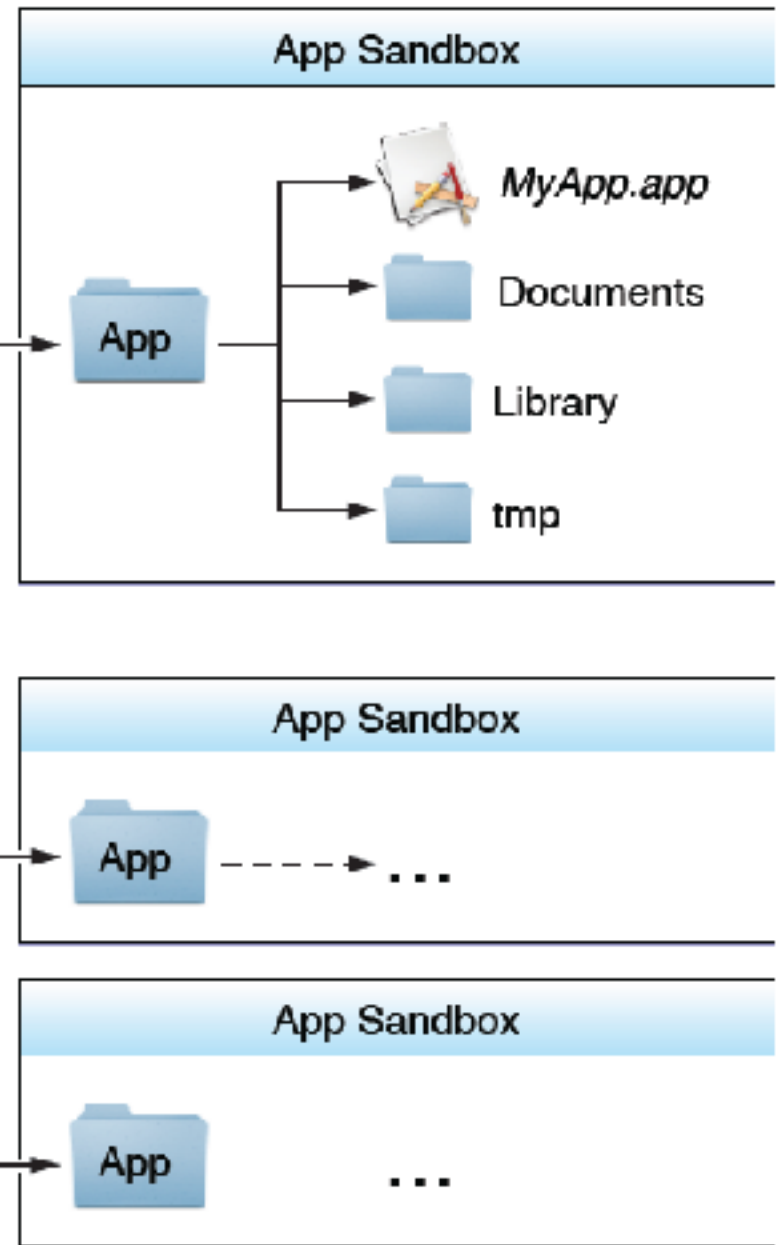
- All apps must be signed using Apple-issued certificate



Application data protection

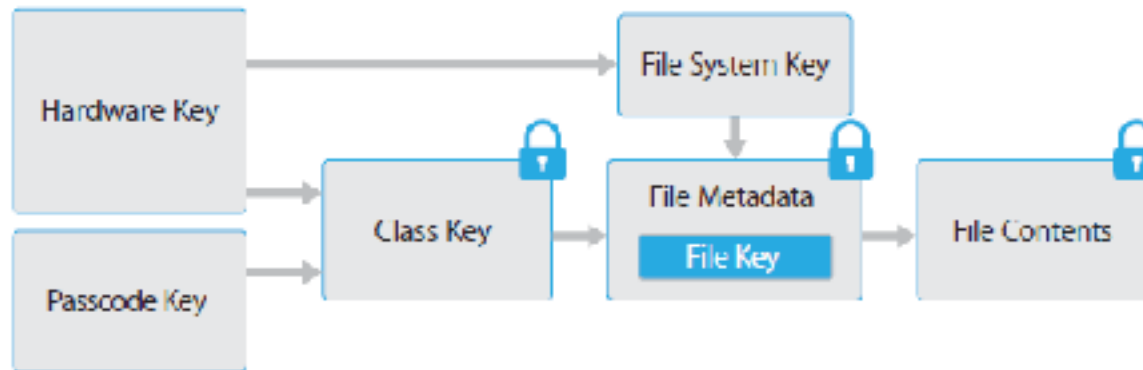
- Apps can leverage built-in hardware encryption

iOS Sandbox



- ◆ Limit app's access to files, preferences, network, other resources
- ◆ Each app has own sandbox directory
- ◆ Limits consequences of attacks
- ◆ Same privileges for each app

