TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones

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

Presentation by Krzysztof Pawlowski

Warsaw, 02.01.2012

Agenda

- What is TaintDroid?
- Approach Overview
- Background: Android
- TaintDroid Implementation
- Privacy Hook Placement
- Application Study
- Performance Evaluation
- Conclusion

What is TaintDroid?

- Access rights for app set while installing
- No way to track how the data is used by the application

PRIVACY-SENSITIVE SOURCES:

- GPS, accelerometer
- Camera, microphone
- Phone number, IMEI, SIM card number

Approach Overview

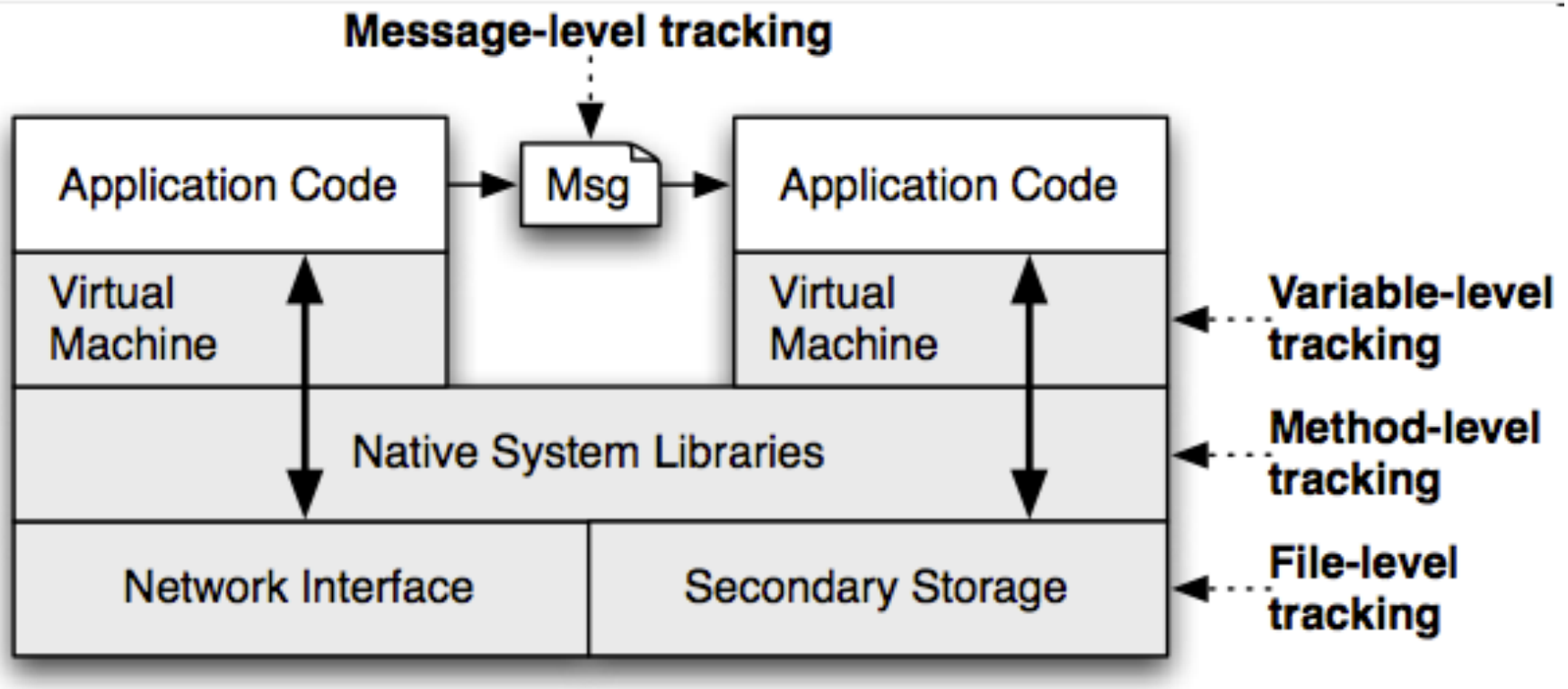
CHALLENGES:

- Static source code analysis infeasible
- Resource constraints on Smartphones
- Several types of privacy sensitive data
- Dynamic data
- Sharing information between apps

Approach Overview

- Dynamic taint analysis
- Taint source
- Taint marking indicating the information type
- Taint propagation
- Instruction level taint analysis -> complexity, taint explosion

Approach Overview



Approach Overview

- Assumption: native code is trusted
- Only 5% of apps using own native-code libraries (2010)
- Modified native library loader -> only native libraries from firmware can be loaded

