Who am I?

Embedded Systems Developer at Bosch

Security Researcher at TU Berlin

Senior OS Engineer at Kernkonzept GmbH
Motivation
Device Accumulation

- Private
- Business
- Development
- You name it
Security

• Emerging threats

• Existing OS not secure

• Future applications
  • eHealth
  • Mobile payment (NFC)

• Encrypted voice and text
Security cont.

- "Everybody" wants its own secure smartphone
- Governments
- Businesses
Usability vs. Security

- Bring your own device
- Security == bad usability
Easy to use!
What can we do?
Patch OS?

Bad update record
Fragmentation
Add security layer?

Change middleware
Improve permission model
Virtualization
L4Android Framework

Make L4Linux run Android
Instead of this ...
... we want this
Microkernel as Hypervisor

- Less code, less errors
- Improvements over monolithic kernels
  - Fault isolation
  - Improved access control
  - Flexibility
- Needs runtime environment
Fiasco.OC + L4Re

- 3rd gen. microkernel + runtime environment
  - x86 and ARM, SMP support
  - SVM, VT-x
  - L4Re provides basic services
Virtualization: L4Linux

• Applicable to non-virtualizable platforms
• Binary compatible to native
• Up-to-date Linux
Steps and Building Blocks
Cebit 2011

- Intel Moorestown prototype
- Virtualized Android + 2nd Linux + driver Linux
- No direct hardware access
- Only touchscreen and display
  - No sensors
  - No network
Prototype Architecture

VM (private)
- Android userlevel software stack (unmodified)
- L4Android Kernel

Platform support & device drivers
- L4Linux (GPIO, Touch, Display, ...)
- mag
- fb-driv

Runtime Environment
- sigma0
- io
- moe

Fiasco.OC

VM
- Busybox
- L4Android Kernel
Implementation Steps

- Patch L4Linux with Android
  - binder, wakelocks, ashmem, …
- GPIO subsystem
  - Required for touchscreen and display
- fb-driv: MMIO replay from Linux driver
Towards the L4Android Framework

- Support for ARM and x86
- Android
  - Generic hardware interface for both architectures
  - Require no hardware modifications or extensions
Run in qemu
L4Android Architecture

- Microkernel
- Kernel
- User

Platform support & device drivers:
- Display
- Touch
- Sensors
- Power Source
- I2C
- SPI
- GPIO
- Timer
- Clocks

Runtime Environment:
- Memory Mgr
- IO Mgr
- Roottask

VM (private):
- Android (unmodified)
- Android HAL
- L4Android Kernel
- Paravirt Drivers

VM (dev):
- Android (unmodified)
- Android HAL
- L4Android Kernel
- Paravirt Drivers

VM (business):
- Android (unmodified)
- Android HAL
- L4Android Kernel
- Paravirt Drivers

–ACM SPSM ’11, Chicago, IL, USA
L4Android

- Open source project
- l4android.org

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?
The Virtual Modem*

- Prevent signaling attacks on phone
- Protect user from cellular trojans

*Taming Mr. Hayes: Mitigating Signaling Based Attacks on Smartphones, IEEE DSN 2012
Virtual Modem Architecture

Android VM
- com.android.phone
- RIL daemon
  - libsect-ril.so
  - rmnet0
  - ttyLv1
  - Virtual NIC
  - Virtual serial

L4Android Kernel

Virtual Modem
- Forwarder
  - ttyLv1
  - Virtual serial
  - eth0
  - Virtual NIC
  - ttyS0
  - rmnet0
  - IP filter
  - Baseband driver

L4Linux Kernel

Command Filter

Microkernel

User

Kernel

Baseband
Virtual Modem Results

- Mitigate known signaling attacks
- Prevent premium number SMS
- Hinders SMS controlled botnets
SiM Ko

- **Sichere Mobile Kommunikation**
- Confidential government communication
  - Data never leaves infrastructure unencrypted
- Meet BSI requirements for VS-NfD (confidential)
SiM Ko2

picture © www.telekom.de
Why SiMKo3?

- SiMKo2 problems
- Hardware EOL
- No real control
- Locked system
- No Apps
SiM Ko3 Requirements

- Open Source Solution
- Emphasis on security (MAC, small TCB)
- Reuse legacy software
  - with existing VS-NfD approval
- Commodity OS
SiM Ko3 Architecture

Multi-server OS

VM (private)
- Android (unmodified)
  - Android HAL
  - L4Android Kernel
  - Paravirt Drivers

Simple Userland, no UI
- VPN, Multiplexing
- Driver L4Linux
  - (Modem, Storage, Smartcard, Audio)

VM (business)
- Android (unmodified)
  - Android HAL
  - L4Android Kernel
  - Paravirt Drivers

Platform support & device drivers
- Display
- Touch
- Sensors
- Power Source
- Power Mgmt
- I2C
- SPI
- GPIO
- Timer
- Clocks

Runtime Environment
- Memory Mgr
- IO Mgr
- Roottask

User

Kernel

SiM Ko3 Architecture
SiMKo3 Timeline

Team of 8 persons
Gave birth to 2 startups

- **Cebit 2011**: Moorestown prototype
- **Summer 2011**: Base system with 1 Android VM on ARM and x86
- **August 2011**: Base system + 1 VM on Galaxy S2
- **Late 2011**: Input + Sensor drivers, first HAL support
- **Cebit 2012**: Galaxy S2 prototype, 2 VMs, Modem
- **Late 2012**: Galaxy S3 port, first power management
- **August 2013**: BSI approval
Crossover: Secure UI
Observation 1: Rough conditions
• One application at a time

• Fullscreen + information panel

Observation 2: Application-centric UI
• Security Level Indicator as trusted path

• Identify active environment

• Switch between VMs
• Security Level Indicator as trusted path

• Identify active environment

• Switch between VMs

• Secure global menu

• Central policy for device-global functions
• Security Level Indicator as trusted path

• Identify active environment

• Switch between VMs

• Secure global menu

• Central policy for device-global functions

• Secure Lockscreen, lock device while idle

• Central notification center
Crossover Architecture

Active VM
- Virtual Framebuffer
- Virtual Input

Background VM
- Virtual Framebuffer
- Virtual Input

Crossover GUI

Multiplexer
- Policy

Input Driver
- Display Driver

Hypervisor

Smartphone Hardware
“Crossover: Secure and Usable User Interface for Mobile Devices With Multiple Personalities”

–ACSAC 2013, New Orleans, LA, USA
“Challenges"
How to boot for fast development cycles?

Custom bootloader with USB and fastboot support.
Is there a serial interface?

Multiplexed via micro USB port or audio jack.
How to develop native drivers?

Read TRMs with 1000+ pages.
Read Linux drivers.
Reuse proprietary binary blobs?

3G baseband
Audio

Use L4Linux as driver
Power management?

- Frequency scaling
- Shutdown idle cores

Now almost equal to native
(More) Questions?