Technische Universität Berlin
FG Security in Telecommunications

Taming the Robot: Sandboxing Android

Matthias Lange & Steffen Liebergeld, March 24th, 2011
{mlange, steffen}@sec.t-labs.tu-berlin.de

Outline

- Introduction
- Background
  - Security Analysis
  - Virtualization
- Microkernel Systems
- Sandboxing Android
- Conclusion

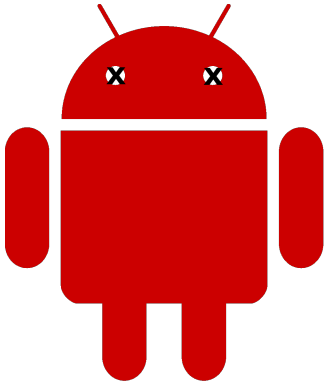
Introduction

Smartphone Market Challenges

- Loss of
  - Customer relation
  - Earnings
- Poorly written applications
- Security requirements for sensitive applications

Why Android?

- Open Source
- Custom 3rd party Apps
- Linux kernel



- Insufficient security policies
- Software not up-to-date
- Linux kernel
  - Outdated
  - Custom drivers

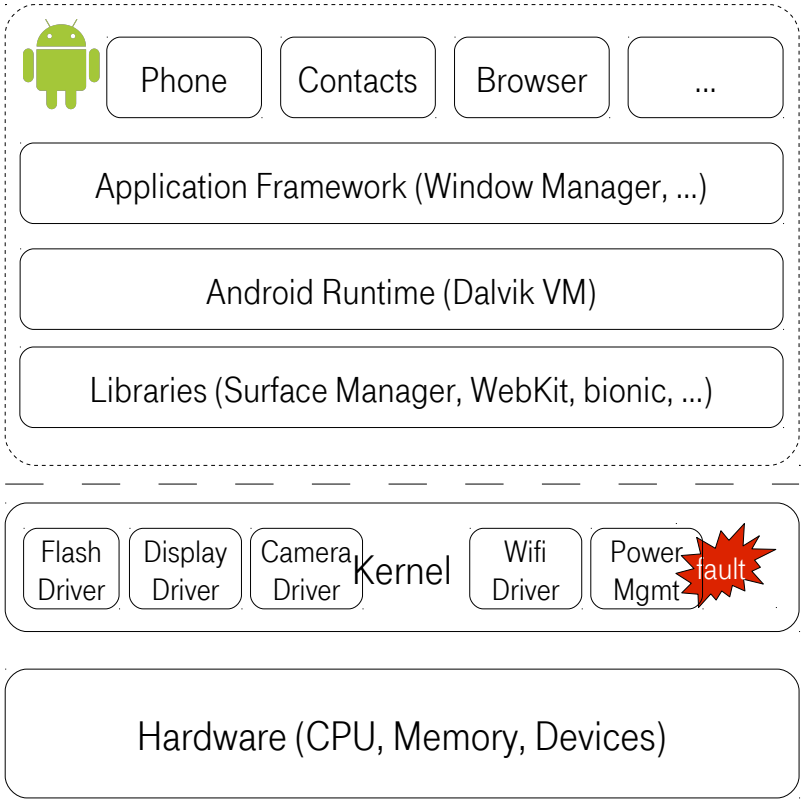
Recent Press Coverage

- Apps found to “leak” private data
- “Infected” Android Apps discovered in Android Market
  - Downloaded > 50.000 times
  - Sent private information to the attacker
- Android Trojan to send (expensive) premium SMS
- Study using static code analysis found 88 critical flaws in the kernel

