SSS12 - HW3: TaintDroid

Alexander Georgii-Hemming Cyon
Andreas Cederholm
Mathias Pedersen
Magnus Bergman
Mattias Uskali
Carl Björkman

Outline

- What is TaintDroid?
- Why TaintDroid?
- Design challenges
- Design of TaintDroid
- Benchmarks and results
- Limitations

Important note

The authors of the paper are the creators of TaintDroid

What is TaintDroid?

- TaintDroid is a software developed for Android with the purpose of analyzing Android applications with aspect to information flow (IF)
- TaintDroid is an example of a dynamic analysis system of IF.
- TaintDroid is developed by various academic persons in cooperation with Intel Labs.
- The source code of TaintDroid is available at: www.appanalysis.org
- TaintDroid modifies the Android OS

Why TaintDroid?

- Applications on Android Market not verified by google(which is the case in AppStore)
- Developers can only request coarse-grained permissions
- Users rarely reads or understands the meaning of the permissions

How IF can be applied in mobile OS

- It is possible to develop applications which exposes sensitive user information to third parties.

- It is not only possible, there are a lot of apps which does so.

- IF analysis helps with detecting those confidentially compromising apps.

Design challenges

- Smartphones are resource constrained. Introducing CPU/RAM overhead is much noticeable on those devices.
- Permission system is too coarse-grained, which gives third party apps access to a lot of sensitive user data.
- Difficult to identify the sensitive data
- Information can be leaked to other apps

