

Privacy on Smartphones

Presentation by Claude Barthels

Roadmap

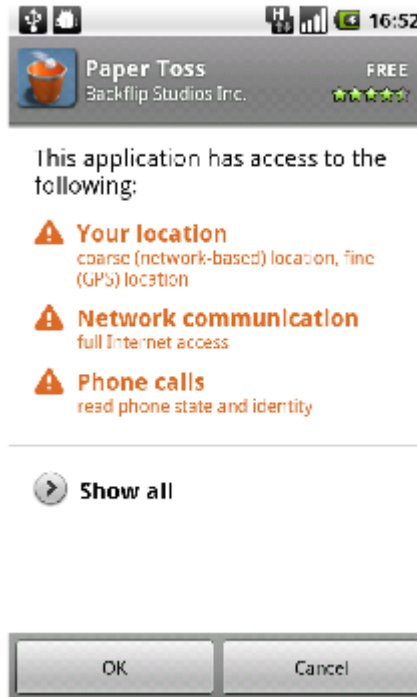
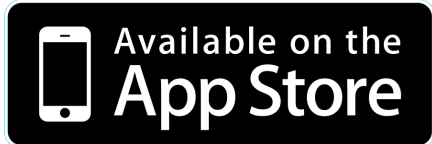
- TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones
- MockDroid: Trading Privacy for Application Functionality on Smartphones
- Paranoid Android: Versatile Protection for Smartphones

TaintDroid

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

Paper by W. Enck, P. Gilbert, B.-G. Chun,
L. P. Cox, J. Jung, P. McDaniel, A. N. Sheth

Problem Setting



What is TaintDroid?

Extension of the Android platform

Tracks flow of information through an application

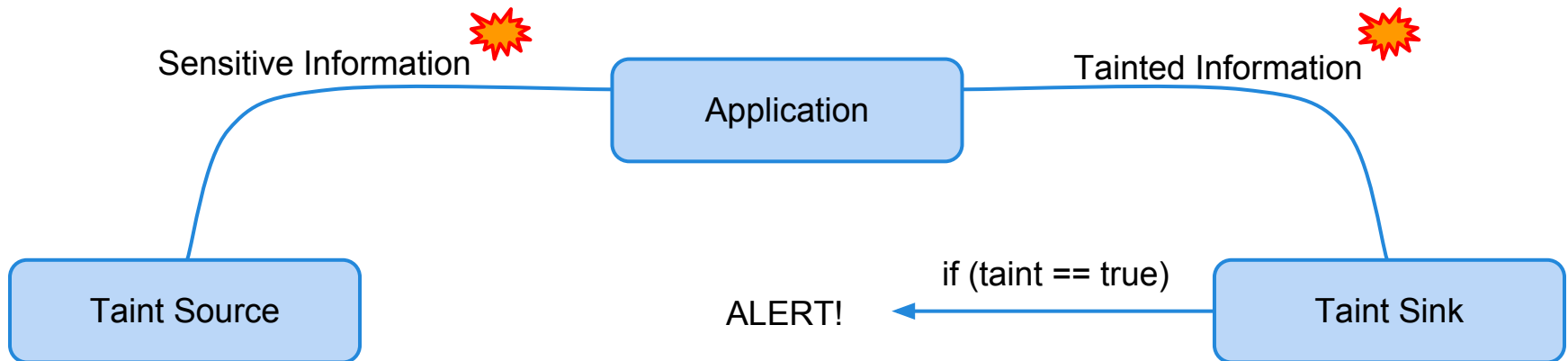
Realtime analysis & feedback

Tracks data between processes (file, IPC, ...)

