

CO 445H

**MOBILE PLATFORM SECURITY
AND MOBILE PRIVACY**

Dr. Benjamin Livshits

Privacy International Data Tracking

2

These companies are like invisible strangers, peering over your shoulder taking notes about what you do online and offline.



<http://privacyinternational.org/feature/2433/i-asked-online-tracking-company-all-my-data-and-heres-what-i-found>

Two Main Attack Vectors

- ❑ Web browser
- ❑ Installed apps
- ❑ Both types of threats increasing in prevalence and sophistication



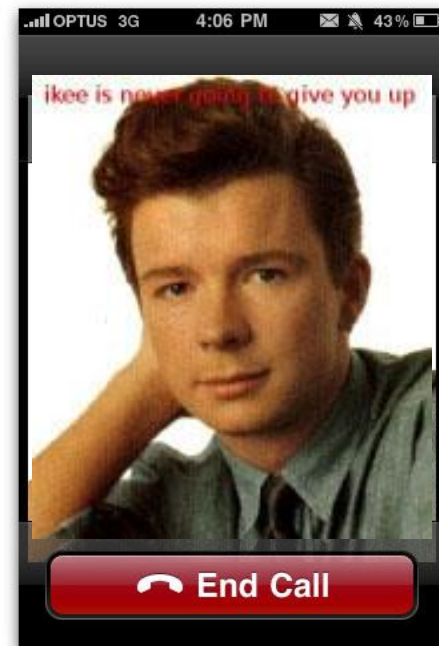
Mobile Malware Attack Vectors

- Unique to phones:
 - ▣ Premium SMS messages
 - ▣ Identify location
 - ▣ Record phone calls
 - ▣ Log SMS
- Similar to desktop/PCs:
 - ▣ Connects to botmasters
 - ▣ Steal data
 - ▣ Phishing
 - ▣ Malvertising



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 (could have been worse)
- 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



What's Different About Mobile Apps?

- App Store: **Approval** process for applications
 - ▣ Market: Vendor controlled/Open
 - ▣ App signing: Vendor-issued/self-signed
 - ▣ User approval of permission

- Programming language for applications
 - ▣ Managed execution: Java, .NET, Swift (most recent)
 - ▣ Native execution: Objective C

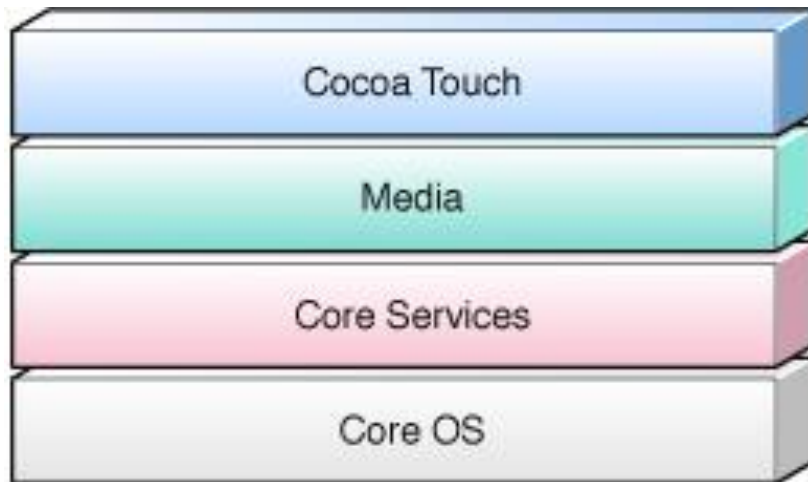
Apple iOS



From: iOS App Programming Guide

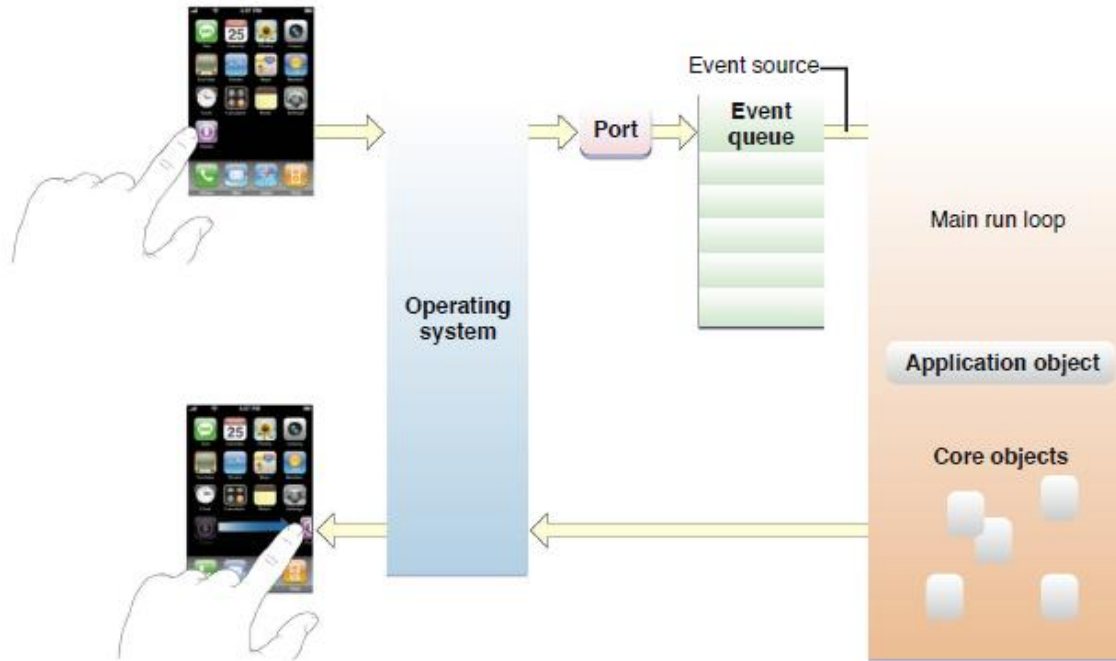
iOS Platform

8



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

iOS Application Development



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

Apple iOS Security

- Device security:
Prevent unauthorized use of the 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

Device Security: Passcodes

- ❑ Strong passcodes
- ❑ Passcode expiration
- ❑ Passcode reuse history
- ❑ Maximum failed attempts
- ❑ Over-the-air passcode enforcement
- ❑ Progressive passcode timeout

Data Security

- Hardware encryption
- Remote wipe
- Local wipe
- Encrypted Configuration Profiles
- Encrypted iTunes backups

“Along with **iOS 8**, Apple made some landmark privacy improvements to your devices, which Google matched with its Android platform only hours later. Your smartphone will soon be encrypted **by default**, and Apple or Google claim they will not be able open it for anyone”

Network Security

- Current accepted network security protocols
 - ▣ IPSec, L2TP, PPTP VPN
 - ▣ SSL VPN via App Store apps
 - ▣ SSL/TLS with X.509 certificates
 - ▣ WPA/WPA2 Enterprise with 802.1X

App Security

□ 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 an Apple-issued certificate