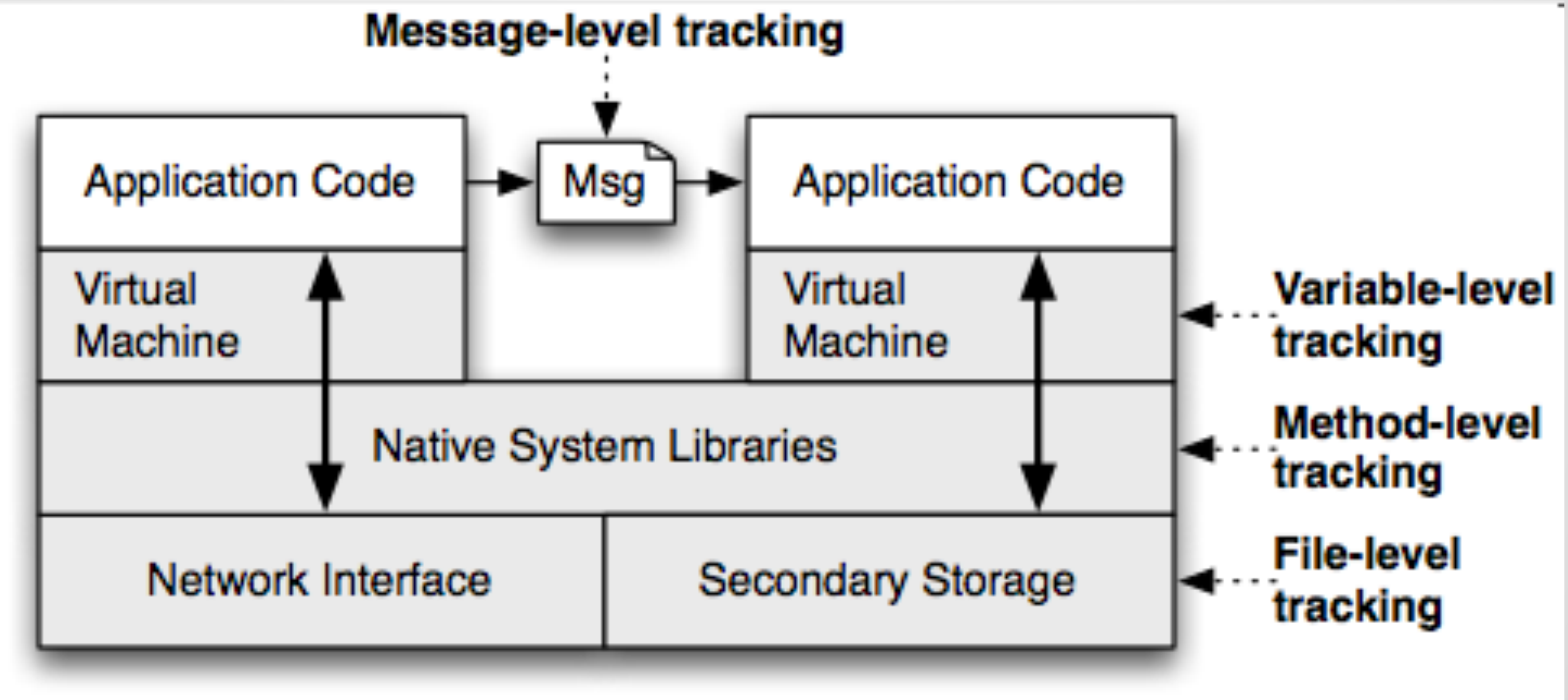
TaintDroid taint sources

- GPS
- Files on SD-card
- Contacts
- Accelerometer
- Microphone
- Camera
- SMS
- Sim card data
- IMEI Number