Background: Android

- Dalvik VM Interpreter
- Native Methods
- Binder IPC

TaintDroid Implementation

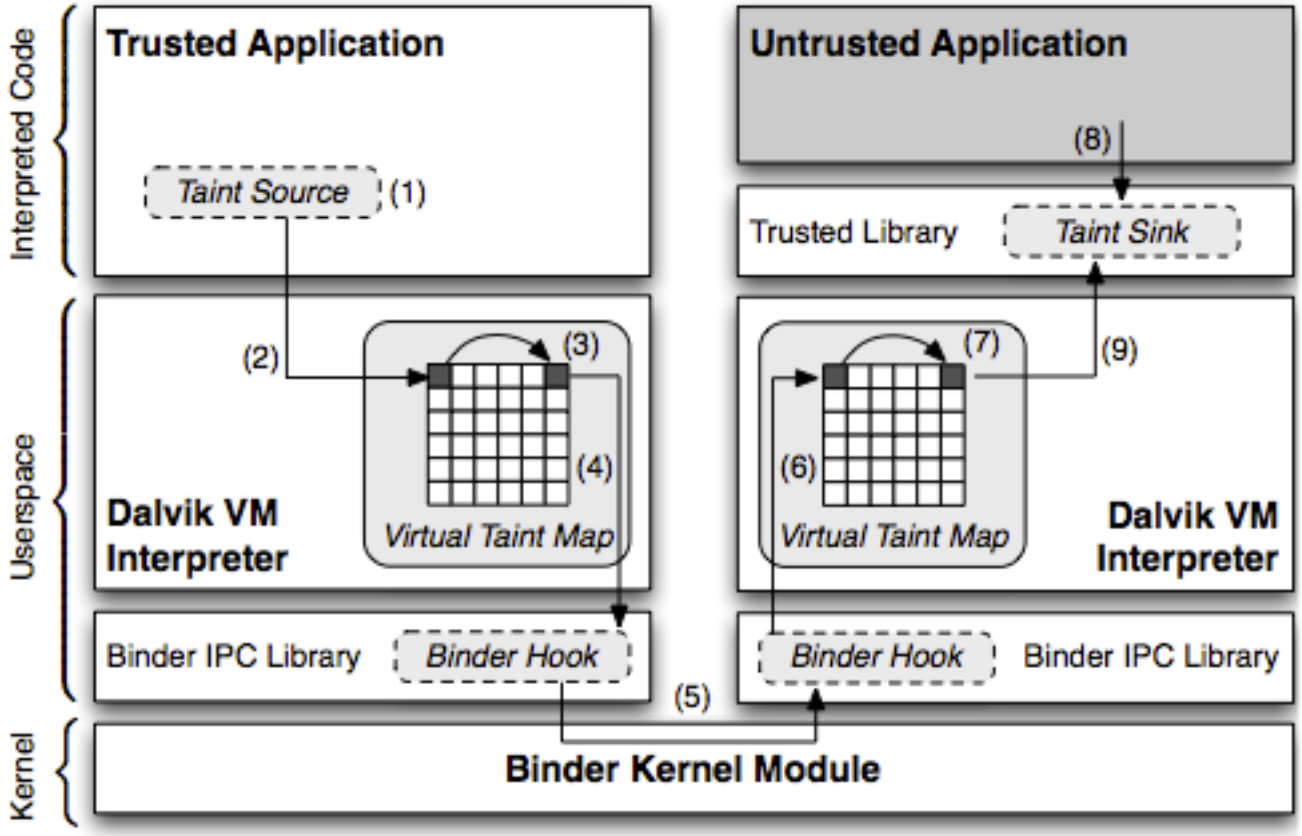


Figure 2: TaintDroid architecture within Android.