Background

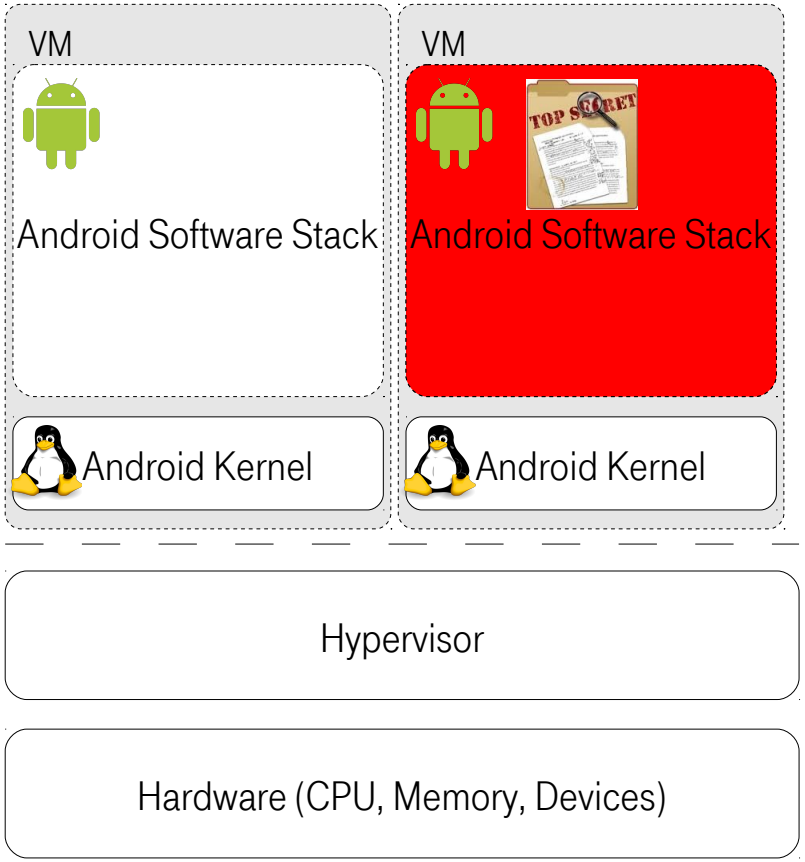
Security Analysis

- Android kernel at the lowest layer in software stack
  - Critical to availability and security
  - In TCB of all components
- Linux kernel ca. 14 million SLOC
  - Device drivers
  - Protocol stacks (e.g. network)
  - Filesystems
- Issues of monolithic kernels:
  - No in-kernel isolation
    - Any vulnerability is fatal
  - Insufficient access control mechanisms
    - ACLs, Users, Groups...



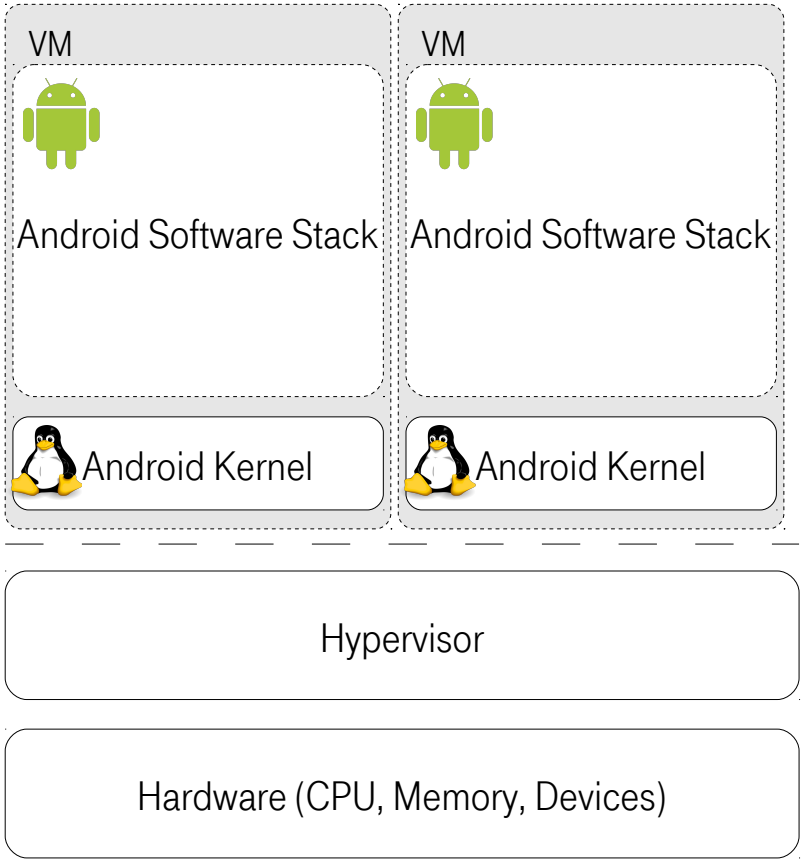
Virtualization

- Ability to run multiple instances of Android concurrently on one device
- Enables new opportunities for preventive security measures:
  - Out-of-band security analysis
  - Run security sensitive tasks besides Android (e.g. smartcard services, micropayment, eHealth)
  - Arbitrate hardware access
  - Multiple Androids with different security clearings