TaintDroid taint sinks

- WiFi
- 3G
- Bluetooth
- SMS
- NFC

Level trackings



Flow of taints within TaintDroid

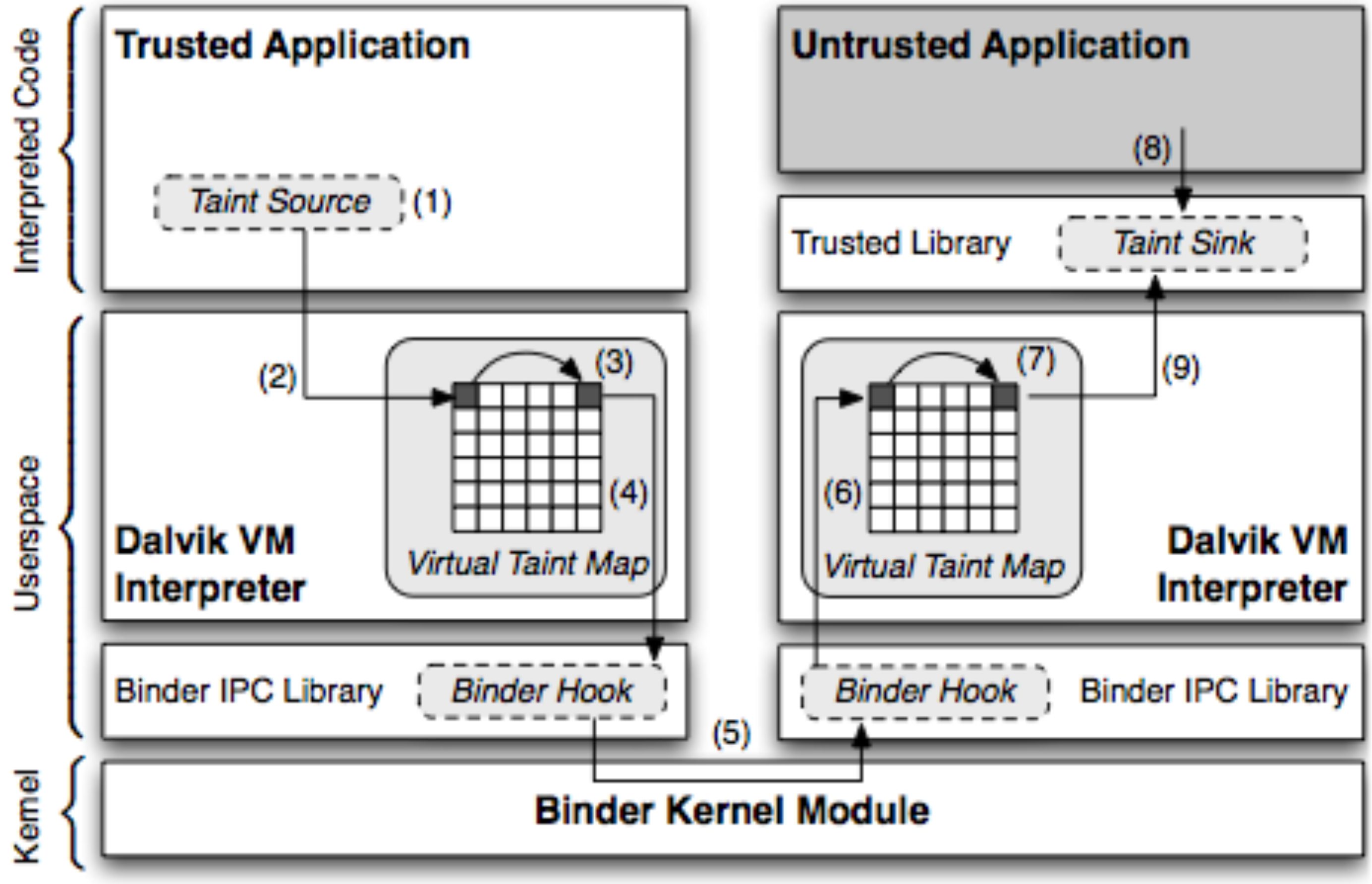
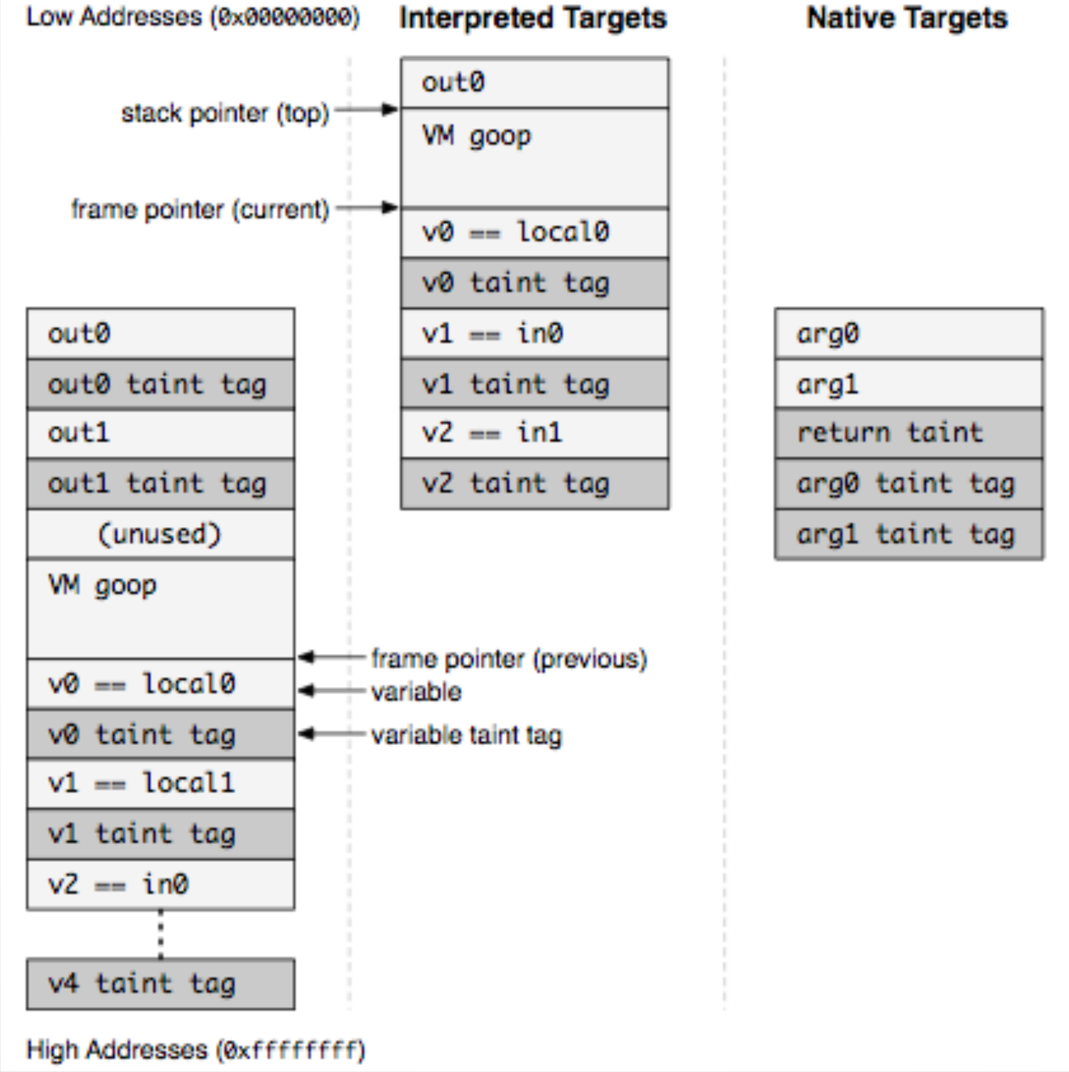


Figure 2: TaintDroid architecture within Android.