General idea

Mark (taint) sensitive information

Taint sources and sinks



Design Challenges

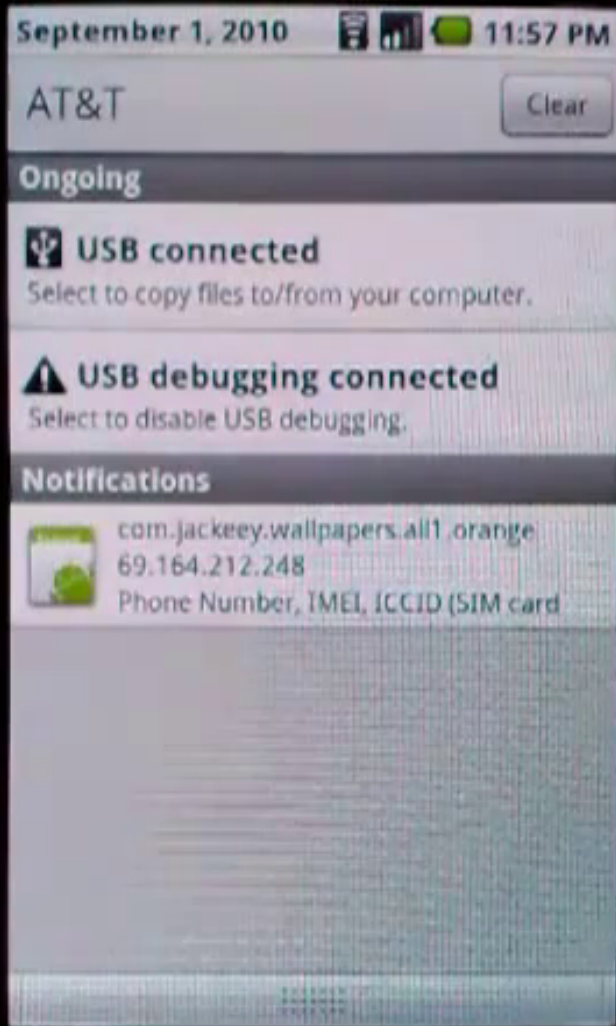
Limited resources & performance

Identifying private information

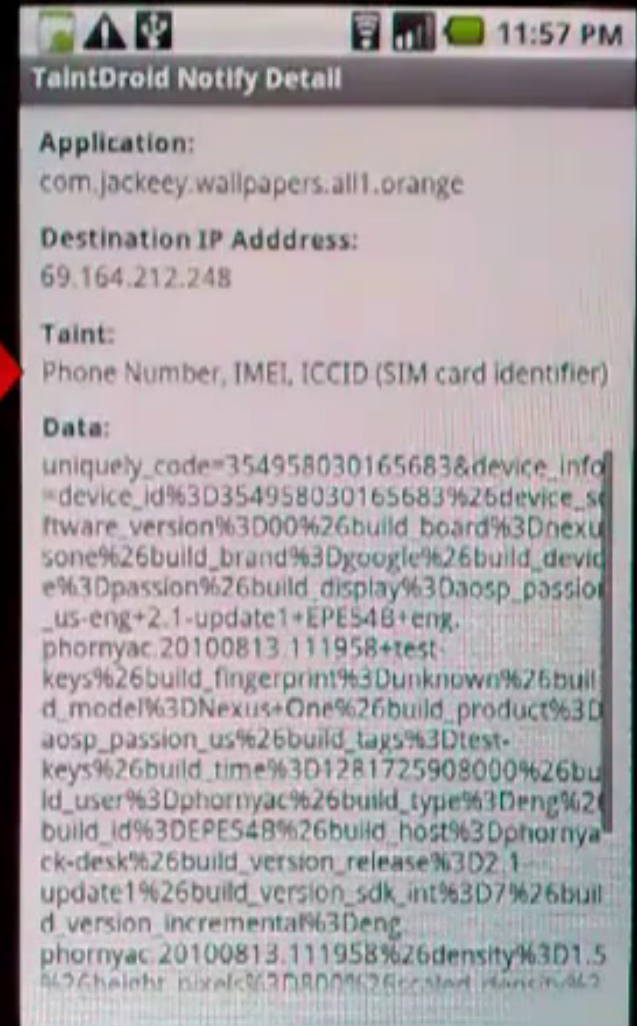
Multiple types and sources of sensitive data

