TAINTDROID: AN INFORMATION-FLOW TRACKING SYSTEM FOR REALTIME PRIVACY MONITORING ON SMARTPHONES

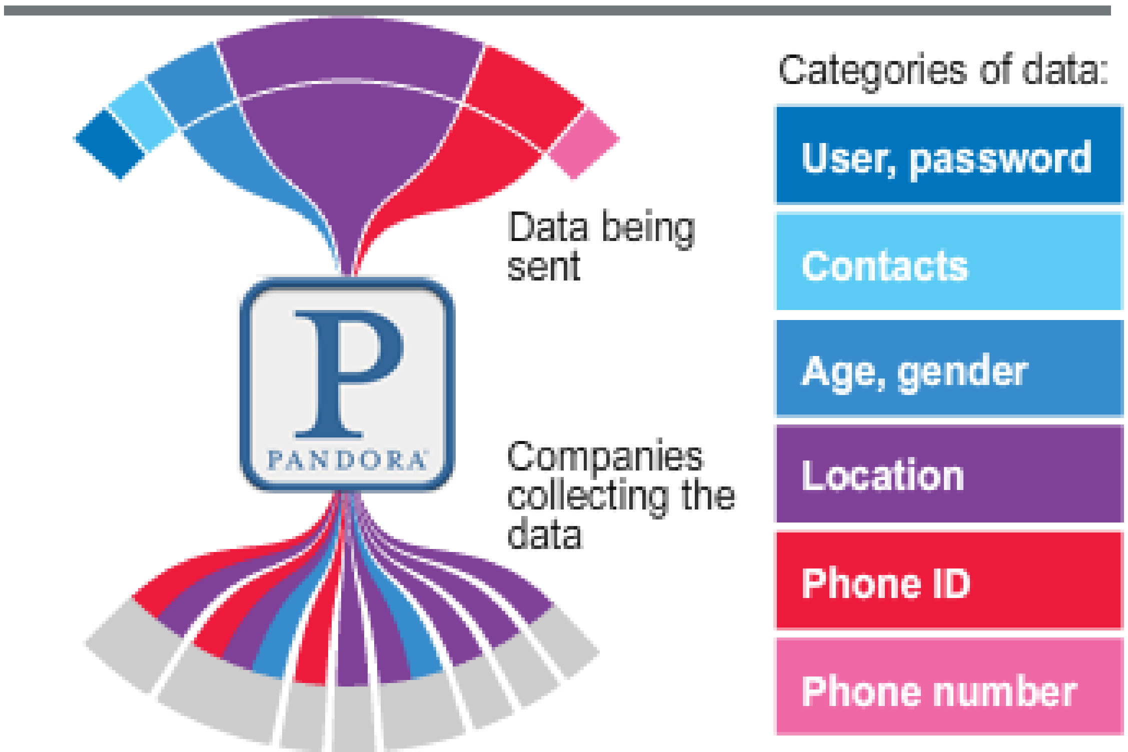
Byung-Gon Chun
Intel Labs Berkeley

Joint work with Jaeyeon Jung (ILS), Anmol Seth (ILS), William Enck (PSU), Patrick McDaniel (PSU), Landon Cox (Duke), Peter Gilbert (Duke)

Intel Labs Berkeley

- A lablet located at Berkeley next to the UC Berkeley campus
- Exploratory research
- An open collaborative model
- Systems/networking, security, programming language, machine learning, HCI

Smartphone Privacy Risks Posed by Third-party Apps



(Credit: WSJ)

Smartphone Privacy Risks Posed by Third-party Apps

NETWORKWORLD

Many Android apps leak user privacy data

Researchers find permitted apps transmit phone numbers, location, and SIM card IDs

BBC

Google Android apps found to be sharing data

cnet News

September 29, 2010 6:52 PM PDT

What's that Android app doing with my data?