TaintDroid Implementation

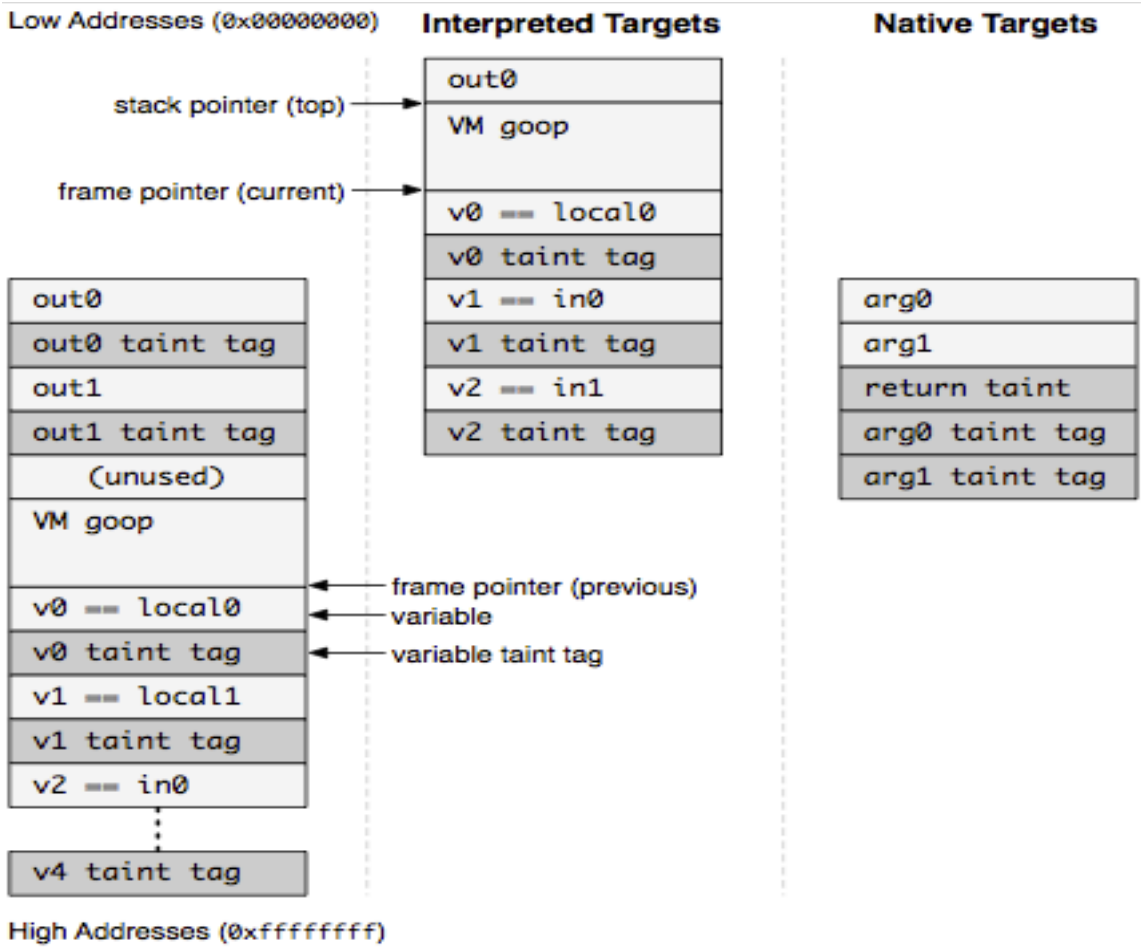
ARCHITECTURE IMPLEMENTATION CHALLENGES:

- Taint Tag Storage
- Interpreted Code Taint Propagation
- Native Code Taint Propagation
- IPC Taint Propagation
- Secondary Storage Taint Propagation

Taint Tag Storage

- Tainted variables types: method local vars, method args, class static fields, class instance fields, arrays
- Method local vars and args kept on an internal stack
- Method invoked => new stack frame allocated
- Allocation taint storage by doubling frame size
  (32-bit register and 32-bit taint tag adjacent to each other)
- One tag per array / string (minimization of storage overhead), but leads to false positives

Taint Tag Storage



Interpreted Code Taint Propagation (Dalvik VM)

DATA FLOW LOGIC:

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

Interpreted Code Taint Propagation (Dalvik VM)

- Data flow logic is straightforward except for aget-op for array and iget-op for class' field

Explanation for aget-op (array index taint):

- Translation table from lowercase to uppercase chars
- If tained val 'a' is used as an array index the resulting 'A' should be tainted even though 'A' value in the array is not

Interpreted Code Taint Propagation (Dalvik VM)

Explanation for iget-op (tainting object references):

```
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);
```

Native Code Taint Propagation

- Native code unmonitored in TaintDroid
- Stack frame augmented (access to java args' taint tags)
- Internal VM methods instrumented manually
- For JNI the JNI bridge is patched (union of method args taint tags is assigned to the result taint tag)
- (a propagation using source code in JNI is planned to be implemented)

IPC Taint Propagation

- Message-level propagation
- Variable-level propagation would be bad (encoding sequence of scalars as string)
- Leads to false positives
- Future plans: word-level taint tags along with additional consistency checks

Secondary Storage Taint Propagation

- Taint tag may be lost when data is written to file
- One taint tag per file => false positives
- Extended attribute support (YAFFS2)

Taint Interface Library

FUNCTIONS OF TAINT INTERFACE LIBRARY:

- Add taint markings to variables
- Retrieve taint markings from variables

- No possibility to set or clear

Privacy Hook Placement

LOW BANDWIDTH SENSORS

- E.g. location and accelerometer
- LocationManager and SensorManager

HIGH BANDWIDTH SENSORS

- E.g. microphone, camera
- OS shares this information via large data buffers, files or both

Privacy Hook Placement

INFORMATION DATABASES

- Data stored in files

DEVICE IDENTIFIERS

- Phone number, SIM card number, IMEI number
- Accessible by well-defined API in Android

Privacy Hook Placement

NETWORK TAINT SINK

- Checking if private-sensitive information is sent away
- VM interpreter-based solution => taint sink placed in Java at the point the native socket library is invoked

Application Study

EXPERIMENTAL SETUP

- From the set of 1100 apps (50 most popular from each category) 358 required Internet permission
- From this 358 apps set 30 apps were randomly selected (8.4% sample size)
- 22,594 packets (8.6 MB)
- 1,130 TCP connections

Application Study

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); Ring-tones (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.