Flow of taints within TaintDroid ct'd

- What Taintdroid does is
- Every data read from a tainted source wich and store it in a variable than that variable will be tainted.
- If that variable then is copied that variable will also be marked as tainted.
- The taint tags are stored next to the variable in the memory in order to get good memory locality

Flow of taints within TaintDroid ct'd



Flow of taints within TaintDroid ct'd

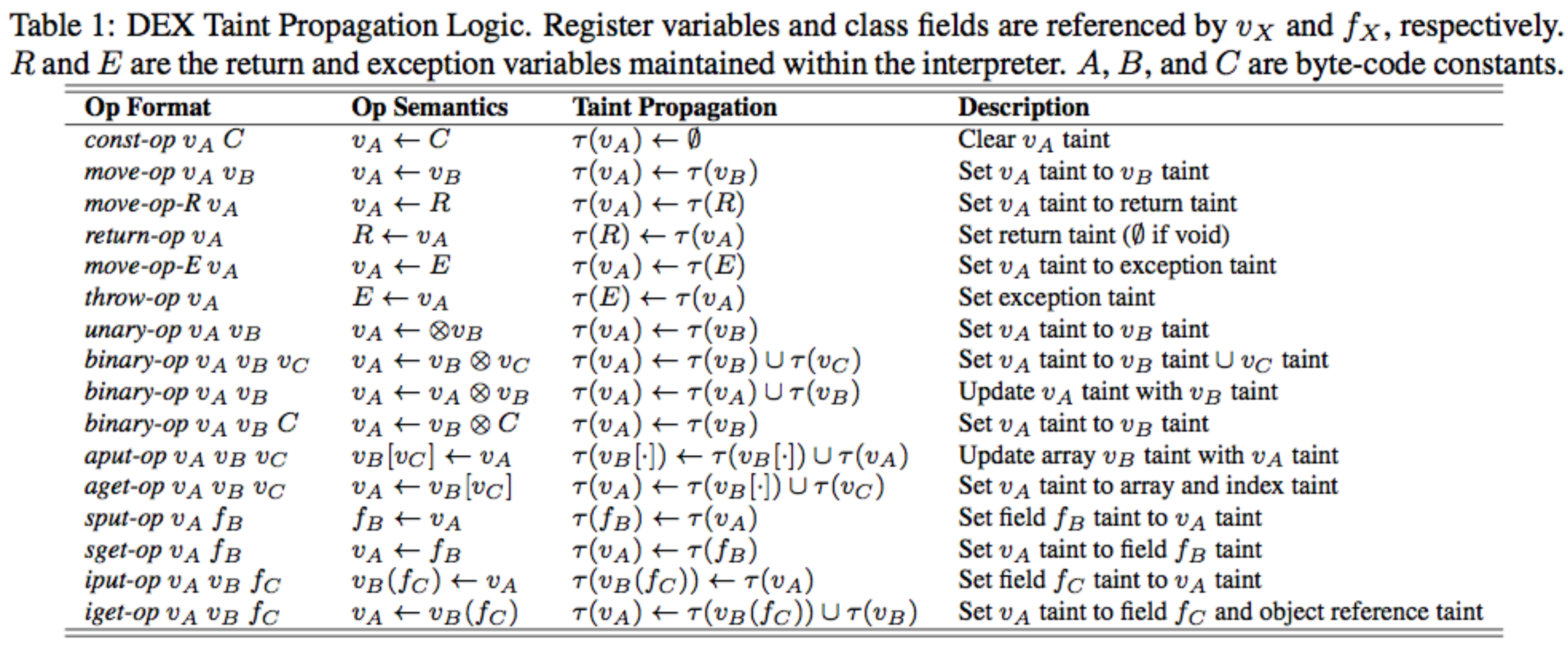


Table 1: DEX Taint Propagation Logic. Register variables and class fields are referenced by v_X and f_X, respectively. R and E are the return and exception variables maintained within the interpreter. A, B, and C are byte-code constants.