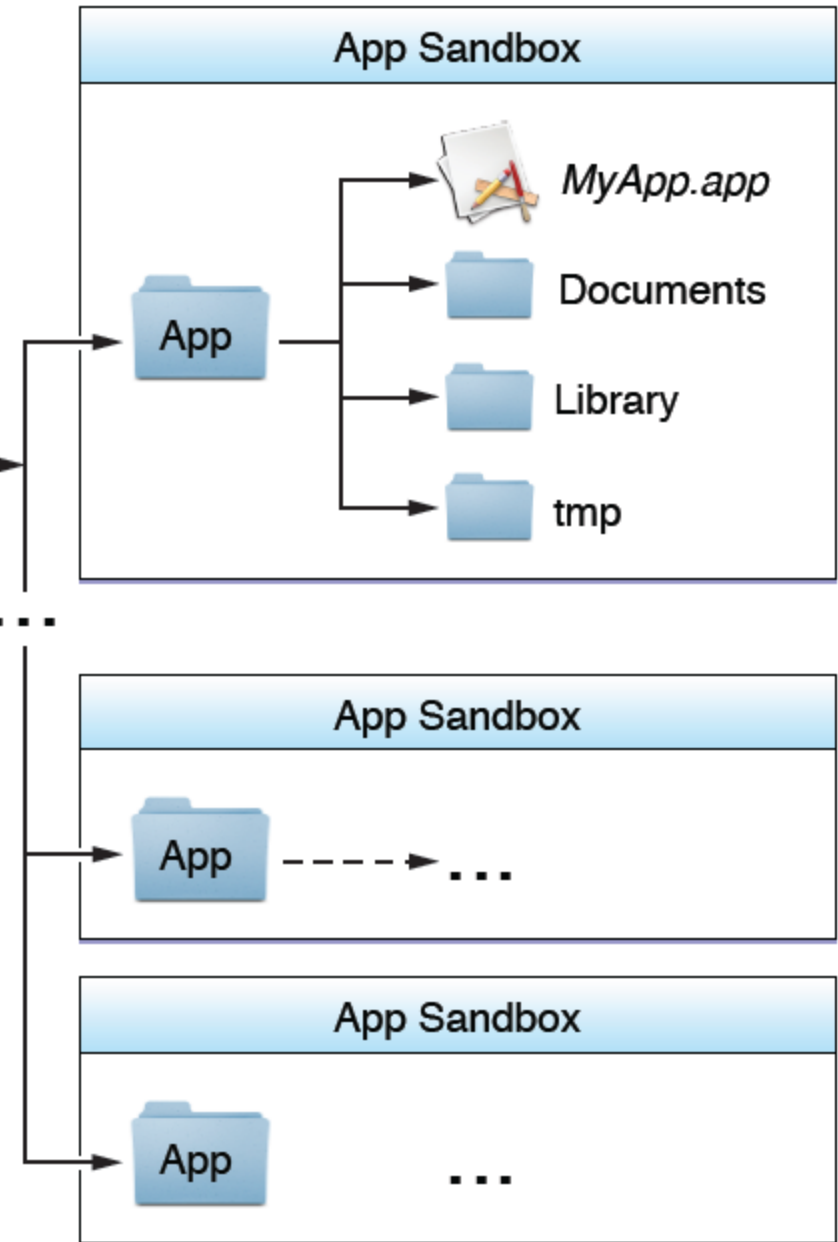
□ Application data protection

- ▣ Apps can take advantage of 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



iOS Permissions

16

- Contacts
- Microphone
- Calendars
- Camera
- Reminders
- HomeKit
- Photos
- Health
- Motion activity and fitness
- Speech recognition
- Location Services
- Bluetooth sharing
- Media Library
- Social media accounts, such as Twitter and Facebook
- ...

App Store Review Times

17

iOS App Store Review Times

Unofficial data, courtesy of AppReviewTimes.com
(14 day trailing average)



iOS App Store Process

18

- Reason #1: Insufficient information provided.....
- Reason #2: Presence of bugs and crashes.....
- Reason #3: Substandard user interface.....
- Reason #4: Non-relevant content.....
- Reason #5: Inappropriate naming of the App.....
- Reason #6: Non-compliant legal requirements.....
- Reason #7: Unauthorised mention of competitors....
- Reason #8: Non-compliant to privacy policies.....
- Reason #9: Inappropriate rating of the app.....
- Reason #10: Violation of Intellectual Property Rights

App Rejection

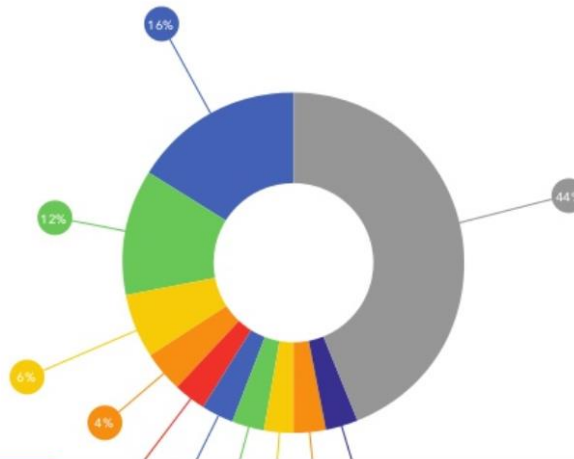
19

AN OVERVIEW OF REASONS WHY APPLE REJECTS APPS

Total Percent of App Rejections

56%
Top 10 Reasons

44%
Other Reasons (<3% each)



Privacy Violations Can Be Grounds for Rejection

20

17. Privacy

- 17.1 Apps cannot transmit data about a user without obtaining the user's prior permission and providing the user with access to information about how and where the data will be used
- 17.2 Apps that require users to share personal information, such as email address and date of birth, in order to function will be rejected
- 17.3 Apps may ask for date of birth (or use other age-gating mechanisms) only for the purpose of complying with applicable children's privacy statutes, but must include some useful functionality or entertainment value regardless of the user's age
- 17.4 Apps that collect, transmit, or have the capability to share personal information (e.g. name, address, email, location, photos, videos, drawings, the ability to chat, other personal data, or persistent identifiers used in combination with any of the above) from a minor must comply with applicable children's privacy statutes, and must include a privacy policy
- 17.5 Apps that include account registration or access a user's existing account must include a privacy policy or they will be rejected

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
 - ▣ Also: video stuff, Bluetooth, vibrate phone, etc.