Data sharing between applications

User notification



phone #, IMEI #,
SIM card ID
sent to *Imnet.us*



How it works - Variable level

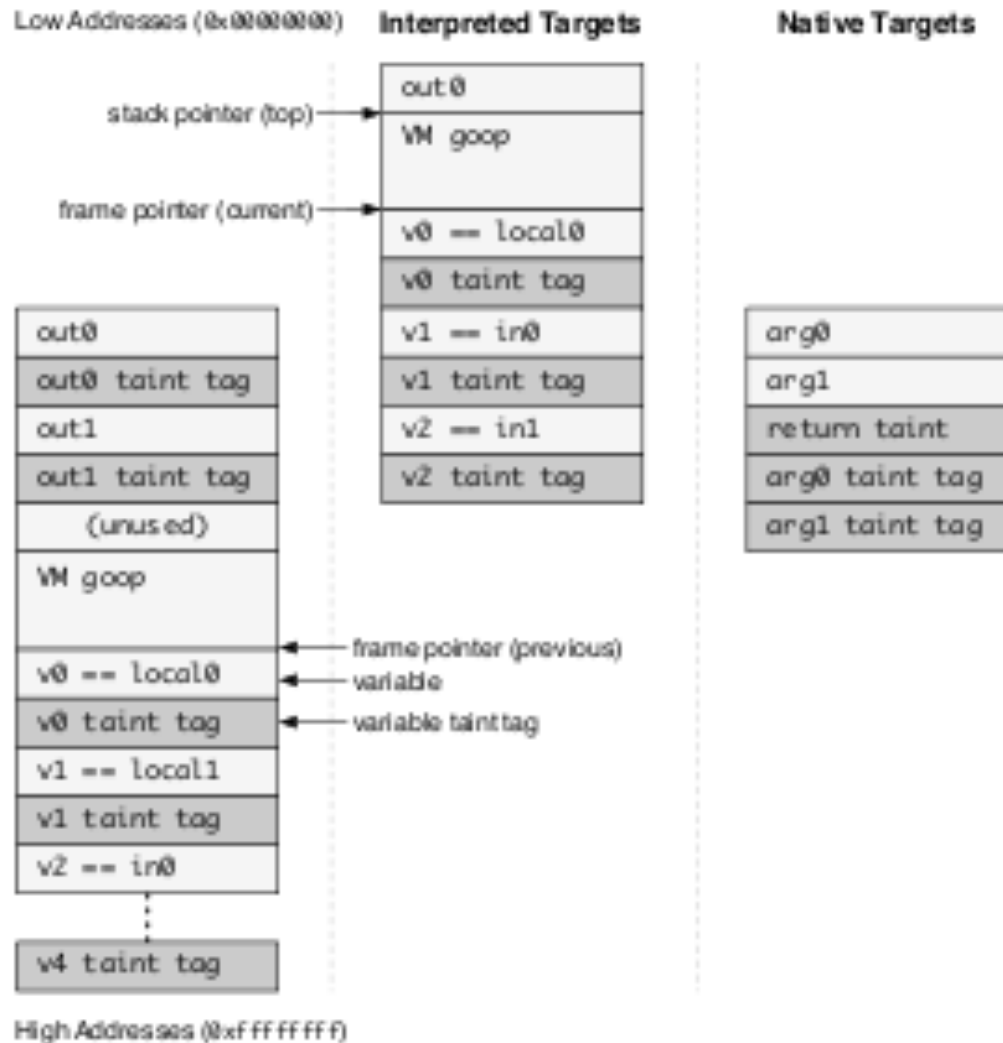
Multiple taint markings stored in a taint tag

Taint tag is a 32bit vector

Stored adjacent to the variables

Only one taint tag per array

How it works - Stack layout



How it works - Message & file level

Only one tag per message or file

Union over all taint tags of the variables contained in the message or file

Potential for false positives

Less overhead than a finer granularity

How it works - Propagation logic

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

Where to place taint sources & sinks?

Low-bandwidth sensors (location, accelerometer, ...)

High-bandwidth sensors (camera, microphone, ...)

Information databases (calendar, address book, ...)

Device identifiers (SIM number, IMEI number, ...)

Network Taint Sink

Limitations

Data flow tracking only / No control flow tracking

Native code is unmonitored

- Conservative heuristic: Assign union of argument taint markings to return type

Sometimes too coarse grained

- One taint tag per message or file
- One taint tag per array

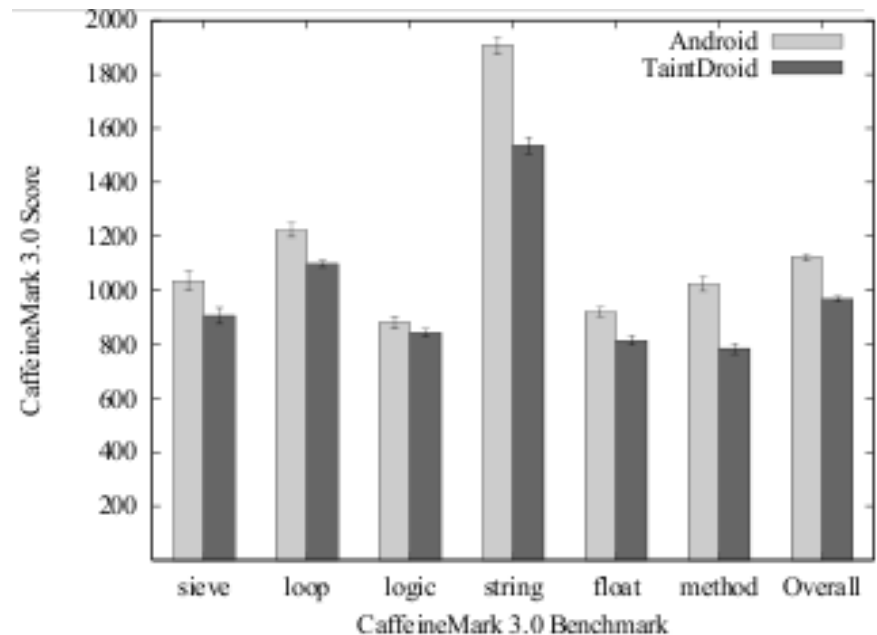
Performance

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

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



Experiment - Setup

30 popular applications

~ 100 minutes of recording

Network access + additional permissions

Nexus One with Android 2.1

Experiment - 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.