Slashdot

Your Rights Online: **Many More Android Apps Leaking User Data**

The Register®

2 out of 3 Android apps use private data 'suspiciously'

Google protections 'insufficient'

msnbc.com

Smartphone Apps Spread Personal Info, Study Finds

WIRED

Study Shows Some Android Apps Leak User Data Without Clear Notifications

A Movie

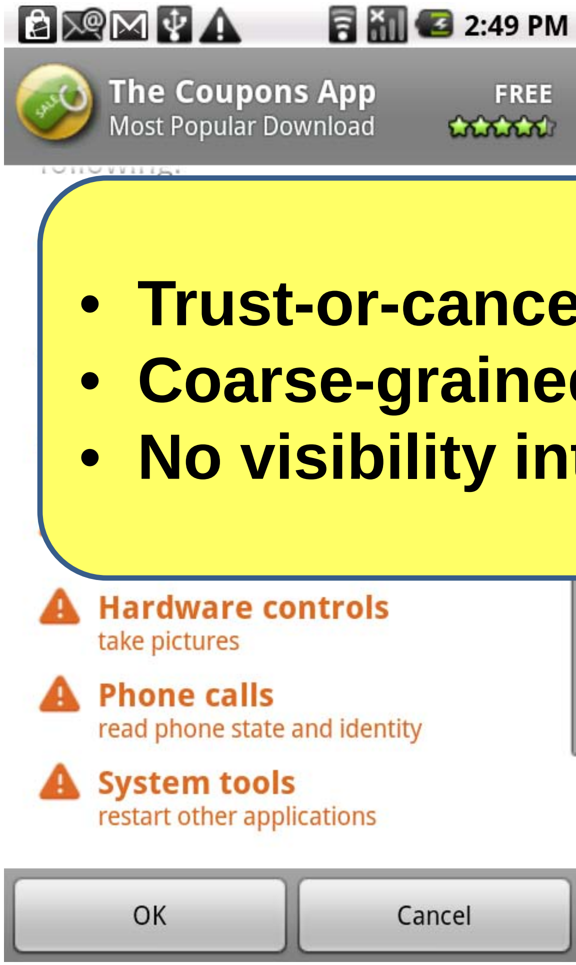
Roadmap

- Motivation
- Our approach
- TaintDroid design
- Performance study
- Application study
- Other research work

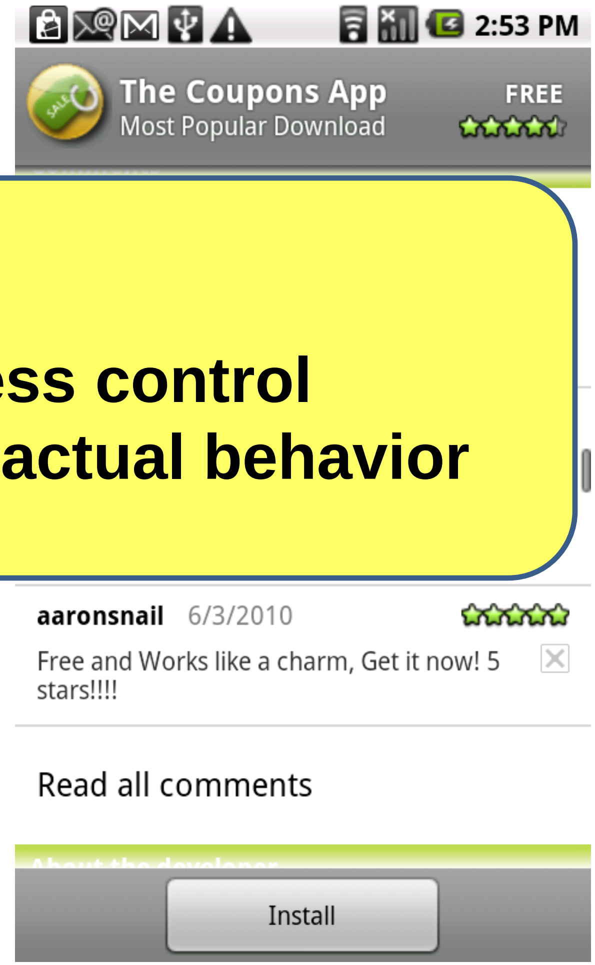
TaintDroid Goal

Monitor app behavior to determine when privacy sensitive information leaves the phone in real time

Current “Best” Practice



- **Trust-or-cancel**
- **Coarse-grained access control**
- **No visibility into the actual behavior**



8

Our Approach

- Look inside of applications to watch how they use privacy sensitive data
- Trust-or-cancel → Trust-but-verify

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 when sent
- Applications can share information

