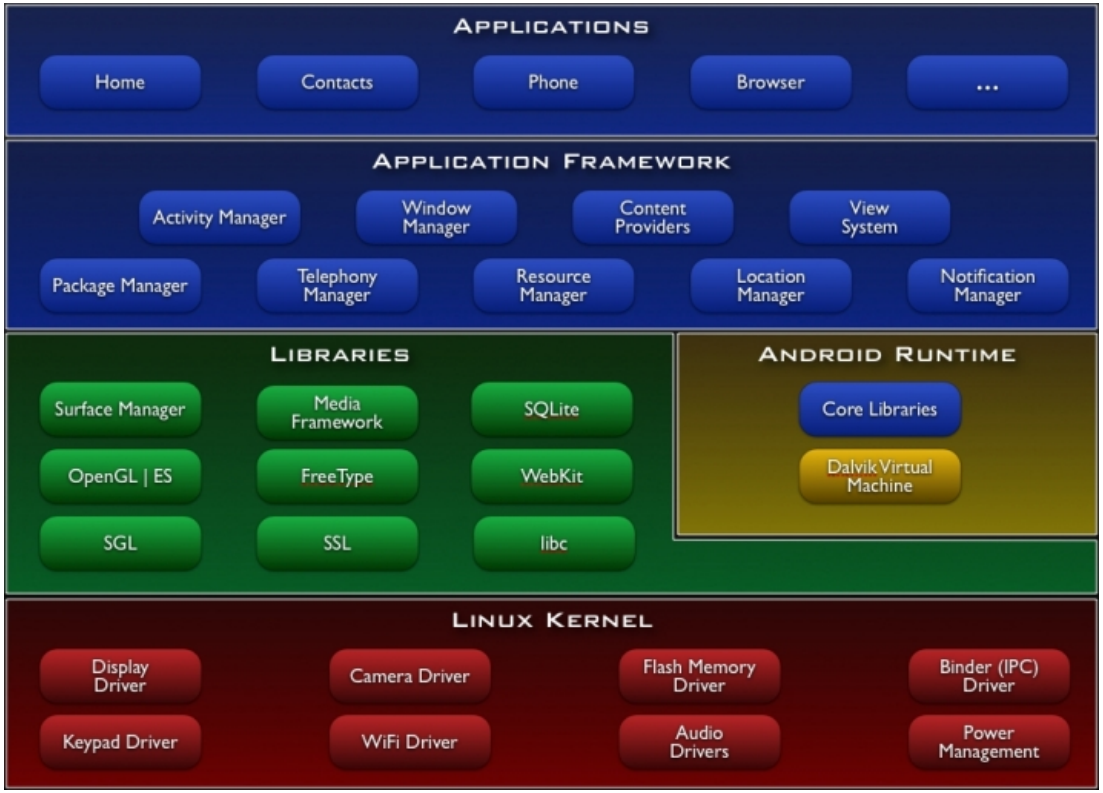
Runtime verification meets Android security

Gil Vegliach

Joint work with Andreas Bauer and Jan-Christoph Küster

Background, what Android is

- Developed by Android Inc. (acquired by Google in 2005)
- Open Handset Alliance (founded in 2007)
- Software stack for mobile devices:
  OS, middleware, key applications



Android's security model

In a nutshell...

System level protection:

- Apps are "sandboxed": unique UID ($\leftrightarrow$ Linux: one UID/user), own virtual machine
- Simple, static permission labels restrict resource access (manifest file)

Observe:

No dynamic security mechanisms

Not a bug—a feature:

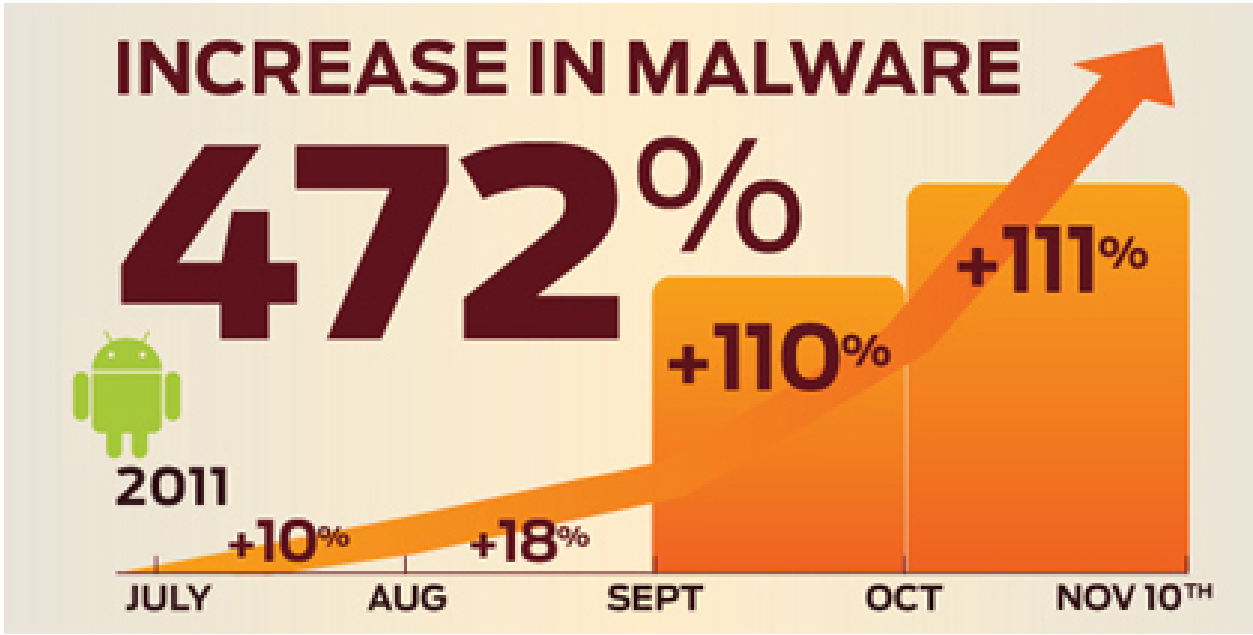
"*Android has no mechanism for granting permissions dynamically (at run-time) because it complicates the user experience to the detriment of security.*"

(Source: http://developer.android.com/guide/topics/security/security.html)

Malware is spreading out

Smart phones and tablet PCs are popular

- June ’11: 550,000 new Android devices activated every day
- (up from 400,000 per day two months earlier in May 2011)
- Security problems for mobile platforms on the rise: *“Since 2007, the number of new antivirus database records for mobile malware has virtually doubled every year.”* – Kaspersky Q1/2011



(Source: Juniper Threat Center → McAfee Q2/2011)

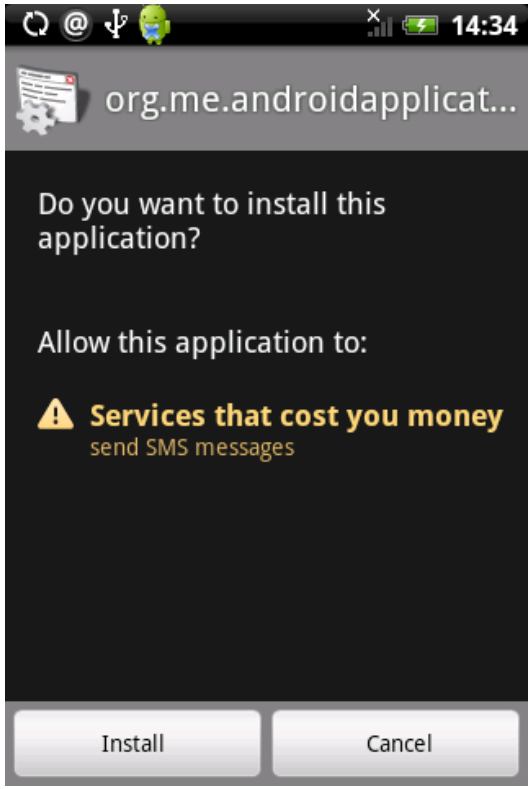
Some malware examples

Android/NickySpy.A

- Records user's phone conversations in adaptive multi-rate format (.amr)
- Stores in /sd-card/shangzhou/callrecord/
- Transmits information to (e.g.) jin.56mo.com on port 2018

Some malware examples

Trojan-SMS.AndroidOS.FakePlayer.A and spyware Android/Actrack.A



- **FakePlayer.A**: First reported in August ’10, Russian movie player sending SMS to premium Russian numbers, string: *“798657”*
- **Actrack.A**: Send GPS location, battery and radio status to a central internet server controlled by the vendor at regular intervals.