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

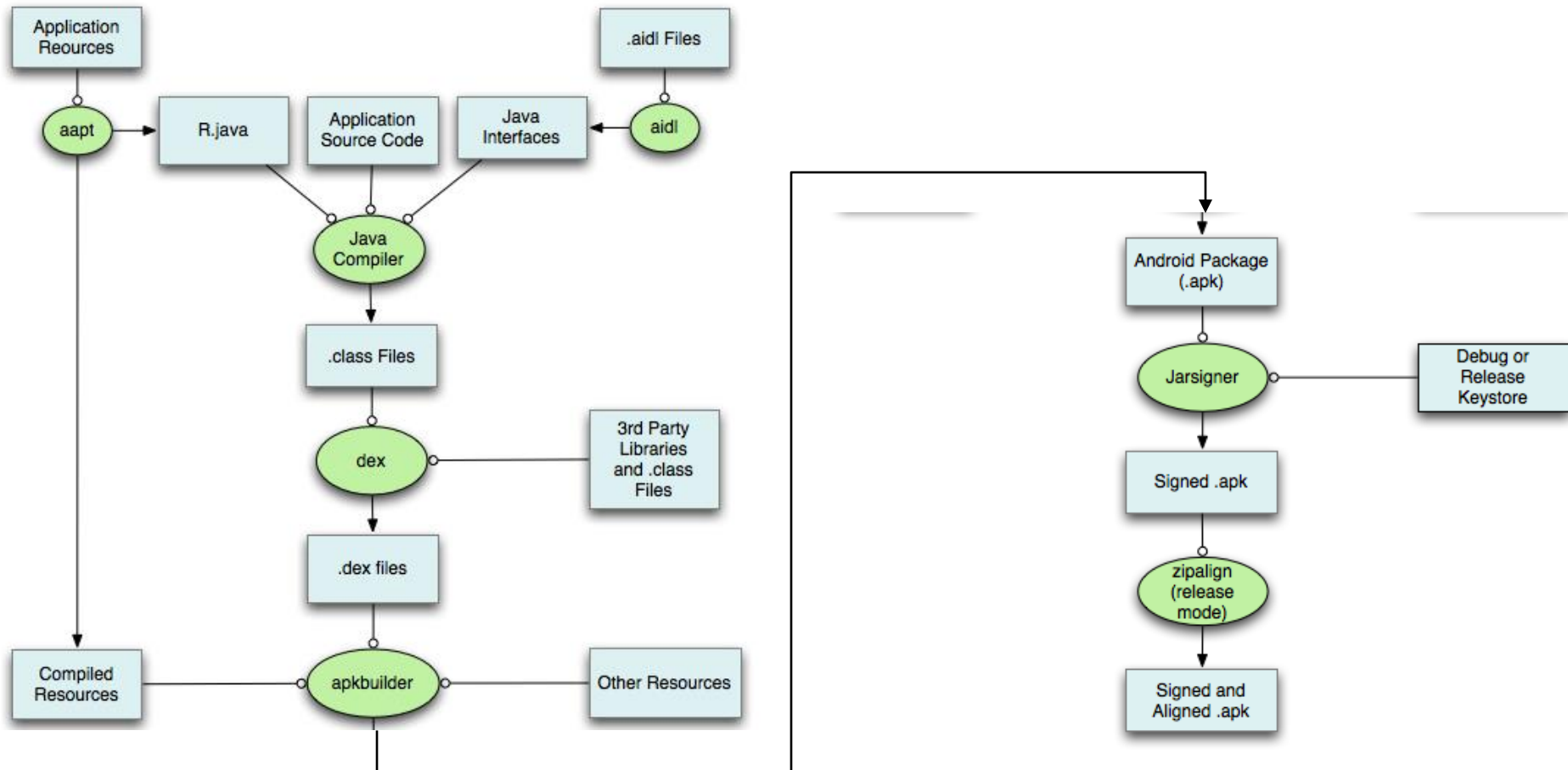
Android Market

- ❑ Self-signed apps
- ❑ Permissions granted on user installation
- ❑ Open
 - ▣ Bad applications may show up on market
 - ▣ Shifts focus from remote exploit to privilege escalation

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 Development Process



Application Development Concepts

- Activity – one-user task
 - ▣ Example: scroll through your inbox
 - ▣ Email client comprises many activities
- **Service** – Java daemon that runs in background:
 - ▣ Example: application that streams an mp3 in background
- Intents – asynchronous messaging system
 - ▣ Fire an intent to switch from one activity to another
 - ▣ Example: email app has inbox, compose activity, viewer activity
 - User click on inbox entry fires an intent to the viewer activity, which then allows user to view that email
- Content provider: Store and share data using a relational database interface
 - ▣ Broadcast receiver: “mailboxes” for messages from other applications

Exploit Prevention

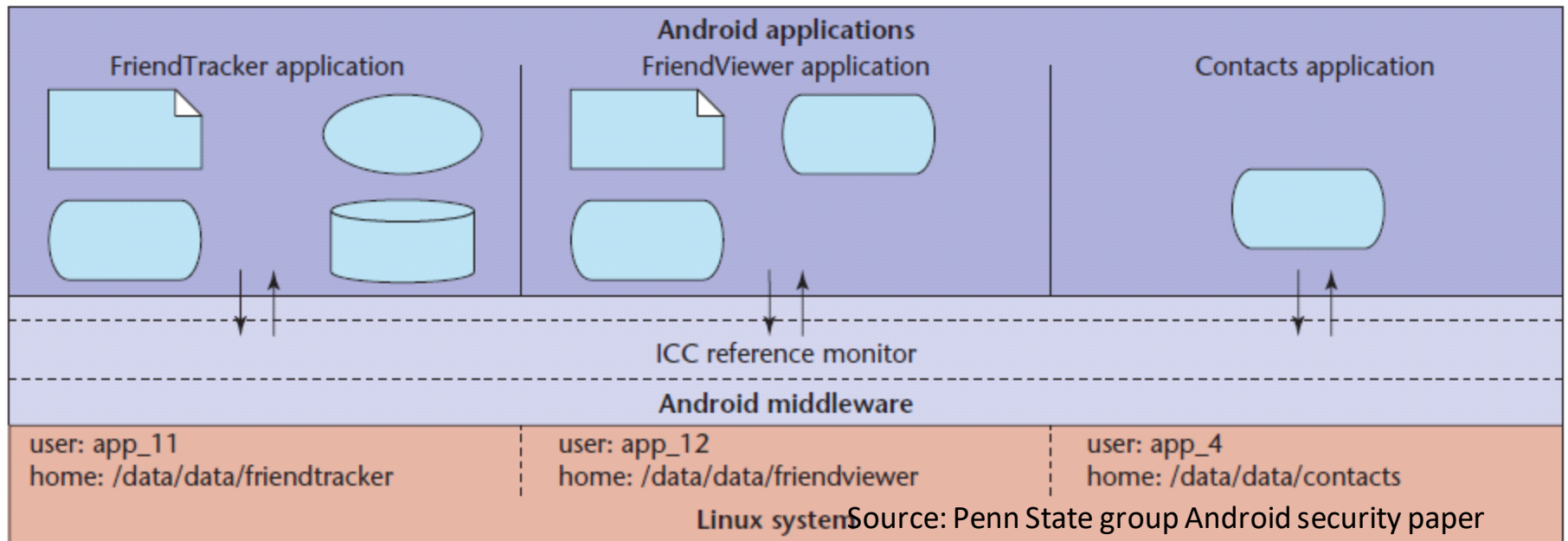
- 100 libraries + 500 million lines new code
 - ▣ Open source -> public review, no obscurity
- Goals
 - ▣ Prevent remote attacks, privilege escalation
 - ▣ Secure drivers, media codecs, new and custom features
- Overflow prevention
 - ▣ ProPolice stack protection: First on the ARM architecture
 - ▣ Some heap overflow protections: Chunk consolidation in DL malloc (from OpenBSD)
- ASLR: Avoided in initial release, but is now supported

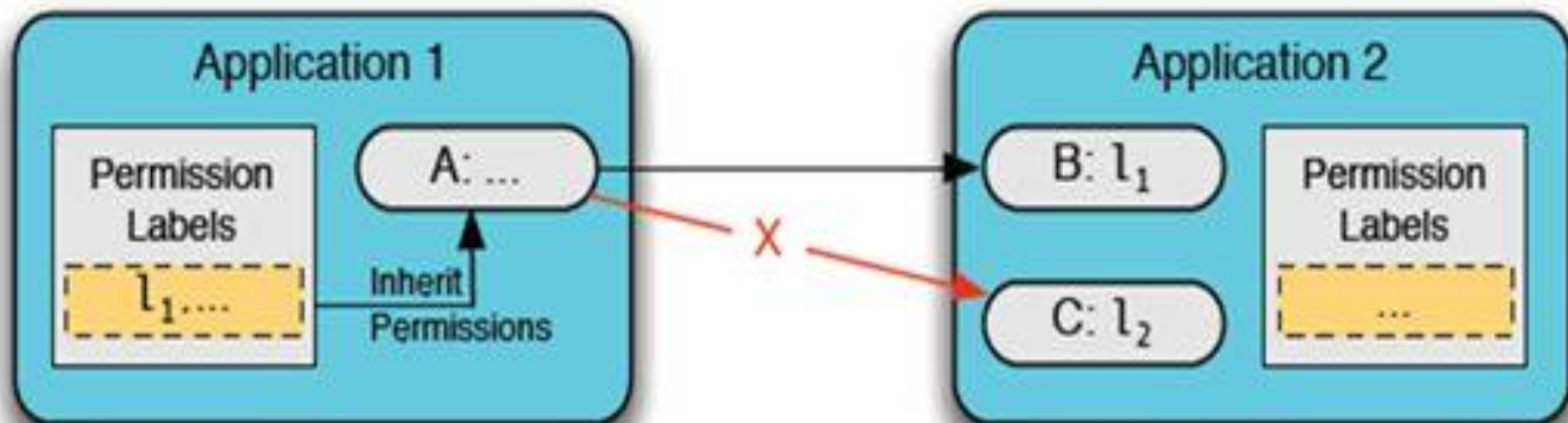
Application Sandbox

- 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 announces permission requirement
 - ▣ Create a whitelist model – user grants access: But don't want to ask user often – all questions asked as install time
 - ▣ Inter-component communication reference monitor checks permissions