Dynamic Taint Analysis

- A technique that tracks information dependencies from an origin
- Taint
  - Source
  - Propagation
  - Sink

```
C = Taint_source()
…
A = B + C
…
Network_send(A)
```

Dynamic Taint Analysis in Action

MOV B,A

instrumentation

backup register state

SetTaint (B, GetTaint(A))

restore register state

MOV B,A

register

0

0

0

- Expensive! 2-20x slowdown.
- Overtainting/undertainting problems.
- Whole-system tracking!

memory

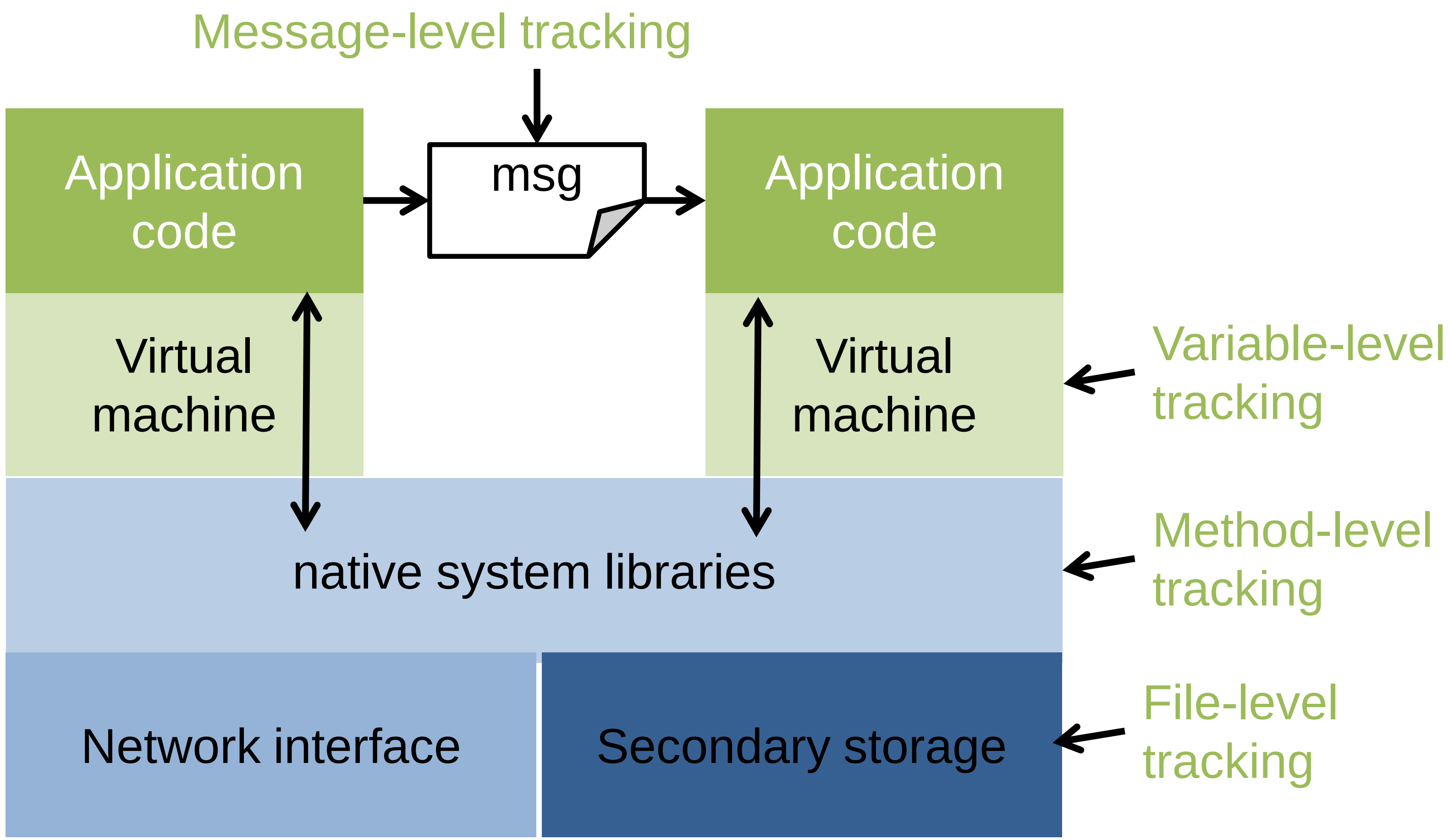
memory

taint map

code snippet from iexplore.exe

TaintDroid

Leverage Android Platform Virtualization



VM Variable-level Tracking

- We modified the Dalvik VM interpreter to store and propagate taint tags (a taint bitvector) on variables
  - Local variables and method args: taint tags stored adjacent to variables on the internal execution stack.
  - Class fields: similar to locals, but inside static field heap objects
  - Arrays: one taint tag per array to minimize overhead