What people are doing about it

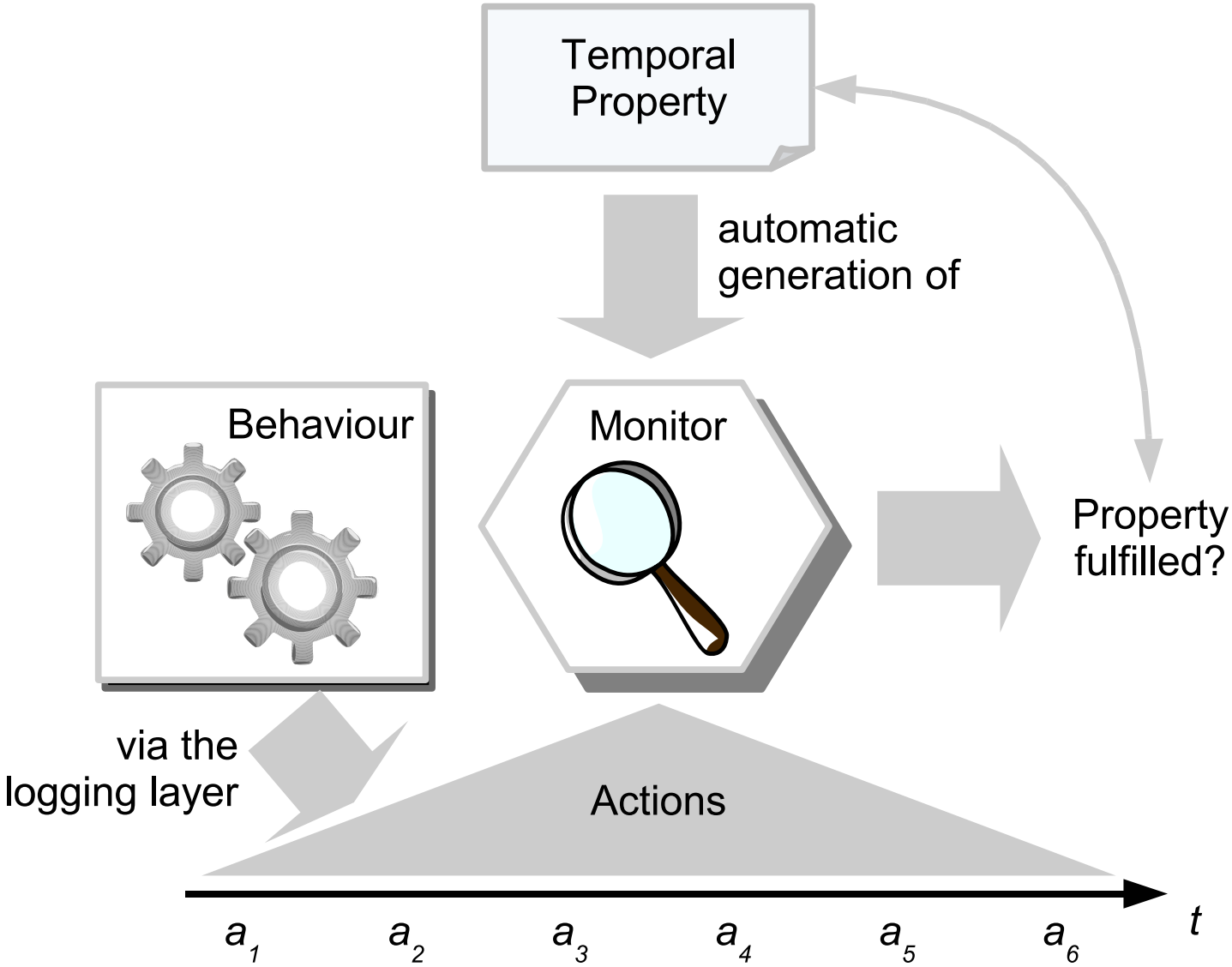
Research community

A recent “explosion” of related papers;
some of the more interesting ones:

- Static analysis of $\geq$ 1,100 Android apps
  (Enck et al, USENIX Security Symposium ’11)
- **Saint** installer (Enck et al, CCS’09)
- **TaintDroid** (Ongtang et al, ACSAC’09)
- **Soundcomber** Trojan (Schlegel et al, NDSS ’11)