Experiment - Results

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).

Reviews

6 Reviews - Average Score 2.16 (accept)

- + Privacy is an issue (Data scandal is a matter of time)
- + Low overhead / Good performance - accuracy tradeoff
- +/- Study with open source software as ground truth
- +/- A lot of implementation details
- No native code tracking or static code analysis
- A lot of Android knowledge required
- Too sophisticated for 'normal' user
- May force developers to create new malicious ways to get the data
- Only notifications / No control

MockDroid

Trading Privacy for Application
Functionality on Smartphones

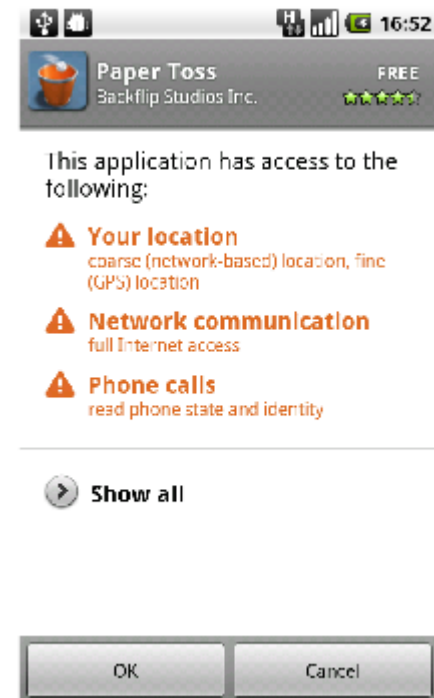
Paper by A. R. Beresford, A. Rice, N. Skehin,
R. Sohan

Problem setting

Similar problem setting as TaintDroid

Applications often require sensitive data to work correctly

Access to resources is granted once at install time and cannot be changed afterwards



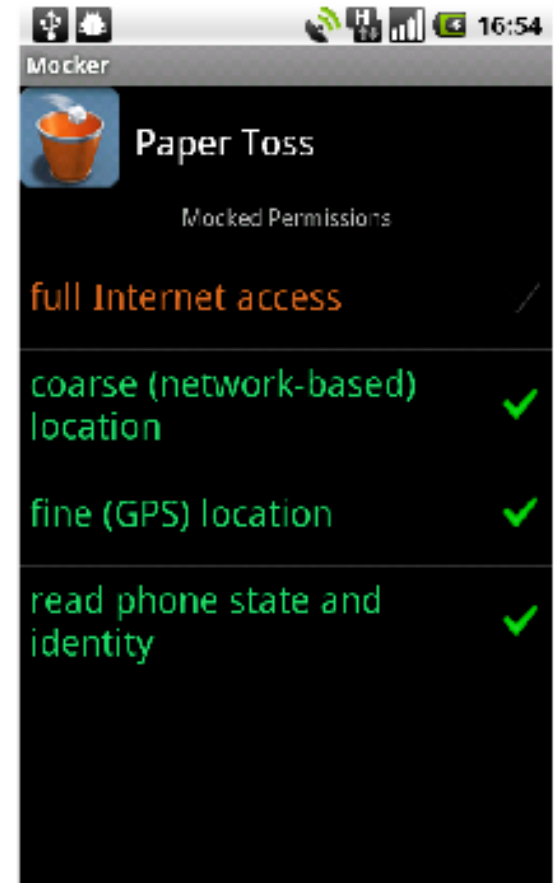
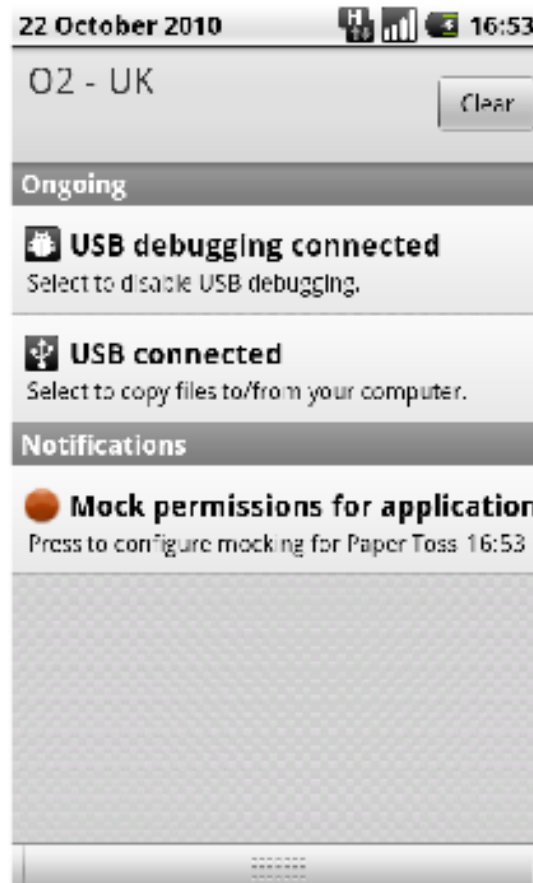
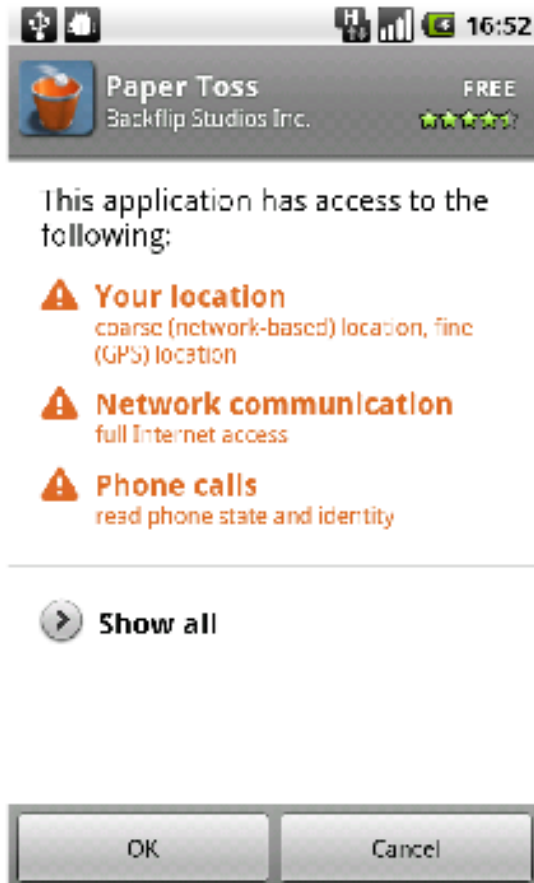
What is MockDroid?