Op Format	Op Semantics	Taint Propagation	Description
const-op v_A C	$v_A \leftarrow C$	$\tau(v_A) \leftarrow \emptyset$	Clear v_A taint
move-op v_A v_B	$v_A \leftarrow v_B$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
move-op-R v_A	$v_A \leftarrow R$	$\tau(v_A) \leftarrow \tau(R)$	Set v_A taint to return taint
return-op v_A	$R \leftarrow v_A$	$\tau(R) \leftarrow \tau(v_A)$	Set return taint ($\emptyset$ if void)
move-op-E v_A	$v_A \leftarrow E$	$\tau(v_A) \leftarrow \tau(E)$	Set v_A taint to exception taint
throw-op v_A	$E \leftarrow v_A$	$\tau(E) \leftarrow \tau(v_A)$	Set exception taint
unary-op 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
binary-op 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
binary-op 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
binary-op 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
aput-op 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
aget-op 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
sput-op 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
sget-op 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
iput-op 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
iget-op 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

Message-level tracking

- Communication between applications
- IPC uses parcels

Method-level tracking

- Used for system-provided native libraries

File-level tracking

- Ensures persistent information conservatively retains its taint markings

Benchmarks

When benchmarking security they found out that out of 105 flagged instances, 37 of them turned out to be well-founded flags.

Benchmarks

When it comes to speed there are two ways of measuring: "macroscopic" and "microscopic" speed benchmarking.

Macroscopic: High-level functionality. "How long does it take to read a post in the contact list?"

Microscopic: Automatable analysis of delays in low-level calls.