Application Study

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.

Performance Evaluation

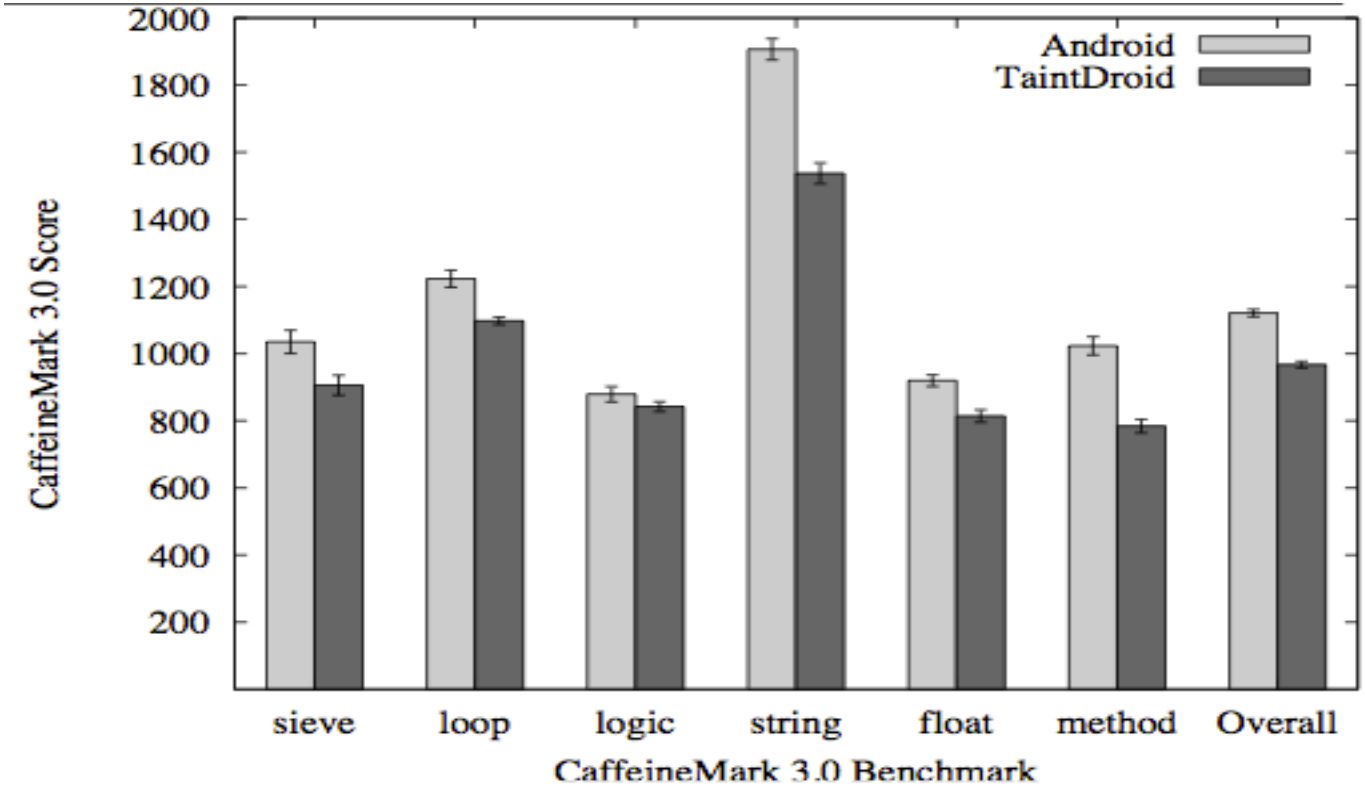
MACROBENCHMARK

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

Performance Evaluation

JAVA MICROBENCHMARK (Caffeine)



Performance Evaluation

IPC MICROBENCHMARK

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

Conclusions

- Tracks only data flows
- Do not track control flows
- 14% performance overhead
- 2/3 of the apps in the study exhibit suspicious handling of sensitive data
- ½ of the apps reported users' location to remote ads servers

THANK YOU!

Questions?