Extension of the Android platform

MockDroid allows to fake (mock) sensitive data

Decision of faking data can be done/changed at runtime

What is MockDroid?



How it works

Granted permissions are stored by Android in an in-memory data structure and on disk

API calls check the in-memory data structure

MockDroid extends the data structure with a 'real' and a 'mocked' version of the permission

Internet permissions requires `inet` group. MockDroid therefore adds a `mocked_inet` group

What can be faked?

Location - no location fix

Internet - connection timeout

Calendar & contacts - empty database - zero rows affected

Device id - Fake constant value

Broadcast intents - Intents never sent/received

Limitations

Limited in what can be faked

- Instead of no location, just an approximate indication (e.g. next big city)
- Instead of empty contact or calendar database, MockDroid could return a subset (like public events)

Evaluation

Local

- location used for location based advertisements
- No reduced functionality

Internet:

- Limited functionality when mocking internet access
- Continue to run even without internet access

Internet	Local
3001 Wisdom Quotes	ABC – Animals
BBC News Live Stream	Antivirus
Coupons	Astrid
Dastelefonbuch	Blackjack
Horoscope	Bump
Layar	Cestos
Manga Browser	Evernote
Movies	Probasketball
Ringtones	Solitaire
The Weather Channel	Traffic Jam
Yellow Pages	Trapster
Wertago	

Paranoid Android

Versatile Protection for Smartphones

Paper by G. Portokalidis, P. Homburg,
K. Anagostakis, H. Bos

Problem setting

Smartphones hold privacy sensitive information

Become highly valuable targets for attacks

Security solutions from PCs are not always applicable to smartphones

What is Paranoid Android?