Virtualization - Problems

- Virtualization layer is new attack vector
- Smartphone CPUs not virtualizable
- Performance
- Needs to be done right!



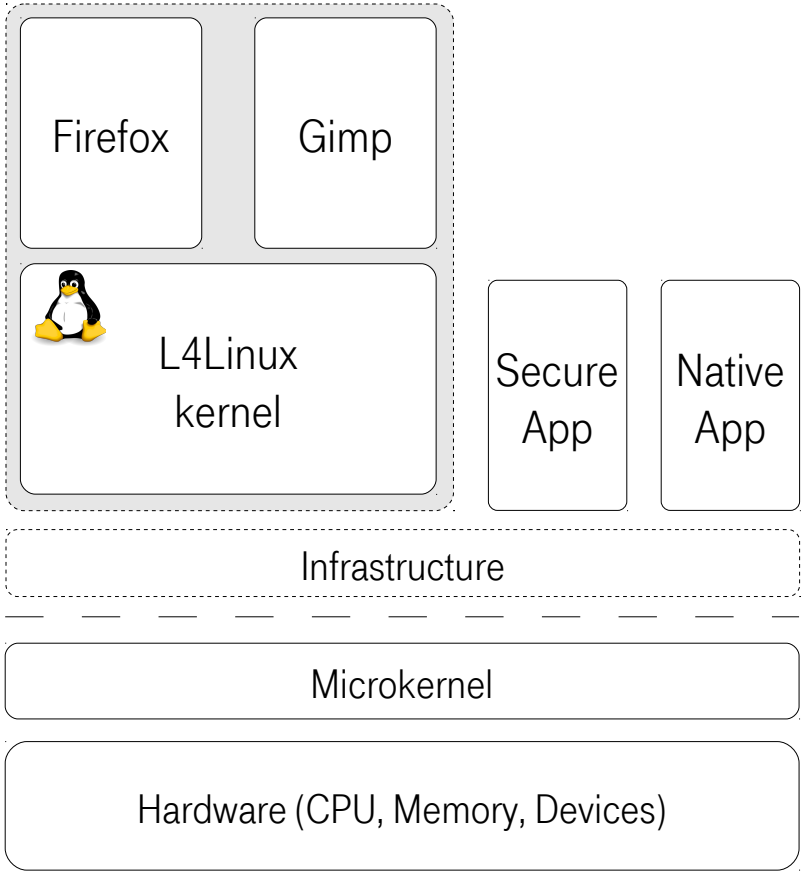
Microkernel Systems

Microkernels

- Design principles
  - Implement only functionality in kernel that cannot be implemented at user level
    - Everything else in user space
  - Hardware enforced isolation boundaries
    - Address spaces
  - Fast communication (IPC)
  - Secure access control mechanism (object capabilities)
- Improvements over monolithic kernels (such as Linux)
  - Fault isolation: limit scope of faults
  - Security: tailor TCB for each application individually and control of information flow
  - Scheduling: execute real-time applications beside non-real-time applications
- Ability to run deprivileged (para-virtualized) OS