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.

dlmalloc (Doug Lea)

- Stores meta data in band
- Heap consolidation attack
 - ▣ Heap overflow can overwrite pointers to previous and next unconsolidated chunks
 - ▣ Overwriting these pointers allows remote code execution
- Change to improve security
 - ▣ Check integrity of forward and backward pointers
 - Simply check that back-forward-back = back, f-b-f=f
 - ▣ Increases the difficulty of heap overflow

Java Sandbox

- Four complementary mechanisms
- Class loader
 - ▣ Separate namespaces for separate class loaders
 - ▣ Associates *protection domain* with each class
- Verifier and JVM run-time tests
 - ▣ NO unchecked casts or other type errors, NO array overflow
 - ▣ Preserves private, protected visibility levels
- Security Manager
 - ▣ Called by library functions to decide if request is allowed
 - ▣ Uses protection domain associated with code, user policy

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**

See also: <http://palisade.plynt.com/issues/2011Oct/android-vs-ios/>

34

Android App Privacy

Ad Libraries

35

Ad Library (version)	INTERNET	ACCESS_NETWORK_STATE	READ_PHONE_STATE	ACCESS_LOCATION	CAMERA	CALL_PHONE	WRITE_EXTERNAL_STORAGE	READ_CALENDAR	WRITE_CALENDAR	READ_CONTACTS	WRITE_CONTACTS	SEND_SMS	RECEIVE_BOOT_COMPLETE	GET_ACCOUNTS	READ_LOGS	ACCESS_WIFI_STATE
adfonic (1.1.4)	R	R		R												
admob (4.3.1)	R	R														
airpush (2-2012)	R	R	R	O									R			
buzzcity (1.0.5)	R		R			R										
greystripe (1.6.1)	R	R	R													
inmobi (3.0.1)	R	O		O		O		X	X							
jump tap (2.3)	R	R	R	O												
millennialmedia (4.5.1)	R	R	R		O		R									
mobclix (3.2.0)	R	O	R	X	X			X	X	X	X			X		
mOcean (2.9.1)	R	R	R	O	O	O	O	O	O			O			O	
smaato (2.5.4)	R	R	R	O												
vdopia (2.0.1)	R	R														
youmi (3.05)	R	R	R	R			R									X

TABLE I: *Ad SDK Permission Usage* As specified in their online documentation, ad libraries may require a permission (R) or declare it optional, but use it if available (O). Worryingly, some ad libraries check for and use undocumented permissions (X).

From “Investigating User Privacy in Android Ad Libraries”

Ad Libraries Steal Private Data

36

Ad Library (version)	App Package Name	GPS Coordinates	Connection Type	Device Make and Model	Wireless Carrier	Age	Gender	Income Level	Keywords
adfonic (1.1.4)		P		P		D	D		D
admob (4.3.1)	A	D				D	D		D
airpush (Feb 2012)	A	P		A	A				
buzzcity (1.0.5)									
greystripe (1.6.1)									
inmobi (3.0.1)	A	P	P	A		D	D		D
jumptap (2.3)		P				D	D	D	
millennialmedia (4.5.1)						D	D	D	D
mobclix (3.2.0)									D
mOcean (2.9.1)	A	P			D	D	D	D	D
smaato (2.5.4)	A	P	P		P	D	D	D	D
vdopia (2.0.1)			A	A					
youmi (3.05)	A								

TABLE II: *Private Data in Ad Requests* Some fields ad libraries will always populate in ad requests (A) while others it will populate only when the application has the appropriate permissions (P), both automatically. Alternatively, some ad libraries choose to only populate fields when the developer explicitly passes the value to the library (D).

Ad Libraries for Popularity

37

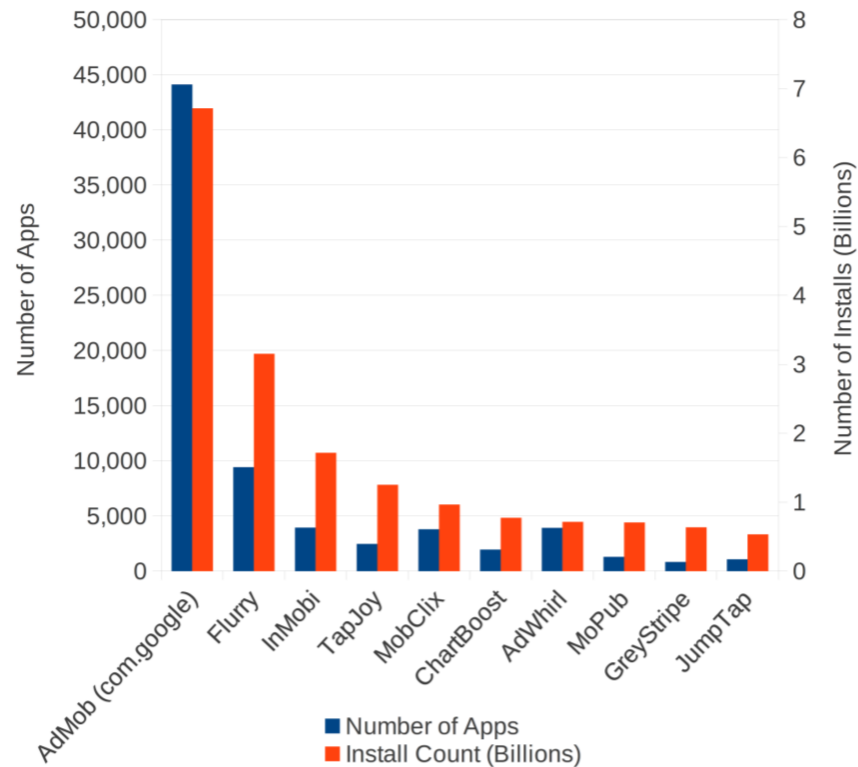


Fig. 4. Top 10 Libraries by Installs. The number of apps containing a given library is presented on the primary Y axis, and the number of installs for a given library on the secondary Y axis.

Libraries and Permissions

	Library Name	Number of Apps	Install Count	Number of Permissions	Package Name
1	AdMob (com.google)	44,107	6,711,025,298	5	com.google.ads
2	Flurry	9,411	3,150,987,621	5	com.flurry
3	InMobi	3,933	1,713,911,340	7	com.inmobi
4	TapJoy	2,456	1,250,599,000	5	com.tapjoy
5	MobClix	3,782	963,004,760	15	com.mobclix
6	ChartBoost	1,935	772,087,100	4	com.chartboost
7	AdWhirl	3,913	712,401,775	4	com.adwhirl
8	MoPub	1,283	703,792,000	5	com.mopub
9	GreyStripe	816	633,353,550	4	com.greystripe
10	JumpTap	1,054	530,924,930	6	com.jumptap
11	Google Analytics	1,258	530,527,776	2	com.google.analytics
12	AdMob (com.admob)	6,931	507,426,872	5	com.admob
13	Burstly	105	433,160,050	7	com.burstly
14	SponsorPay	454	393,370,000	3	com.sponsorpay
15	Cauly	1,504	367,011,400	7	com.cauly
16	MobFox	1,931	356,714,780	6	com.mobfox
17	Vpon.com	687	315,392,200	6	com.vpon
18	AppBrain	2,081	311,924,110	2	com.appbrain
19	Daum.net	1,888	310,528,010	11	net.daum
20	AdMarvel	611	244,899,550	6	com.admarvel
21	AppLovin	987	244,547,600	9	com.applovin
22	AdFonic	767	225,023,320	5	com.adfonic
23	MdotM	637	218,410,710	5	com.mdotm
24	GetJar	255	191,135,000	6	com.getjar

TaintDroid: System for Realtime Privacy Monitoring on Smartphones

William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth

**Based on slides presented by
Yang Sun
Jiayan Guo and original TaintDroid slides**

Summary

40

- TaintDroid provides efficient, system-wide, dynamic taint tracking and analysis for Android.
- We found 20 of the 30 studied applications to share information in a way that was not expected.