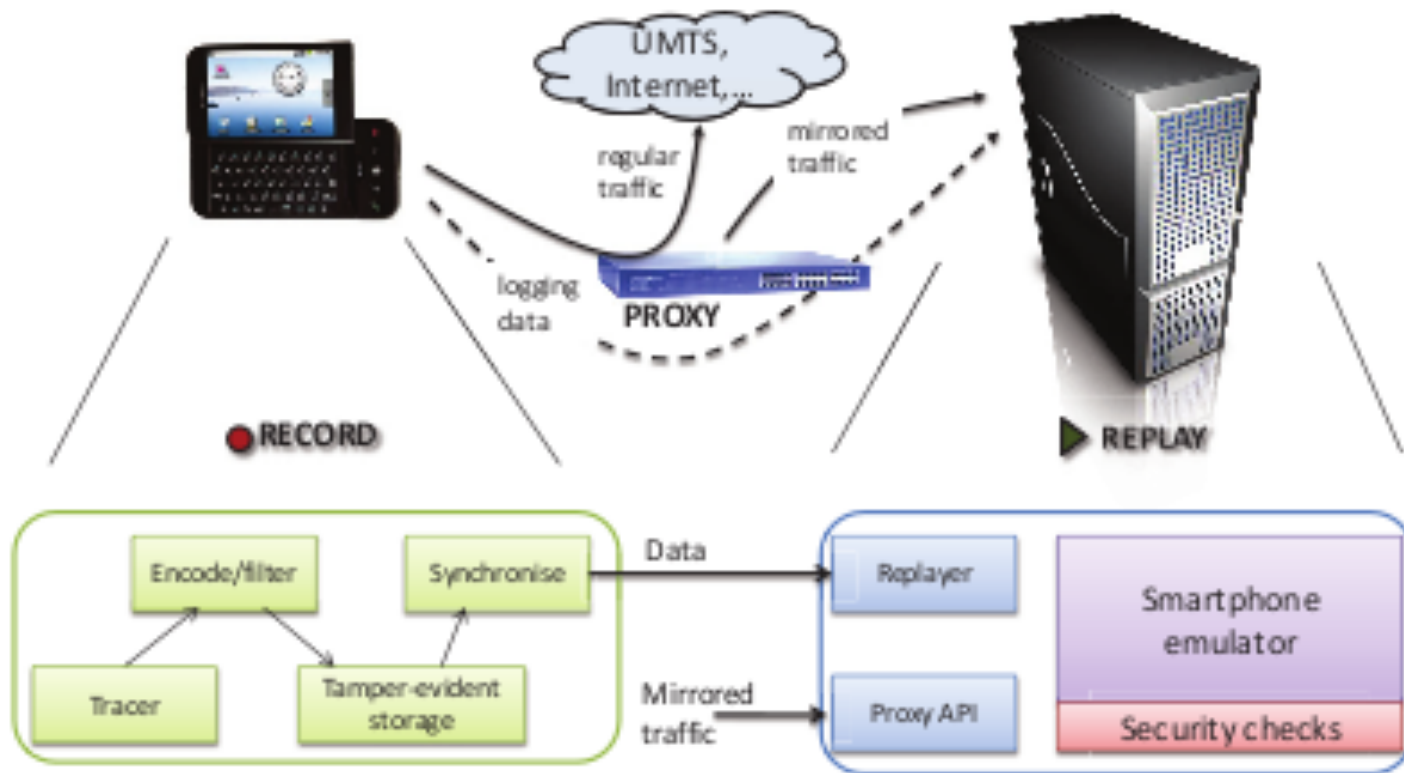
Security as a service

Security checks are performed by security servers

Security servers hold an exact replica of the phone in a virtual environment

Record & replay model

Overall architecture



Security Model

Buffer overflows & Code injection
(implemented in prototype)

Open source AntiVirus scanner (for file scans)
(implemented in prototype)