L4Linux – Solving the Performance Problem

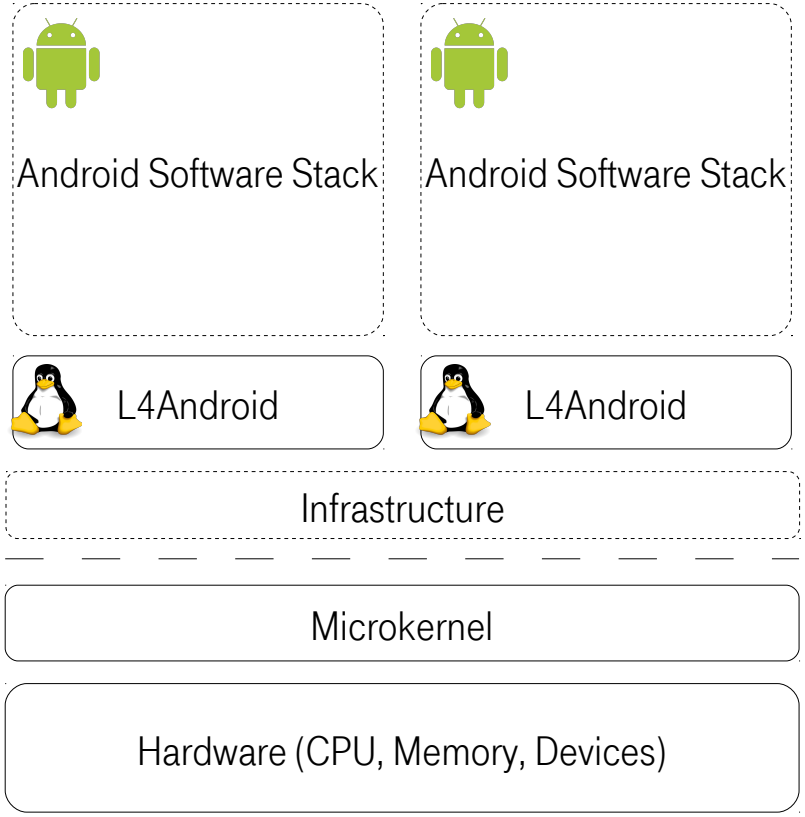
- Many Smart phone CPUs not natively virtualizable
  - Emulation (slow)
  - Binary translation (slow, huge effort)
  - De-privileging (good performance, but large initial porting effort)
- L4Linux:
  - Port of the Linux kernel
  - Runs in its own address space
  - Binary compatible at Linux kernel API
  - Current version 2.6.37
  - Applicable to non-virtualizable platforms (ARM)
  - Good performance in most workloads



Sandboxing Android

L4Android

- Make L4Linux run Android userland
  - Port of Android kernel code to L4Linux
  - Packaging of Android userland into ramdisk
  - Lots and lots of debugging
- State of the Union:
  - L4Android works (proof of concept)
  - Eclair (2.1), Froyo (2.2) and Gingerbread (2.3) supported
  - Used as research vehicle
- Work in progress:
  - Virtualize mass storage, modem
  - Implement fast and stable graphics driver



L4Android.org

- Open Source Project
- See l4android.org for details

L4Android | Download | Building | FAQ | Screenshots

L4Android

Android on top of L4

Welcome to L4Android

Recently some major and minor players proposed virtualization solutions for smartphones. At the Mobile World Congress 2011 VMware showed off two Android instances running on one smartphone, e.g. one instance for private the other for business purposes. Working on the same topic for almost a year now we thought it is time to move our Android related research project from academia to the public.
Our solution is completely open source and you can go ahead and check out the code or try it out with our demo images.

What is L4Android?

L^{4}Android is derived from the L4Linux project, which is developed at the Technische Universität Dresden. L4Linux is a modified Linux kernel, which runs on top of the Fiasco.OC microkernel. It is binary compatible with the normal Linux kernel. L^{4}Android combines both the L^{4}Linux and Google modifications of the Linux kernel and thus enables us to run Android on top of a microkernel.
Currently we support both the Froyo (2.2) and Gingerbread (2.3) release on the x86 platform. We are working on an ARM release as well.

Who we are

The L^{4}Android project is joint work of the Operating Systems Group at the Technische Universität Dresden and the chair of Security in Telecommunications at the University of Technology Berlin.

© 2011 Info

Quick Links

Who we are

Related

Fiasco.OC
L4Re
L4Linux

Meta

Twitter
Contact
Impressum

Demo

Conclusion

- Virtualization can help with security
  - (if implemented correctly)
- Microkernel forms a suitable basis
  - Provides strong isolation
  - Allows isolated high-security components (micropayment, smartcard, eHealth)
- L4Android
  - Efficient virtualized Android
  - Out-of-band security measures possible
  - Enables new business models

Questions?

Thank you!