Benchmarks

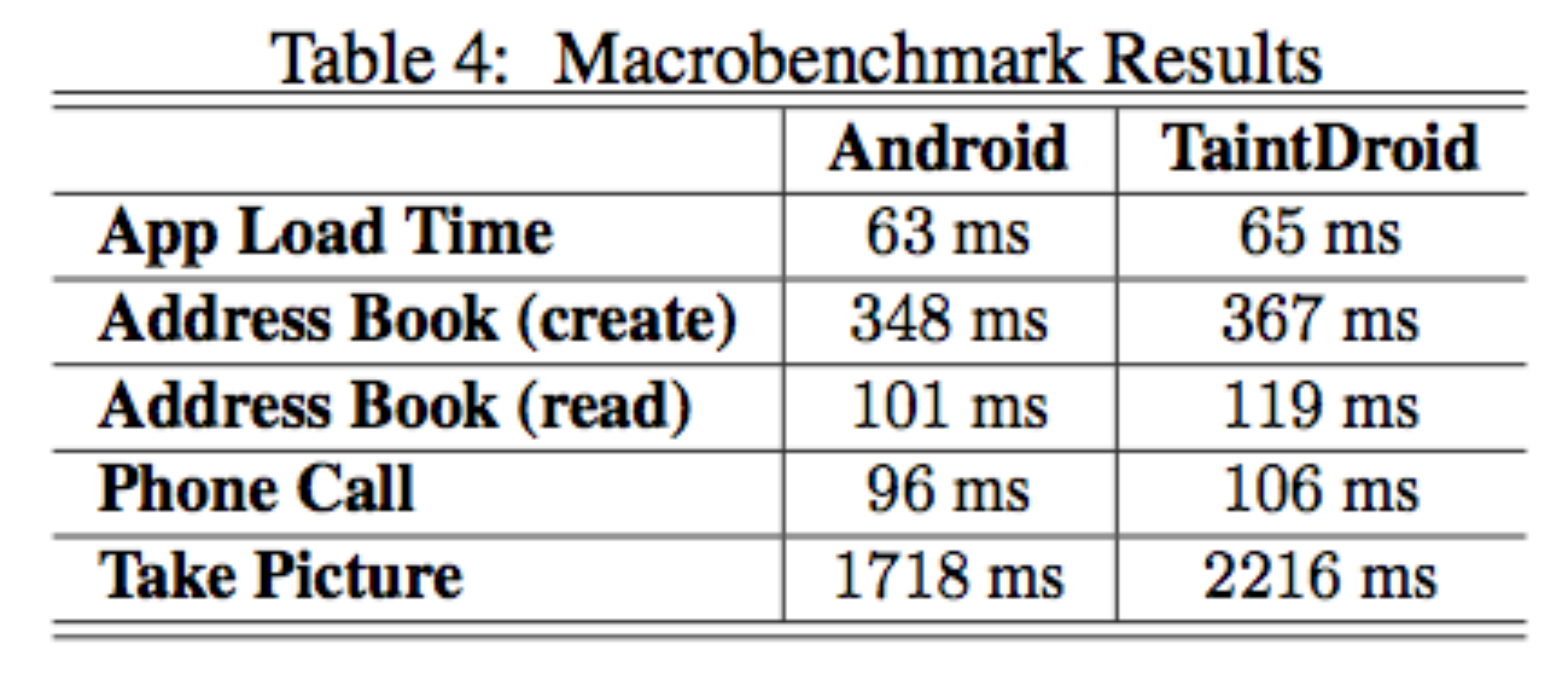


Table 4: Macrobenchmark Results

	Android	TaintDroid
App Load Time	63 ms	65 ms
Address Book (create)	348 ms	367 ms
Address Book (read)	101 ms	119 ms
Phone Call	96 ms	106 ms
Take Picture	1718 ms	2216 ms

Benchmarks

Speed overhead in macroscopic analysis:
App load time: 3%
Address Book (create): 5%
Address Book (read): 18%
Phone Call: 10%
Take Picture: 29%

Benchmarks

Speed overhead in microscopic analysis:

Java Microbench (CaffeineMark): 14% increase in score (more = bad)

Benchmarks

Memory overhead in IPC throughput:

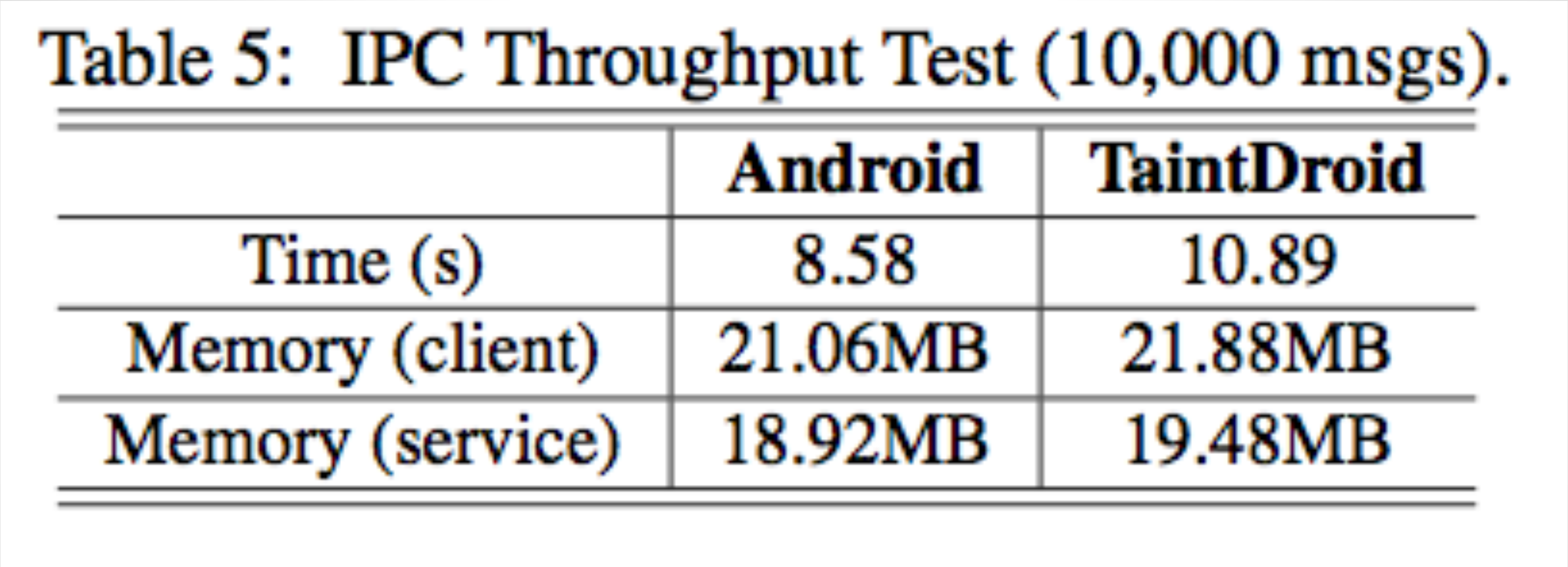


Table 5: IPC Throughput Test (10,000 msgs).

	Android	**TaintDroid**
Time (s)	8.58	10.89
Memory (client)	21.06MB	21.88MB
Memory (service)	18.92MB	19.48MB

Benchmarks

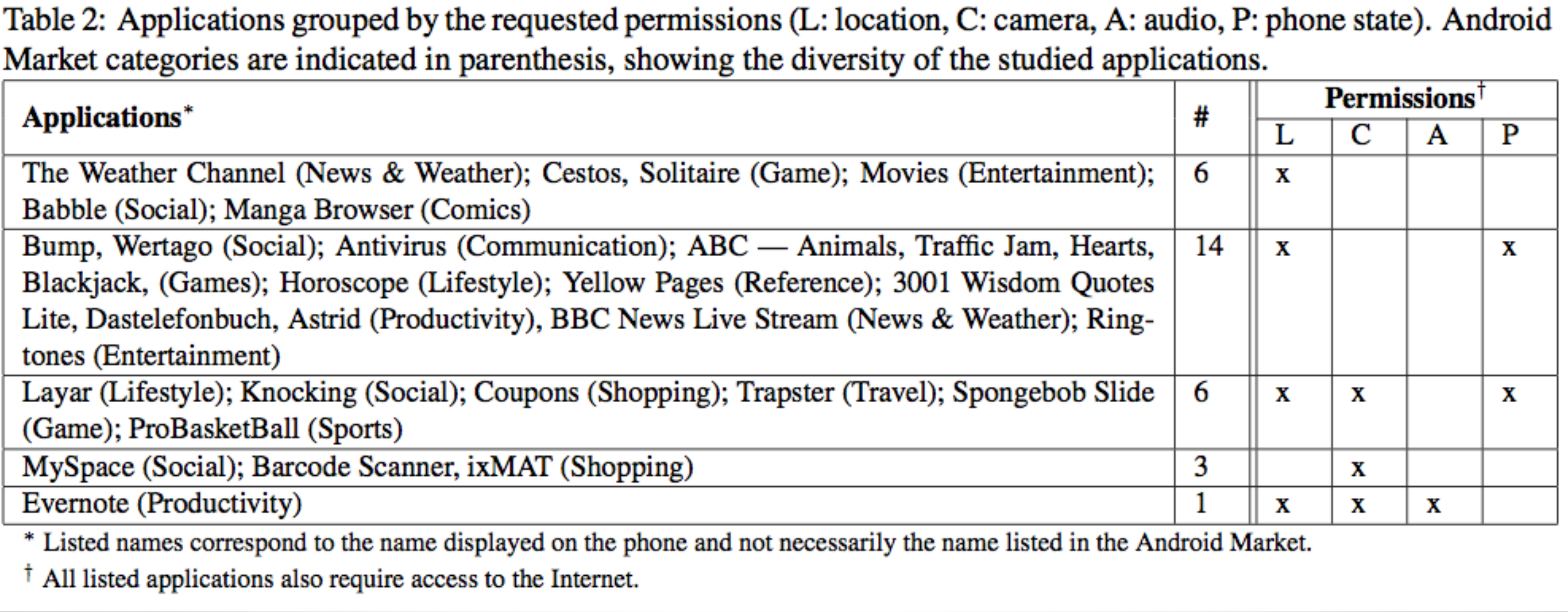


Table 2: Applications grouped by the requested permissions (L: location, C: camera, A: audio, P: phone state). Android Market categories are indicated in parenthesis, showing the diversity of the studied applications.