Memory scanner for patterns of malicious code

Abnormal system call detection

... flexible model which can be extended

Notification & Recovery

Notifications, Emails or SMS may be blocked

Hardware support

Restore to clean state using the replica

Minimizing data loss

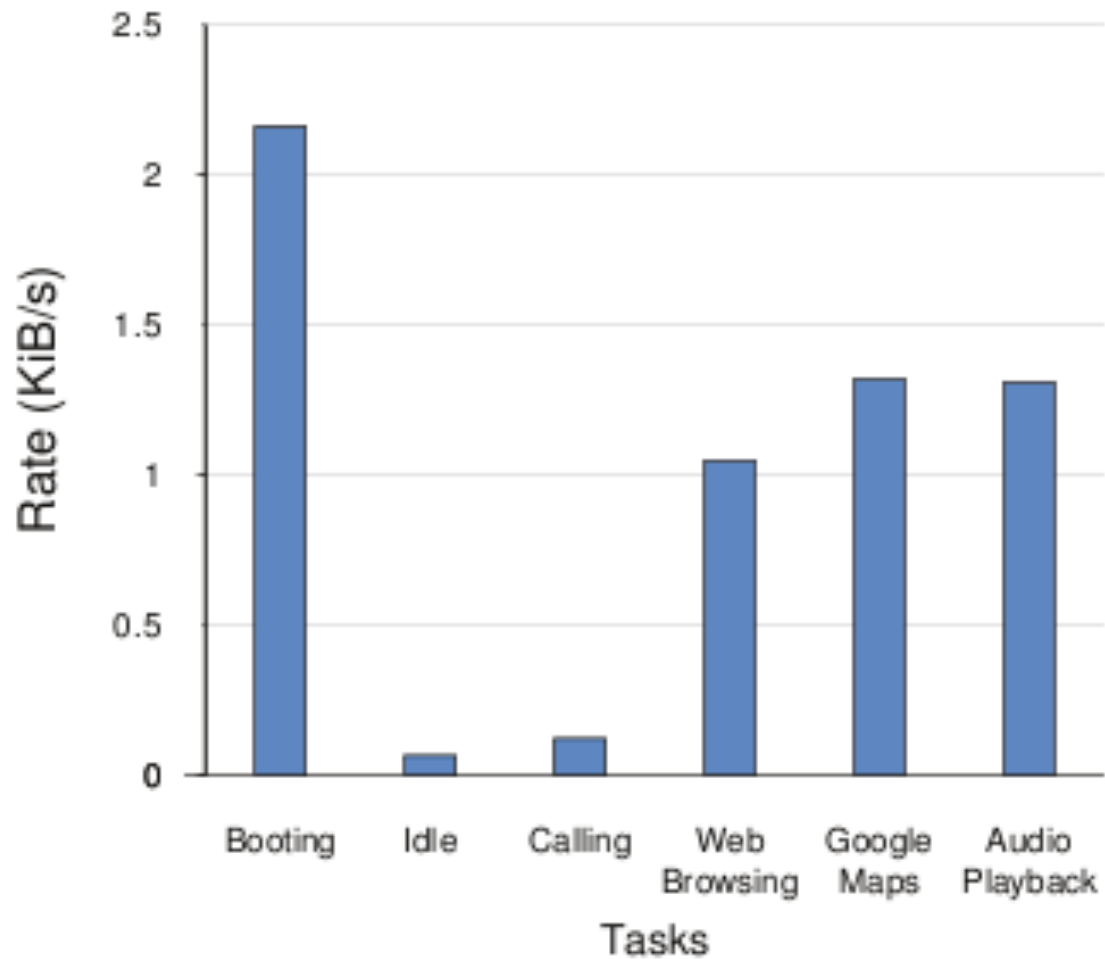
Evaluation

Amount of trace data

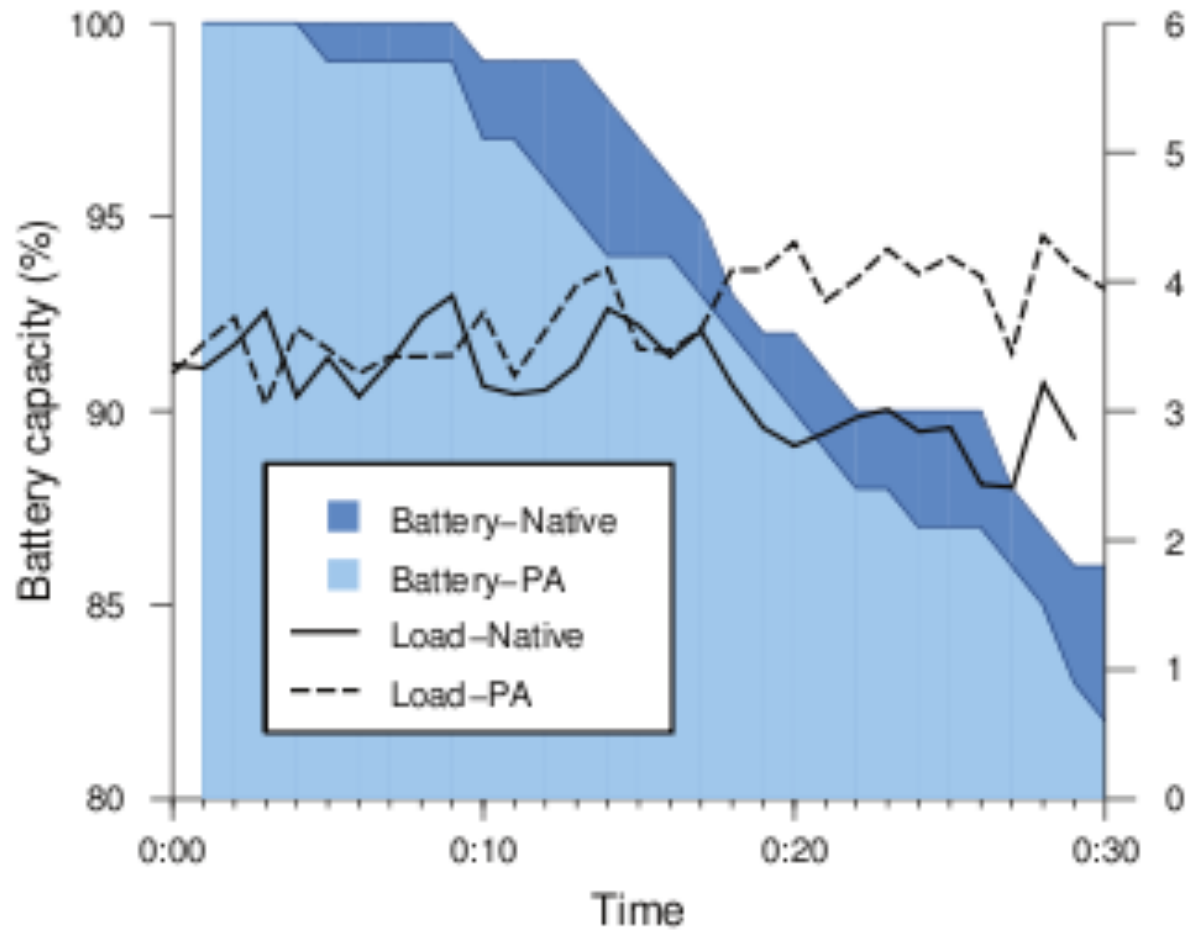
Overhead of the tracer

Performance and scalability of the server