Smartphone Platform Market Share

41

Top Smartphone Platforms

3 Month Avg. Ending Jan. 2014 vs. 3 Month Avg. Ending Oct. 2013

Total U.S. Smartphone Subscribers Age 13+

Source: comScore MobiLens

	Share (%) of Smartphone Subscribers		
	Oct-13	Jan-14	Point Change
<i>Total Smartphone Subscribers</i>	100.0%	100.0%	N/A
Android	52.2%	51.7%	-0.5
Apple	40.6%	41.6%	1.0
BlackBerry	3.6%	3.1%	-0.5
Microsoft	3.2%	3.2%	0.0
Symbian	0.2%	0.2%	0.0

Challenges

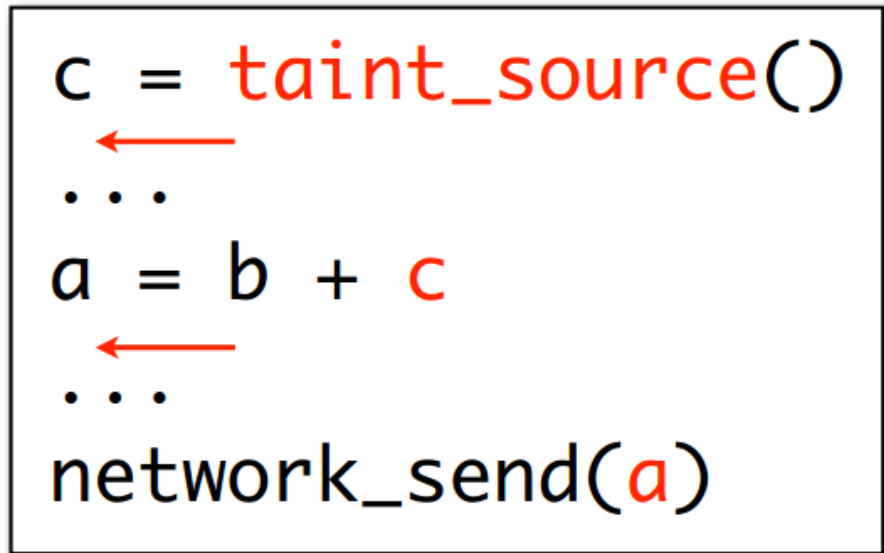
42

- **Goal:** Monitor app behavior to determine when privacy sensitive information leaves the phone.
- **Challenges ...**
 - Smartphones are resource constrained
 - Third-party applications are entrusted with several types of privacy sensitive information
 - Context-based privacy information is dynamic and can be difficult to identify even when sent in the clear
 - Applications can share information

Dynamic Taint Analysis

43

- Dynamic taint analysis is a technique that tracks information dependencies from an origin.
- Conceptual idea:
 - Taint source
 - Taint propagation
 - Taint sink



Approach to Taint Tracking

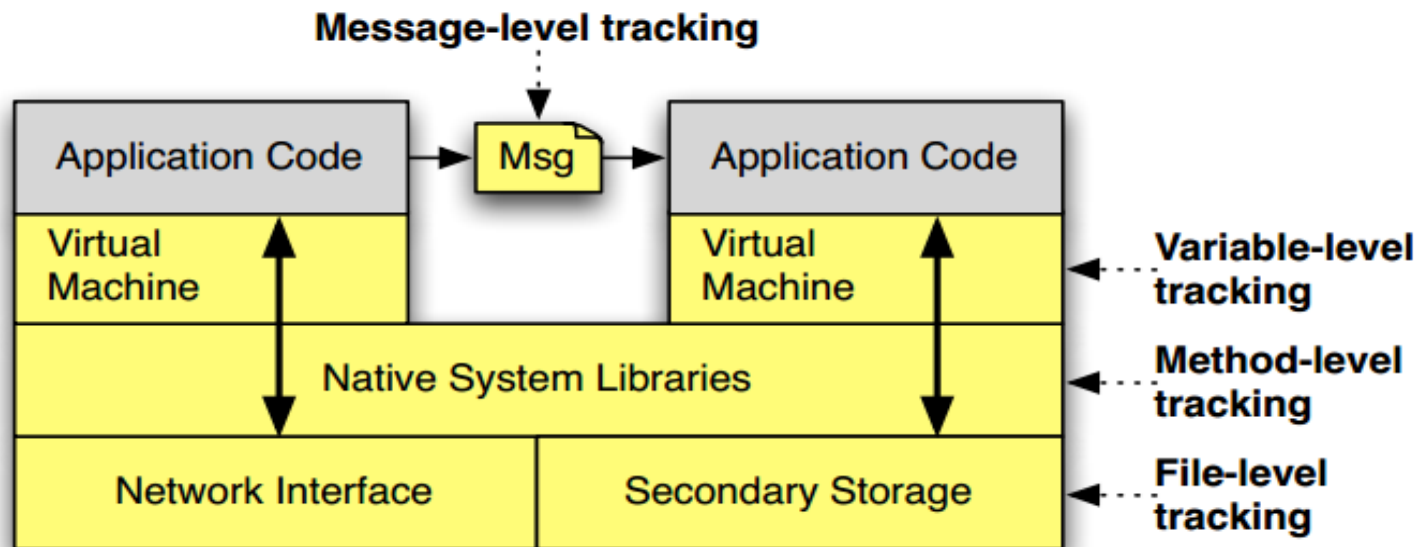
44

A novel, efficient, system-wide, multiple-marking, taint tracking design by combining multiple granularities of information tracking.

- **Variable-level**: VM interpreter provide variable-level tracking within untrusted application code.
- **Method-level**: System-provide native libraries.
- **Message-level**: Message-level tracking between applications.
- **File-level**: file-level tracking to ensure persistent information conservatively retains its taint markings.