DEX Taint Propagation Logic

Op Format	Op Semantics	Taint Propagation	Description
const-op vA C	vA ← C	T(vA) ← 0	Clear vA taint
move-op vA vB	vA ← vB	T(vA) ← T(vB)	Set vA taint to vB taint
move-op-R vA	vA ← R	T(vA) ← T(R)	Set vA taint to return taint
return-op vA	R ← vA	T(R) ← T(vA)	Set return taint (0 if void)
move-op-E vA	vA ← E	T(vA) ← T(E)	Set vA taint to exception taint
throw-op vA	E ← vA	T(E) ← T(vA)	Set exception taint
unary-op vA vB	vA ← op vB	T(vA) ← T(vB)	Set vA taint to vB taint
binary-op vA vB vC	vA ← vB op vC	T(vA) ← T(vB)UT(vC)	Set vA taint to vB taint U vC taint
binary-op vA vB	vA ← vA op vB	T(vA) ← T(vA)UT(vB)	Set vA taint to vA taint U vB taint
binary-op vA vB C	vA ← vB op C	T(vA) ← T(vB)	Set vA taint to vB taint
aput-op vA vB vC	vB[vC] ← vA	T(vB[]) ← T(vB[]) UT(vA)	Update array vB taint with vA taint
…			

Native Methods

- Applications execute native methods through the Java Native Interface (JNI)

- TaintDroid uses a combination of heuristics and method profiles to patch VM tracking state

IPC and File Taint Propagation

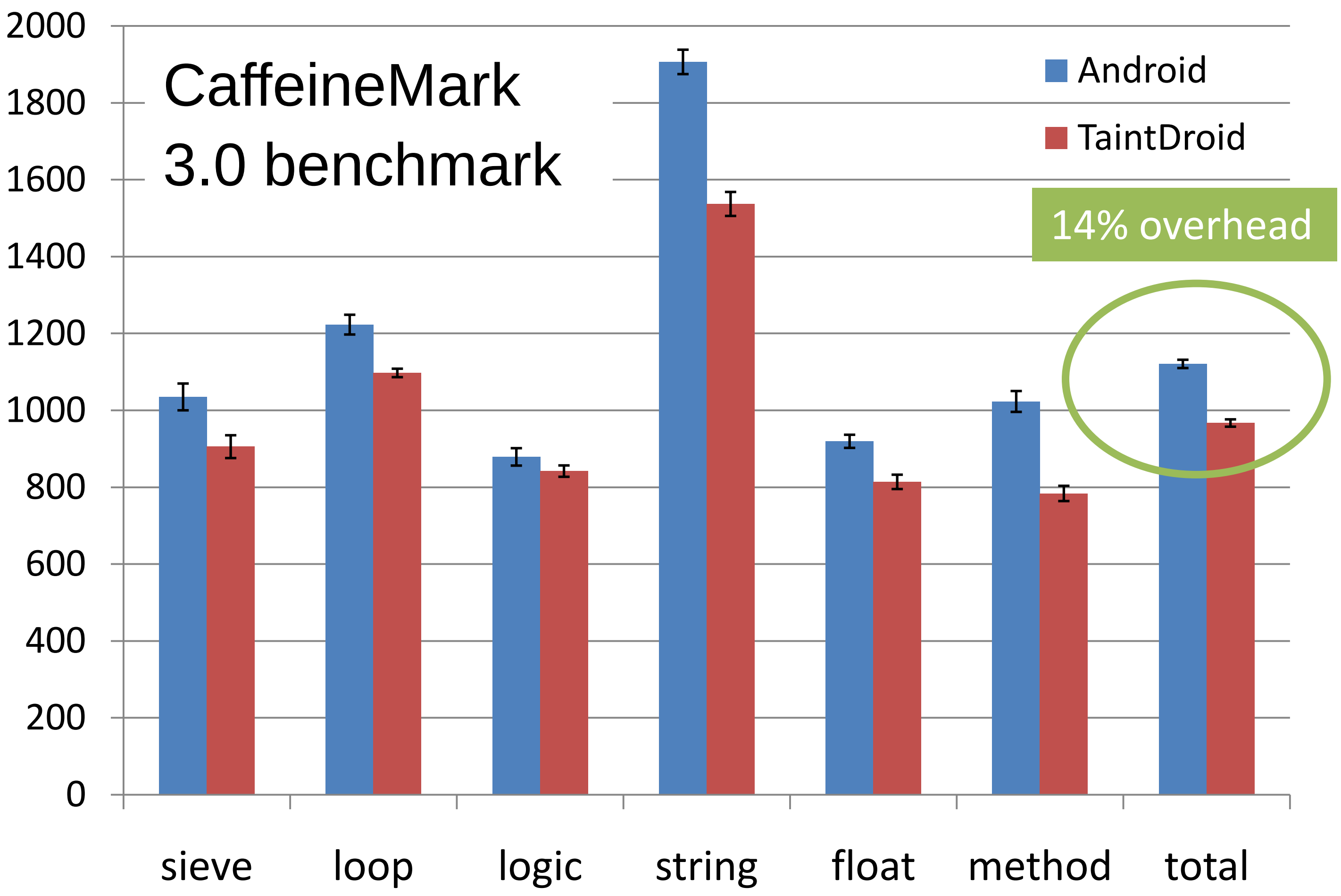
- Message-level tracking for IPC
  - Marshall data items
  - Unmarshall data items