Applications*	#	Permissions†			
		L	C	A	P
The Weather Channel (News & Weather); Cestos, Solitaire (Game); Movies (Entertainment); Babble (Social); Manga Browser (Comics)	6	x			
Bump, Wertago (Social); Antivirus (Communication); ABC — Animals, Traffic Jam, Hearts, Blackjack, (Games); Horoscope (Lifestyle); Yellow Pages (Reference); 3001 Wisdom Quotes Lite, Dastelefonbuch, Astrid (Productivity), BBC News Live Stream (News & Weather); Ringtones (Entertainment)	14	x			x
Layar (Lifestyle); Knocking (Social); Coupons (Shopping); Trapster (Travel); Spongebob Slide (Game); ProBasketBall (Sports)	6	x	x		x
MySpace (Social); Barcode Scanner, ixMAT (Shopping)	3		x		
Evernote (Productivity)	1	x	x	x	

* Listed names correspond to the name displayed on the phone and not necessarily the name listed in the Android Market.

† All listed applications also require access to the Internet.

Benchmarks

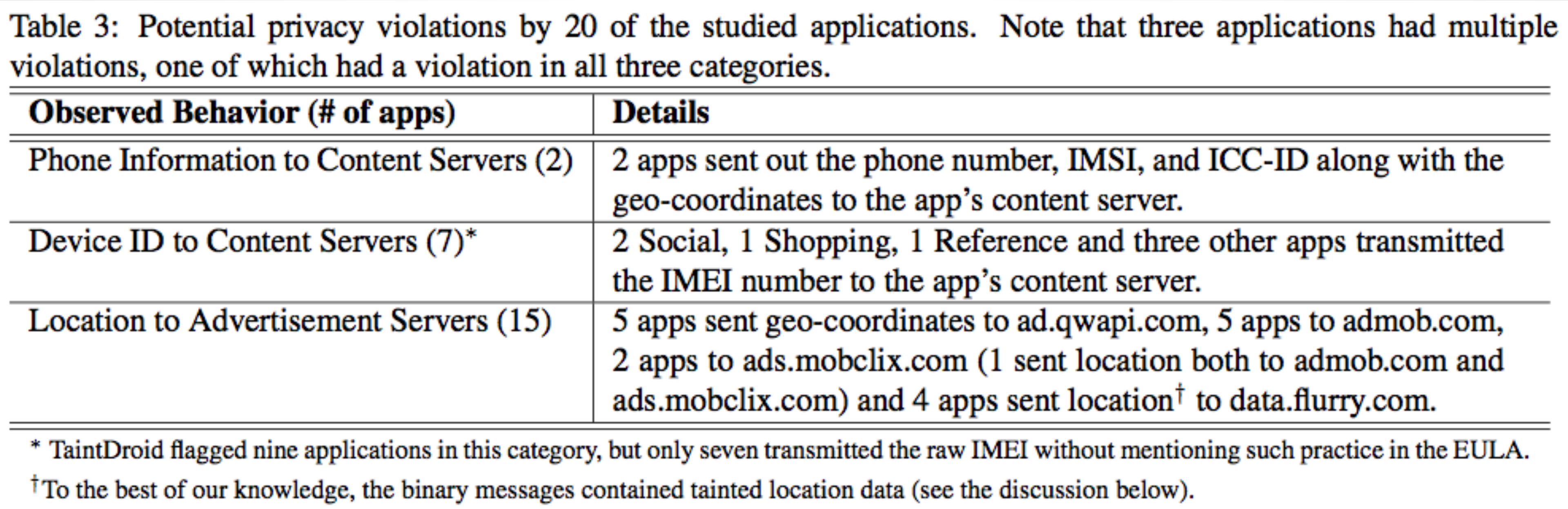


Table 3: Potential privacy violations by 20 of the studied applications. Note that three applications had multiple violations, one of which had a violation in all three categories.

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.

* TaintDroid flagged nine applications in this category, but only seven transmitted the raw IMEI without mentioning such practice in the EULA.

† To the best of our knowledge, the binary messages contained tainted location data (see the discussion below).

TaintDroid limitations

- TaintDroid is incapable of detecting implicit IF

- Only dynamic analysis, not static.

- A lot of false positives

- Only detecting, **not preventing,** leak of sensitive user information

- Requires Android 2.1

- Modifies the Android OS