Continue

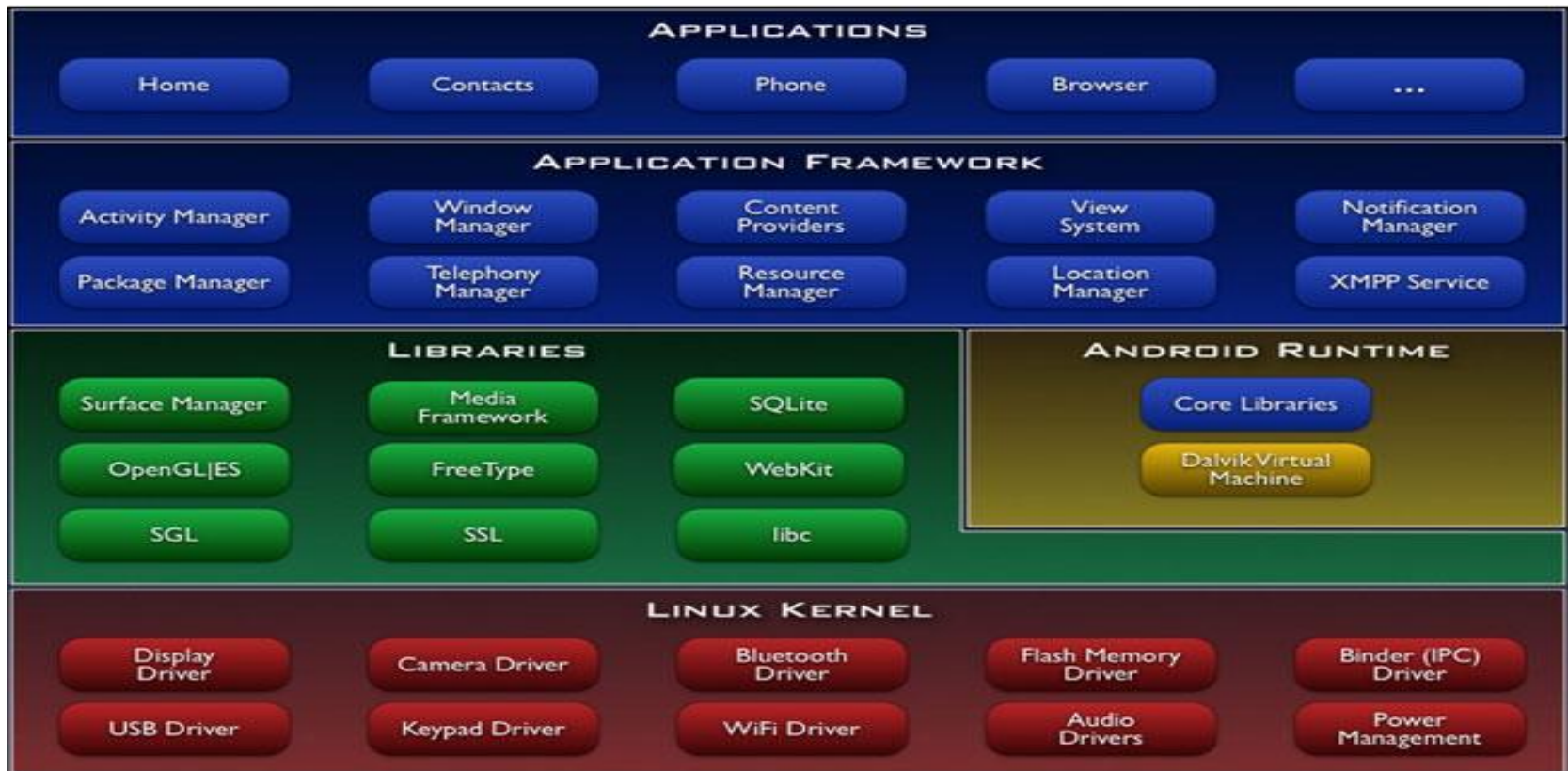
45



Multi-level approach for performance efficient taint tracking within a common smartphone architecture

Architecture of Android System

46



Tracking System

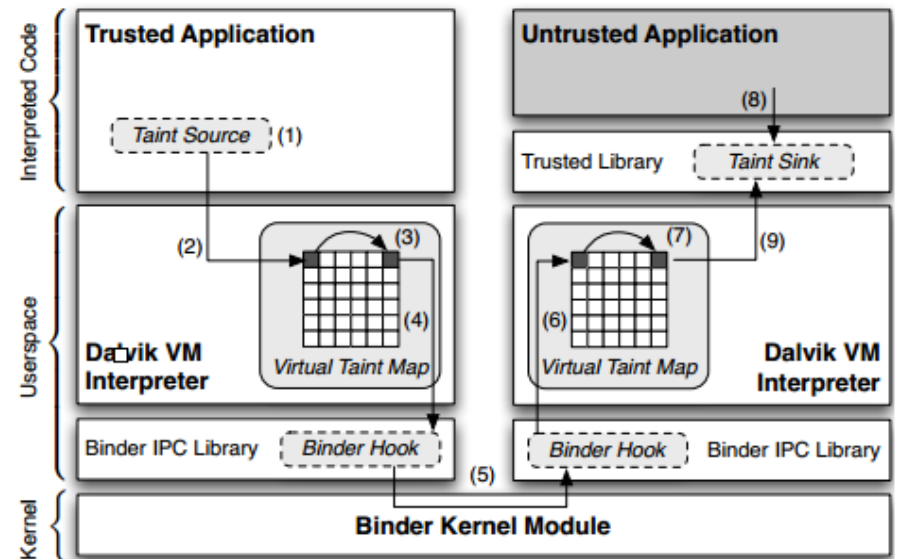
47

- **Dalvik VM Interpreter:** Dalvik EXecutable (DEX) byte code is a register-based machine language. The Dalvik VM interpreter manages method registers with an internal execution state stack.
- **Native Method:** Native methods are written in C/C++ and expose functionality provided by the underlying Linux kernel and services. They can also access Java internals, and hence are including in our trusted computing base.
- **Binder IPC:** All Android IPC occurs through binder. Binder is a component-based processing and IPC framework designed for BeOS, extended by Palm Inc., and customized for Android by Google.

TaintDroid

48

TaintDroid is a realization of our multiple granularity taint tracking approach within Android.



TaintDroid architecture within Android.

Implementing Challenges

49

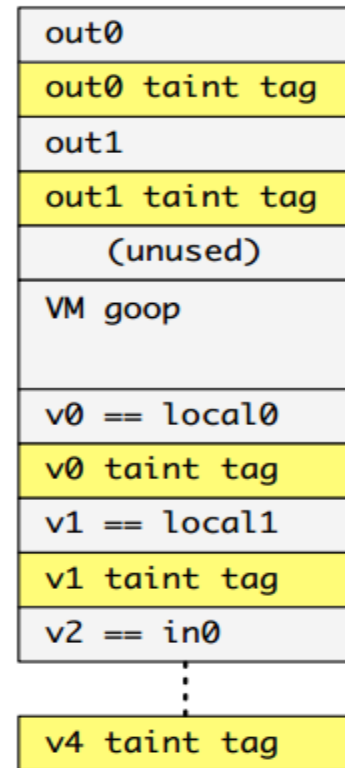
- a) Taint tag storage
- b) Interpreted code taint propagation
- c) Native code taint propagation
- d) IPC taint propagation
- e) Secondary storage taint propagation

Taint Tag Storage

50

Modified the Dalvik VM interpreter to store and propagate taint tags (a taint bit-vector) on variables.

- **Local variables and args**: taint tags stored adjacent to variables on the internal execution stack.
 - 64-bit variables span 32-bit storage
- **Class fields**: similar to locals, but inside static and instance field heap objects
- **Arrays**: one taint tag per array to minimize overhead



DEX Taint Propagation Logic

51



Table 1: DEX Taint Propagation Logic. Register variables and class fields are referenced by v_X and f_X , respectively. R and E are the return and exception variables maintained within the interpreter. A , B , and C are byte-code constants.