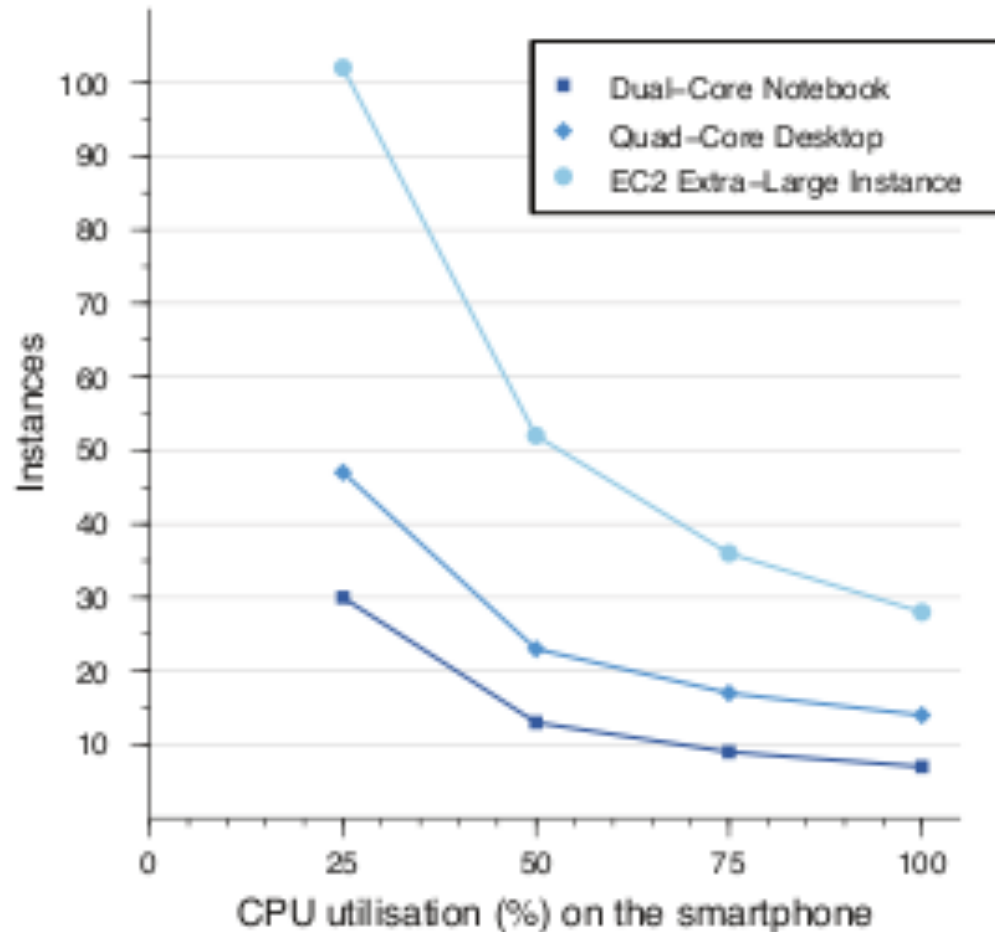
Evaluation - Amount of trace data



Evaluation - Overhead



Evaluation - Server scalability



Questions & Discussion

- Which approach do you like most? Or other ways to protect privacy?
- Will it become a necessity to run AV software on a phone?
- Has anyone installed an AV already?
- What is a better approach: restricted platforms like iOS or more open platforms like Android?

Thank you very much for your attention!