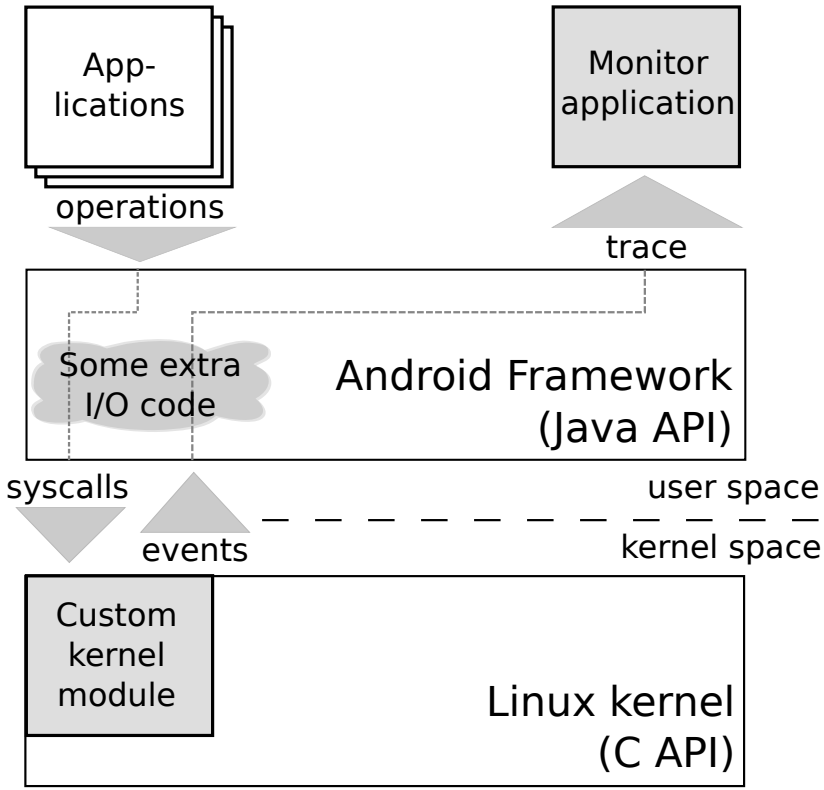
What **we** are doing about it

Runtime verification for security



Implementation

Architecture overview



- Monitor/GUI app (Java), application level
- Logging code, in the framework
- Kernel module, internet and bluetooth permissions

Not “vaporware”:

Runs on an actual phone, Samsung Nexus S

Runtime verification on Android

The policy language

Syntax

$$\varphi ::= p(t) | \neg\varphi | \varphi \wedge \varphi | \mathbf{X}\varphi | \varphi \mathbf{U} \varphi | \forall x : p.\ \varphi, \quad (p/1)$$

Ex event: { sms(123), battery(low), email("nasa@gov.com") }

Semantics

$$\begin{array}{lcl} w, i \models p(t) & \Leftrightarrow & p(t\downarrow) \in w(i) \\ \ldots & & \\ w, i \models \varphi \mathbf{U} \psi & \Leftrightarrow & \exists k \geq i.\ w, k \models \psi \wedge \forall j.\ i \leq j < k \Rightarrow w, j \models \varphi \\ w, i \models \forall x : p.\ \varphi & \Leftrightarrow & \forall c.\ p(c) \in w(i) \Rightarrow w, i \models \varphi[x/c] \end{array}$$

Ex: $\{\{p(2), p(3)\}, \{p(5)\}, \{q(4)\}^{\omega}\} \models \mathbf{G} \forall x : p.\ prime(x)$

Example policies

- **Android/NickySpy.A**: record conversation (.amr), store on sdcard, send through internet

$$\mathbf{G}\forall x : \mathit{sd\_write}.\ \mathit{amr\_file}(x) \implies (\nexists y : \mathit{connect}(y))$$

- **AndroidOS.FakePlayer.A**: send SMS to premium Russian numbers

$$\mathbf{G}\forall x : \mathit{sms}.\ \neg \mathit{sms}(x)\mathbf{U}\mathit{contact}(x)$$

- **Android/Actrack.A**: send GPS location, battery and radio status through internet

$$\mathbf{G}(\neg((\mathbf{F}\exists x : \mathit{connect}(x)) \wedge \mathit{gps}))$$

Finite trace semantics

u is finite trace of events, then:

$$u, 0 \models_3 \varphi := \begin{cases} \top & \text{if for any infinite trace } w,\ uw, 0 \models \varphi, \\ \bot & \text{if for any infinite trace } w,\ uw, 0 \not\models \varphi, \\ ? & \text{otherwise.} \end{cases}$$

That is, a monitor detects *good* and *bad prefixes* of $\mathcal{L}(\varphi)$.

Not all formulae have good and/or bad prefixes!

Why is this world-class research?

This is work in progress, so let's hope it turns into world-class research some day. :-)

But some points to notice:

- Not yet another logic looking for an application.
- Not just engineering either.
- Most related work either
  - completely modify Android framework (not portable), or
  - do not delve deep enough into the system to get meaningful information (e.g. device feature collection on the application-level)
- Our work, arguably, is sufficiently low-level, yet portable.
- To the best of our knowledge, only *behavioural detection* tool for Android in existence.

Conclusions & Future work

- Small paper accepted at *Nasa Formal Methods Symposium (NFM) 2012*: “Android security meets runtime verification”
- Proof of concept: runtime verification on mobiles
- Implemented on an actual mobile phone, run smoothly
- Need to extend pre-defined policy collections, more high-level policy language
- Need to develop further the logic

Thank you for your attention!