Op Format	Op Semantics	Taint Propagation	Description
<i>const-op</i> $v_A C$	$v_A \leftarrow C$	$\tau(v_A) \leftarrow \emptyset$	Clear v_A taint
<i>move-op</i> $v_A v_B$	$v_A \leftarrow v_B$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
<i>move-op-R</i> v_A	$v_A \leftarrow R$	$\tau(v_A) \leftarrow \tau(R)$	Set v_A taint to return taint
<i>return-op</i> v_A	$R \leftarrow v_A$	$\tau(R) \leftarrow \tau(v_A)$	Set return taint (\emptyset if void)
<i>move-op-E</i> v_A	$v_A \leftarrow E$	$\tau(v_A) \leftarrow \tau(E)$	Set v_A taint to exception taint
<i>throw-op</i> v_A	$E \leftarrow v_A$	$\tau(E) \leftarrow \tau(v_A)$	Set exception taint
<i>unary-op</i> $v_A v_B$	$v_A \leftarrow \otimes v_B$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
<i>binary-op</i> $v_A v_B v_C$	$v_A \leftarrow v_B \otimes v_C$	$\tau(v_A) \leftarrow \tau(v_B) \cup \tau(v_C)$	Set v_A taint to v_B taint \cup v_C taint
<i>binary-op</i> $v_A v_B$	$v_A \leftarrow v_A \otimes v_B$	$\tau(v_A) \leftarrow \tau(v_A) \cup \tau(v_B)$	Update v_A taint with v_B taint
<i>binary-op</i> $v_A v_B C$	$v_A \leftarrow v_B \otimes C$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
<i>aput-op</i> $v_A v_B v_C$	$v_B[v_C] \leftarrow v_A$	$\tau(v_B[\cdot]) \leftarrow \tau(v_B[\cdot]) \cup \tau(v_A)$	Update array v_B taint with v_A taint
<i>aget-op</i> $v_A v_B v_C$	$v_A \leftarrow v_B[v_C]$	$\tau(v_A) \leftarrow \tau(v_B[\cdot]) \cup \tau(v_C)$	Set v_A taint to array and index taint
<i>sput-op</i> $v_A f_B$	$f_B \leftarrow v_A$	$\tau(f_B) \leftarrow \tau(v_A)$	Set field f_B taint to v_A taint
<i>sget-op</i> $v_A f_B$	$v_A \leftarrow f_B$	$\tau(v_A) \leftarrow \tau(f_B)$	Set v_A taint to field f_B taint
<i>iput-op</i> $v_A v_B f_C$	$v_B(f_C) \leftarrow v_A$	$\tau(v_B(f_C)) \leftarrow \tau(v_A)$	Set field f_C taint to v_A taint
<i>iget-op</i> $v_A v_B f_C$	$v_A \leftarrow v_B(f_C)$	$\tau(v_A) \leftarrow \tau(v_B(f_C)) \cup \tau(v_B)$	Set v_A taint to field f_C and object reference taint

Example

52

```
public static Integer valueOf(int i) {
    if (i < -128 || i > 127) {
        return new Integer(i); }
    return valueOfCache.CACHE [i+128];
}
static class valueOfCache {
    static final Integer[] CACHE = new Integer[256];
    static {
        for(int i=-128; i<=127; i++) {
            CACHE[i+128] = new Integer(i); } }
}

Object intProxy(int val) { return val; }
int out = (Integer) intProxy(tVal);
```

val <- tval (tval=1)

53

Op Format	Op Semantics	Taint Propagation	Description
<i>const-op</i> $v_A C$	$v_A \leftarrow C$	$\tau(v_A) \leftarrow \emptyset$	Clear v_A taint
<i>move-op</i> $v_A v_B$	$v_A \leftarrow v_B$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
<i>move-op-R</i> v_A	$v_A \leftarrow R$	$\tau(v_A) \leftarrow \tau(R)$	Set v_A taint to return taint
<i>return-op</i> v_A	$R \leftarrow v_A$	$\tau(R) \leftarrow \tau(v_A)$	Set return taint (\emptyset if void)
<i>move-op-E</i> v_A	$v_A \leftarrow E$	$\tau(v_A) \leftarrow \tau(E)$	Set v_A taint to exception taint
<i>throw-op</i> v_A	$E \leftarrow v_A$	$\tau(E) \leftarrow \tau(v_A)$	Set exception taint
<i>unary-op</i> $v_A v_B$	$v_A \leftarrow \otimes v_B$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
<i>binary-op</i> $v_A v_B v_C$	$v_A \leftarrow v_B \otimes v_C$	$\tau(v_A) \leftarrow \tau(v_B) \cup \tau(v_C)$	Set v_A taint to v_B taint \cup v_C taint
<i>binary-op</i> $v_A v_B$	$v_A \leftarrow v_A \otimes v_B$	$\tau(v_A) \leftarrow \tau(v_A) \cup \tau(v_B)$	Update v_A taint with v_B taint
<i>binary-op</i> $v_A v_B C$	$v_A \leftarrow v_B \otimes C$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
<i>aput-op</i> $v_A v_B v_C$	$v_B[v_C] \leftarrow v_A$	$\tau(v_B[\cdot]) \leftarrow \tau(v_B[\cdot]) \cup \tau(v_A)$	Update array v_B taint with v_A taint
<i>aget-op</i> $v_A v_B v_C$	$v_A \leftarrow v_B[v_C]$	$\tau(v_A) \leftarrow \tau(v_B[\cdot]) \cup \tau(v_C)$	Set v_A taint to array and index taint
<i>sput-op</i> $v_A f_B$	$f_B \leftarrow v_A$	$\tau(f_B) \leftarrow \tau(v_A)$	Set field f_B taint to v_A taint
<i>sget-op</i> $v_A f_B$	$v_A \leftarrow f_B$	$\tau(v_A) \leftarrow \tau(f_B)$	Set v_A taint to field f_B taint
<i>iput-op</i> $v_A v_B f_C$	$v_B(f_C) \leftarrow v_A$	$\tau(v_B(f_C)) \leftarrow \tau(v_A)$	Set field f_C taint to v_A taint
<i>iget-op</i> $v_A v_B f_C$	$v_A \leftarrow v_B(f_C)$	$\tau(v_A) \leftarrow \tau(v_B(f_C)) \cup \tau(v_B)$	Set v_A taint to field f_C and object reference taint

Continue

54



Op Format	Op Semantics	Taint Propagation	Description
<i>const-op</i> $v_A C$	$v_A \leftarrow C$	$\tau(v_A) \leftarrow \emptyset$	Clear v_A taint
<i>move-op</i> $v_A v_B$	$v_A \leftarrow v_B$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
<i>move-op-R</i> v_A	$v_A \leftarrow R$	$\tau(v_A) \leftarrow \tau(R)$	Set v_A taint to return taint
<i>return-op</i> v_A	$R \leftarrow v_A$	$\tau(R) \leftarrow \tau(v_A)$	Set return taint (\emptyset if void)
<i>move-op-E</i> v_A	$v_A \leftarrow E$	$\tau(v_A) \leftarrow \tau(E)$	Set v_A taint to exception taint
<i>throw-op</i> v_A	$E \leftarrow v_A$	$\tau(E) \leftarrow \tau(v_A)$	Set exception taint
<i>unary-op</i> $v_A v_B$	$v_A \leftarrow \otimes v_B$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
<i>binary-op</i> $v_A v_B v_C$	$v_A \leftarrow v_B \otimes v_C$	$\tau(v_A) \leftarrow \tau(v_B) \cup \tau(v_C)$	Set v_A taint to v_B taint \cup v_C taint
<i>binary-op</i> $v_A v_B$	$v_A \leftarrow v_A \otimes v_B$	$\tau(v_A) \leftarrow \tau(v_A) \cup \tau(v_B)$	Update v_A taint with v_B taint
<i>binary-op</i> $v_A v_B C$	$v_A \leftarrow v_B \otimes C$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
<i>aput-op</i> $v_A v_B v_C$	$v_B[v_C] \leftarrow v_A$	$\tau(v_B[\cdot]) \leftarrow \tau(v_B[\cdot]) \cup \tau(v_A)$	Update array v_B taint with v_A taint
<i>aget-op</i> $v_A v_B v_C$	$v_A \leftarrow v_B[v_C]$	$\tau(v_A) \leftarrow \tau(v_B[\cdot]) \cup \tau(v_C)$	Set v_A taint to array and index taint
<i>sput-op</i> $v_A f_B$	$f_B \leftarrow v_A$	$\tau(f_B) \leftarrow \tau(v_A)$	Set field f_B taint to v_A taint
<i>sget-op</i> $v_A f_B$	$v_A \leftarrow f_B$	$\tau(v_A) \leftarrow \tau(f_B)$	Set v_A taint to field f_B taint
<i>iput-op</i> $v_A v_B f_C$	$v_B(f_C) \leftarrow v_A$	$\tau(v_B(f_C)) \leftarrow \tau(v_A)$	Set field f_C taint to v_A taint
<i>iget-op</i> $v_A v_B f_C$	$v_A \leftarrow v_B(f_C)$	$\tau(v_A) \leftarrow \tau(v_B(f_C)) \cup \tau(v_B)$	Set v_A taint to field f_C and object reference taint

Native Code Taint Propagation

55

Rules: 1) all accessed external variables are assigned taint tags.

