Virtualization as an Enabler for Security in Mobile Devices

Jörg Brakensiek, Nokia Research Center Palo Alto
Axel Dröge, Martin Botteck, Nokia Research Center Bochum
Hermann Härtig, Adam Lackorzynski, Technische Universität Dresden

1st Workshop on Isolation and Integration in Embedded Systems (IIES'08)
April 1, 2008, Glasgow, Scotland
Challenges for Mobile Devices – Complexity

Complexity Challenge

- Exploding number of different device configurations and platform variants
- Rapid introduction of new functionalities

This is addressed by

- Definition of hardware abstraction layers and APIs,
- Rigorous architecture management principles
- Modularization of the device architecture

![Diagram of mobile devices and system interconnect](image)
Challenges for Mobile Devices – Openness

Why opening a mobile device’s platform?
- Leverage external innovations and developments
- Broaden the application developer base
- Create the mobile platform for innovative solutions

Challenges in opening a mobile phone platform
- User as a potential attacker, having physical access to the device
  - 3rd party content and applications
  - Both regulatory and operator requirements exist
    - Manufacturer responsibility
    - Phone operation (e.g. IMEI, Network DoS, emergency call)
- Integrity, robustness and protection are contradicting openness
- External attacker (like in PC domain)
- Protection of personal content on mobile devices
Challenges for Mobile Devices – Openness

Current Solutions
- Implement devices open on OS level without regulated cellular systems
- Implement devices, having two-chip architectures, with a self-contained cellular engine
- Use strict API level certification policies, which limit access to critical API’s
- Provide managed run-times only giving limited access to critical platform resources
Virtualization Definitions

Virtualization
• Technique to emulate interfaces and behavior of a virtual system using an underlying physical system.

Virtual Machine (VM)
• SW implementation of a virtual computer system using virtualization, which allows execution of computer programs.

Virtual Machine Monitor (VMM)
• SW implementation, which emulates multiple instances of a VM and allows their configuration and management
• Also known as “Hypervisor”
Virtualization Architecture

Virtualization is applied to a given abstraction level
- Our interest: System Virtualization

Multiple Architectures
- Hosted architecture
- Our interest: Native Hypervisor

Multiple copies of the same virtual system may exist at the same time
- Our interest: RT vs. non RT
- Our interest: Linux vs. Symbian

Virtual system may be different from the underlying physical system
- Full-virtualization
- Our interest: Para-virtualization
Discussion – Para vs. Full-Virtualization

Mid-term Perspective
- Para-virtualization
  - Effort in adaptation of the guest OS is acceptable
  - Low performance overhead; guest de-privileging
  - Need committed owner for kernel adaptation

Long-term Perspective
- Full-virtualization of the CPU Core (memory management, interrupts)
  - Low performance overhead requires hardware virtualization support; no guest de-privileging
  - No kernel adaptation work needed;
- Para-virtualization of the hardware components and peripherals
  - Performance limitations & diversity of mobile platforms forbid full virtualization of whole platform
Discussion – HW Support for Virtualization

Close virtualization holes
- Do not fail silently, if called with missing privileges
- Do not show different behavior, if called with missing privileges

VMM performance / memory impact
- Virtualize page tables, TLB, Load/store CPU status registers
- Avoid unnecessary VM exits

Enhance isolation of the VM domains
- New operational mode
- IO virtualization, Interrupts, DMA

First steps in embedded domain, but learn from PC domain

Example: ARM TrustZone

Example: Intel VT-x
Discussion – Trusted Computing Base & Security

Hypervisor as an enabler of Security

- Narrow interface
- Small code size compared to commodity operating systems
- Layered security model
- Hardware security provides root of trust (e.g. secure boot)

Components to be part of the TCB

- Hypervisor Core
- Critical device drivers
- Secure Guest
- VMM Management functions

Native Hypervisors have a smaller trusted code base than hosted ones
Case Study 1 – Symbian & eCos

Motivation

• Large number of platform variants
• Remove HW dependencies within Guest OS

Requirements

• Efficient Inter-Domain-Communication
• Reuse of existing device driver frameworks

Setup

• Symbian microkernel and eCos on Symbian nanokernel
• Example device driver: Linux key pad driver
• Prototype on OMAP 2420 based phone platform
• Reasonable performance (10.5 us signaling time)
Case Study 2 – Symbian & Linux: Overview

Motivation
- Symbian EKA-2 nanokernel is not providing any isolation
- Evaluate a microkernel based VMM

Requirements
- Isolation of Guests

Setup
- HW Platform: ARM MPCore (ARM 1176)
- L4/Fiasco-based Hypervisor
- Para-Virtualized Guests
  - L4Linux-SMP
  - L4Symbian
- L4 microkernel in privileged mode
  - L4 services and guests in user mode
Case Study 2 – Symbian & Linux: Architecture

- Kernel Mode
  - L4 Microkernel
  - ARM based Hardware Platform
- User Mode
  - L4 Symbian
  - L4 Linux
  - L4 Service 1
  - L4 Service 2
  - L4 Service n
  - System-Call Interface
  - EKERNEL
  - Memory Model
  - Microkernel
  - Nanokernel
  - Kernel Extensions
  - ASSP/Variant
  - Drivers LDD
  - Drivers PDD

Symbian Application
Linux Application
Linux Kernel
Case Study 2 – Symbian & Linux: Porting Statistics

Almost all modifications done in platform dependent parts
L4Symbian binary image consists of 29 binaries
• Total size: 0.8 MB
• Plus 13 KB Bootstrap
Components heavily modified:
• Bootstrap
  • Completely re-written
• Variant/ASSP (Interrupt handling, Timer handling, Serial driver, …)
  • Touched 16 files out of 16
• EKERN (nano- & micro-kernel, memory model, …)
  • Touched 33 files out of 120

<table>
<thead>
<tr>
<th>Category</th>
<th>Porting effort</th>
<th># files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbian OS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nano-kernel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Common</td>
<td>Modified</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>As is</td>
<td>4</td>
</tr>
<tr>
<td>L4 Specific</td>
<td>Modified</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>As is</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>3</td>
</tr>
<tr>
<td>Symbian OS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>micro-kernel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Common</td>
<td>Modified</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>As is</td>
<td>42</td>
</tr>
<tr>
<td>L4 Specific</td>
<td>Modified</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>As is</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Removed</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>1</td>
</tr>
<tr>
<td>Library</td>
<td>Modified</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>As is</td>
<td>2</td>
</tr>
</tbody>
</table>
Summary and Conclusion

Cost reduction drives virtualization in the PC and desktop domain

Security in conjunction with open platforms drives it in mobile domain

- Protection from remote attackers
- Protection from attackers having physical access to the device

Experience from our two case studies:

- Para-virtualization based approach is feasible for mobile devices
- Virtualization offers a promising solution to the security challenge
- Our experiments have not encountered any fundamental difficulties

Next steps

- Further study Hypervisor impact (performance, code size, ...)
- Address critical OS features like power management
- Drive evolution in hardware support for virtualization