- Persistent storage tracked at the file level
  - Single taint tag stored in the file system XATTR

Roadmap

- Motivation
- Our approach
- TaintDroid design
- Performance study
- Application study
- Other research work

Performance Study: Microbenchmark



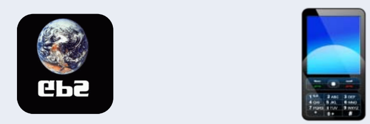
Performance Study

- Memory overhead: 4.4%
- IPC overhead: 27%
- Macro-benchmark
  - App load: 3% (2ms)
  - Address book: (<20ms) 5.5% create, 18% read
  - Phone call: 10% (10ms)
  - Take picture: 29% (0.5s)

Taint Adaptors

- Taint sources and sinks must be carefully integrated into the existing architectural framework.
- Sources
  - Low-bandwidth sensors: location, accelerometer
  - High-bandwidth sensors: microphone, camera
  - Information databases: address book, SMS storage
  - Device identifiers: IMEI, IMSI, ICC-ID, Phone #
- Sink: network

Application Study

Applications (with the Internet permission)	#	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, Barcode Scanner, Coupons, Trapster, Spongebot Slide, ProBasketBall	7	
MySpace, ixMAT	2	
Evernote	1	

Findings: Location

- 15 of the 30 apps shared physical location with an ad server (admob.com, ad.qwapi.com, ads.mobclix.com, data.flurry.com)

```
e.g., received data with tag 0x411 data=[GET
  /servernameA1?hello=1&time=1&bumpid=354957
  030504982&locale=en_US&gpslong=-
  122.316&gpslat=47.662&gpsaccuracy=32.000&t
  imezone=0…
```

- In no case was sharing obvious to user or in EULA
  - In some cases, periodic and occurred without app use

Findings: Phone Identifiers

- 7 apps sent device (IMEI) and 2 apps sent phone #, IMSI, ICC-ID to remote servers without informing the user
- Frequency was app-specific, e.g., one app sent phone information every time the phone booted

Demo

What We've Learned

- Efficient, system-wide, dynamic taint tracking for mobile platforms.
  - 14% overhead for computing-intensive work
- Private data leak is prevalent
  - 20 of the 30 studied applications share information in a way that was not expected

On-going Work

- AppInspector: automated privacy testing of smartphone applications

- AppShield: exploring runtime context for flexible and useful control of personal data exposure, UI issues

THANK YOU!
Q & A