2) the return values is assigned a taint tag.

JNI Methods: JNI methods are invoked through the JNI call bridge.

- TaintDroid uses a combination of heuristics and method profiles to patch VM tracking state.
 - A method profile is a list of (from, to) pairs indicating flows between variables, which may be method parameters, class variables, or return values.
 - The heuristic is conservative for JNI methods that only operate on primitive and String arguments and return values.

IPC and File Propagation

56

- TaintDroid uses message-level taint tracking. A message taint tag represents the upper bound of taint markings assigned to variables contained in the message.
- Persistent storage tracked at the [file level](#).
The design stores one taint tag per file. The taint tag is updated on file write and propagated to data on file read.

Privacy Hook Placement











57

- **Low-bandwidth Sensors:** placed hooks in Android's LocationManager and SensorManager applications
- **High-bandwidth Sensors:** placed hooks for both data buffer and file tainting for tracking microphone and camera information.
- **Information Databases:** address book, SMS storage
- **Device Identifiers:** the phone number, SIM card identifiers (IMSI, ICC-ID), and device identifier (IMEI)
- **Network Taint Sink:** privacy analysis identifies when **tainted** information transmits out the network interface.

Experimental Setup

58

- 2010 Android Market 1,100 apps
- 358 of 1,100 required Internet permissions
- Most popular 50 of 358(8.4%) from 12 categories
- Randomly select 30 out of 50

applications	#	permissions
The Weather Channel, Cetos, Solitarie, Movies, Babble, Manga Browser	6	
Bump, Wertago, Antivirus, ABC --- Animals, Traffic Jam, Hearts, Blackjack, Horoscope, 3001 Wisdom Quotes Lite, Yellow Pages, Datelefonbuch, Astrid, BBC News Live Stream, Ringtones	14	 
Layer, Knocking, Coupons, Trapster, Spongebot Slide, ProBasketBall	6	  
MySpace, Barcode Scanner, ixMAT	3	
Evernote	1	  

Result Summary

59

Table 3: Potential privacy violations by 20 of the studied applications. Note that three applications had multiple violations, one of which had a violation in all three categories.

Observed Behavior (# of apps)	Details
Phone Information to Content Servers (2)	2 apps sent out the phone number, IMSI, and ICC-ID along with the geo-coordinates to the app's content server.
Device ID to Content Servers (7)*	2 Social, 1 Shopping, 1 Reference and three other apps transmitted the IMEI number to the app's content server.
Location to Advertisement Servers (15)	5 apps sent geo-coordinates to ad.qwapi.com, 5 apps to admob.com, 2 apps to ads.mobclix.com (1 sent location both to admob.com and ads.mobclix.com) and 4 apps sent location [†] to data.flurry.com.

Findings: Phone Identifiers

60

- 7 applications sent device ID (IMEI) and 2 apps sent phone info (Ph. #, IMSI*, ICC-ID) to a remote server without informing the user
 - one application transmits the phone information **every time** the phone **boots**. This application transmits the phone data immediately after install, before first use.
 - One application **displays** a privacy statement that clearly indicates that the application collects the device ID
 - other uses the **hash** of the IMEI instead of the number itself

Findings: Location

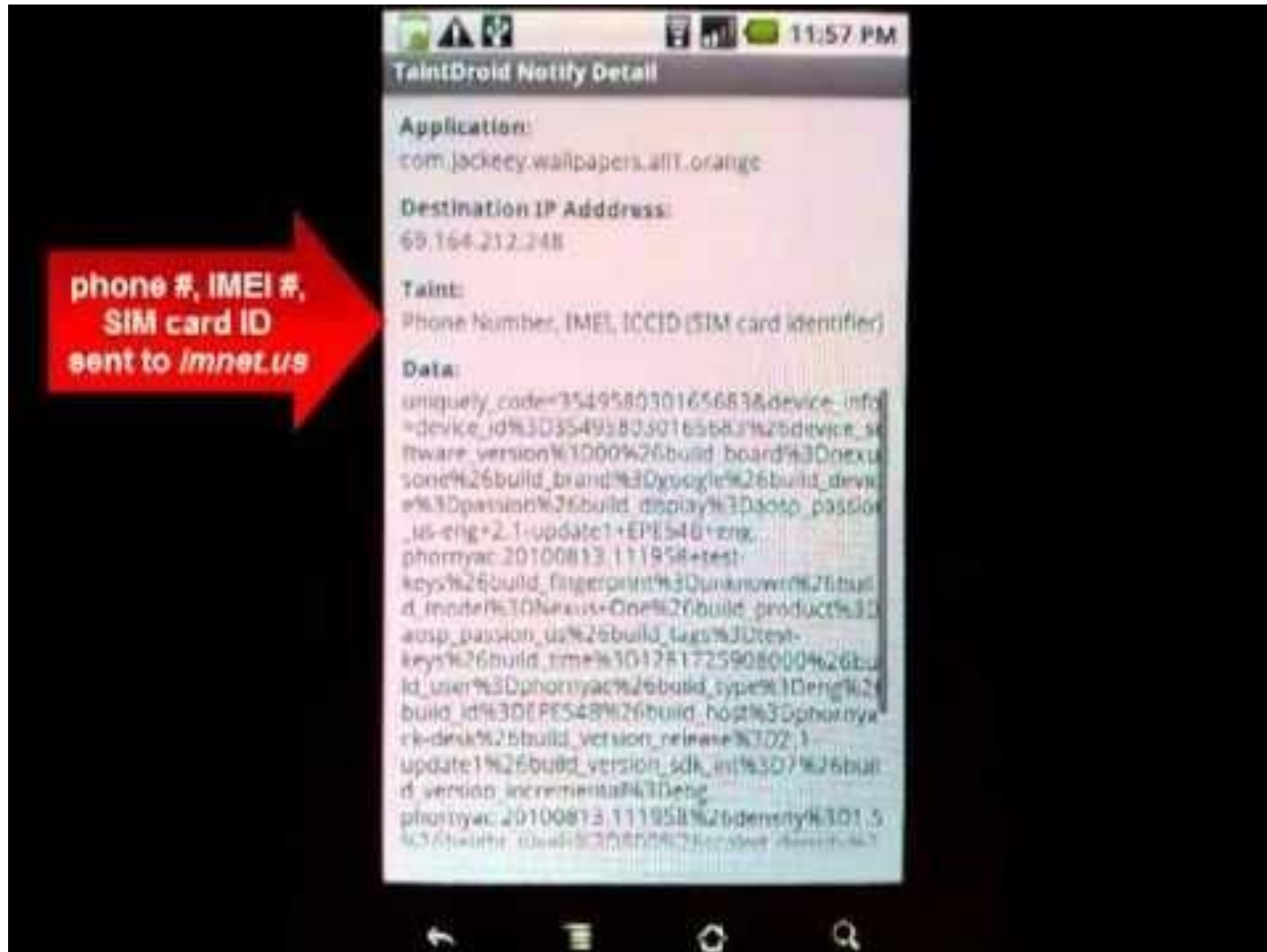
61

- 15 of the 30 applications shared physical location with an ad server (admob.com, ad.qwapi.com, ads.mobclix.com, data.flurry.com)
- Exposure of location information occurred both in plaintext (11) and in binary format (4).

```
...&s=a14a4a93f1e4c68&..&t=062A1CB1D476DE85  
B717D9195A6722A9&d%5Bcoord%5D=47.6612278900  
00006%2C-122.31589477&...
```

Demo

62



Demo available at <http://youtu.be/qnLujX1Dw4Y>

63