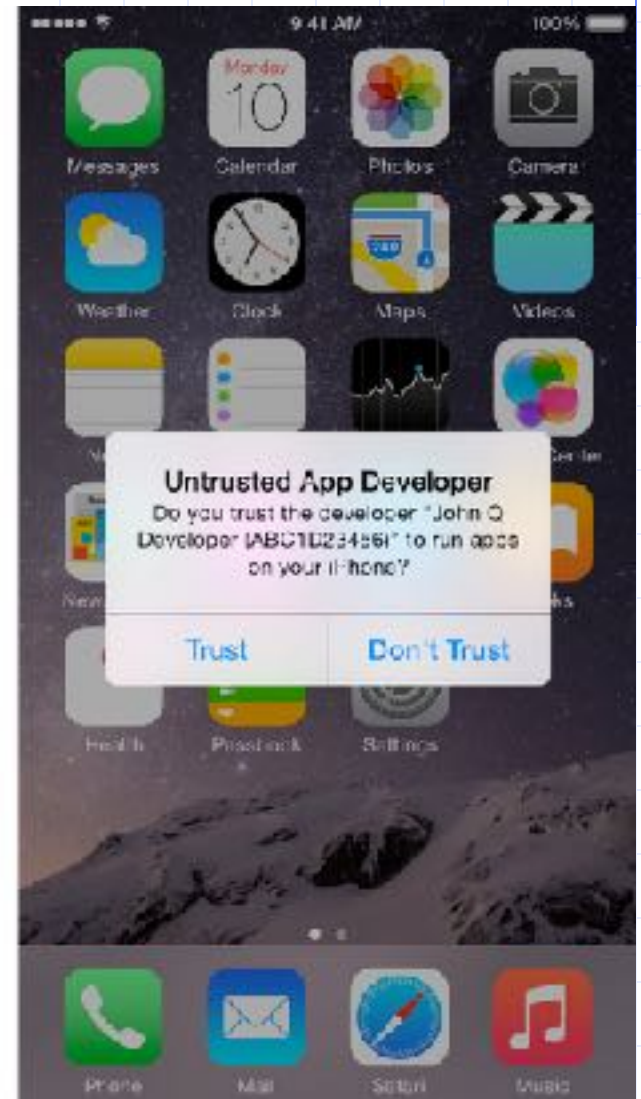
File encryption



- ◆ The content of a file is encrypted with a per-file key, which is wrapped with a class key and stored in a file's metadata, which is in turn encrypted with the file system key.
 - When a file is opened, its metadata is decrypted with the file system key, revealing the wrapped per-file key and a notation on which class protects it
 - The per-file key is unwrapped with the class key, then supplied to the hardware AES engine, decrypting the file as it is read from flash memory
- ◆ The metadata of all files is encrypted with a random key. Since it's stored on the device, used only for quick erased on demand.

“Masque Attack”

- ◆ iOS app installed using enterprise/ad-hoc provisioning could replace genuine app installed through the App Store, if both apps have same bundle identifier
- ◆ This vulnerability existed because iOS didn't enforce matching certificates for apps with the same bundle identifier



Comparison: iOS vs Android

◆ App approval process

- Android apps from open app store
- iOS vendor-controlled store of vetted apps

◆ Application permissions

- Android permission based on install-time manifest
- All iOS apps have same set of “sandbox” privileges

◆ App programming language

- Android apps written in Java; no buffer overflow...
- iOS apps written in Objective-C

Comparison

	iOS	Android	Windows
Unix	x	x	
Windows			
Open market		x	
Closed market	x		
Vendor signed	x		
Self-signed		x	
User approval of permissions		x	
Managed code		x	
Native code	x		

Comparison

	iOS	Android	Windows
Unix	x	x	
Windows			x
Open market		x	
Closed market	x		x
Vendor signed	x		
Self-signed		x	x
User approval of permissions		x	7-> 8
Managed code		x	x
Native code	x		

Conclusion

◆ Overview: Platform, market, threats

◆ Android security model

- Platform security features
- Isolated process with separate VM
- Permission model
- App communication via intents

◆ Apple iOS security model

- App sandbox based on file isolation
- File encryption

◆ ~~Windows Mobile security model~~