Secure Architecture and Implementation of Xen on ARM for Mobile Devices

Sang-bum Suh
sbuk.suh@samsung.com
SW Laboratories
CTO, Samsung Electronics
April 17, 2007

Presented at Xen Summit Spring 2007, IBM TJ Watson
Contributor

- Sang-bum Suh
- Joo-Young Hwang
- Sung-min Lee
- Sungkwan Heo
- Sangdok Mo
- ChanJu Park
- Seong-Yeol Park
- Jong-Tae Kim
- Bokdeuk Jeong
- Chul ryun Kim
- Jaemin Ryu
- Jaera Lee
- Mikhail Pozhenko
Agenda

- Requirements for Beyond 3G Mobile Device
- Goal and Approach
- Xen on ARM
  - Xen on ARM Architecture
  - System Virtualization
  - System Boot Operation
- Security
  - Security Architecture and Its Components
  - Implementation: Status
- Conclusions and Future Work
- Appendix
Requirements for Beyond 3G Mobile Devices

High-level Requirements

- **End user**: Secure and reliable mobile terminals for mobile Internet services using WiBro
- **Manufacturer**: Robustness though complexity of devices gets increased
- **Contents provider**: Protection of IP rights in end-user terminals
- **Carrier companies**: Open and Secure Mobile Platform
- **OSTI (Open Secure Terminal Initiative)**: NTT DoCoMo, Intel

Expected Beyond 3G Environments

- **m-Commerce**
- **Downloadable Application**
- **Internet/Cellular Integration**
- **Voice over IP (VoIP)**
- **Apps. & Services**
- **Multimedia Service**
- **Web Browsing**
- **Internet Banking**
- **Mobile 3D Game**
- **U-Health**

System Complexity

- **CPU**: > 500 MIPS
- **Memory**: > 64MB
- **Component Reusability**
- **High-speed Modem (10~100Mbps)**

User Needs

- **Security, Reliability**
  - (Secure Terminal)

Manufacturer Needs

- **Robustness, Time-to-market**
Threats to Mobile Devices

- According to McAfee, threats to mobile devices will continue to grow in 2007
  - The number of malware created for Windows CE/Mobile and Symbian was expected to reach 726 by the end of 2006, from an estimated 226 at the end of 2005 [KAW06]

- Attacks on mobile banking and trading
  - Steals financial data and sends them to a remote attacker
  - Examples [GOS06]

- Denial of service (DoS) attacks
  - Inappropriate execution of instructions consuming system resources (e.g., memory, CPU, battery), resetting a system
  - Examples [GOS06]
Without VMM

- m-Wallet Server (Trusted Server)
- Non-trusted Servers on Internet
- Secure Channel
- m-Wallet Client App.
- App1, App2, App3
- Unstable or Malicious App.

With VMM

- m-Wallet Server (Trusted Server)
- OTA (over-the-air) app. download
- Secure Channel
- Secure Domain
- m-Wallet Client App.
- App1, App2, App3
- Non-secure Domain
- VMM

Typical User Scenario

* VMM = Virtual Machine Monitor
Features for Secure Mobile Devices

- Low-overhead system virtualization
- Separation of guest domains
- Hot plug-in/-out of guest domains
- Secure boot
- Secure storage
- Access control
Agenda

- Requirements for Beyond 3G Mobile Device
- Goal and Approach
- Xen on ARM
  - Xen on ARM Architecture
  - System Virtualization
  - System Boot Operation
- Security
  - Security Architecture and Its Components
  - Implementation: Status
- Conclusions and Future Work
- Appendix
Goal and Approach

Goal
- Light-weight secure virtualization technology for beyond 3G mobile devices

Approach
- Design and implementation of
  - VMM on ARM using Xen architecture
  - Security features using Xen on ARM: guaranteeing confidentiality, integrity, and availability

Deliverables
- VMM: Secure Xen on ARM
- Dom0, DomU: Para-virtualized ARM Linux-2.6.11 kernel/ device drivers
Architecture: Secure Xen on ARM

Dom 0
- Application
  - Back-end Drivers
  - Native Drivers
- VM Interface
- Domain Manager
- Resource Allocator
- Access Control
- Hardware: Peripheral Devices, CPU, System Memory, Flash Memory

Dom U
- Application
  - Front-end Drivers
- VM Interface

Secure Xen on ARM
Development Environments

HW and SW Environments

A Reference System for Implementation

SW

- Xen : Xen-3.0.2
- Linux : ARM Linux-2.6.11
- GUI : Qtopia

HW

- Processor : ARM-9 266Mhz (Freescale i.MX21)
- Memory : 64MB
- Flash : NOR 32MB / NAND 64MB
- LCD : 3.5 inch
- Network : CS8900A 10Base-T Ethernet Controller

Development Environments

- OS : Fedora Core 6
- Cross-compiler: Montavista ARM GCC 3.3.1
- Debugger : Trace32 ICD (In Circuit Debugger)
Agenda

Requirements for Beyond 3G Mobile Device
Goal and Approach

Xen on ARM

Xen on ARM Architecture
System Virtualization
System Boot Operation

Security

Security Architecture and Its Components
Implementation: Status

Conclusions and Future Work

Appendix
CPU Virtualization (1/2)

Physically two privilege modes (User mode and Supervisor mode) in ARM CPU. However,

- Supervisor mode is assigned to Xen mode
- User mode is split into two logical modes (kernel and user process of Linux)
- Address space protection between kernel mode and user process mode is guaranteed by ARM domain access control mechanism.
CPU Virtualization (2/2)

Exception Handling

- Para-virtualization of system calls.
  - System calls are implemented with software interrupt.
  - In Xen on ARM, system calls are interpreted by Xen
Memory Map

Xen and guest domain (kernel + user process) are mapped on a same virtual address space.
Memory Virtualization (2/3)

- Domain Access Control is used to prevent a user process from accessing to address space of kernel in ARM CPU user mode.

- **Kernel Mode**: D0, D1, D2 enabled
- **User Process Mode**: D0, D2 enabled, D1 disabled

Virtual Address Space

- ARM Domain (Dynamic)
  - D0: S: RW, U: No Access
  - D1: S: RW, U: RW
  - D2: S: RW, U: RW
  - D3 ~ D15: reserved for future use.

**Page Table Access Permission Field (static)**

- S: ARM Supervisor mode
- U: ARM User mode

**ARM Domain access bit assignments [ARM01]**

<table>
<thead>
<tr>
<th>Access</th>
<th>Bit Field</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manager</td>
<td>11</td>
<td>No access control</td>
</tr>
<tr>
<td>Reserved</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Client (Enabled)</td>
<td>01</td>
<td>Use page table access permission field.</td>
</tr>
<tr>
<td>No Access</td>
<td>00</td>
<td></td>
</tr>
</tbody>
</table>

* S : ARM Supervisor mode
U : ARM User mode
Keep Xen address translation info from being flushed.

- After page table changes (domain/process switching), TLB entries are flushed explicitly.
- TLB lockdown mechanism provided by processor can be used to avoid TLB flushing and reloading.
- Two lockdown TLB entries used for Xen pages
  - ARM926 provides 8 lockdown TLB entries
### I/O Virtualization (1/2)

**Mixed Device Driver Architecture**

- Split device drivers and coordinated native device drivers

**Dom 0**
- Application
  - Back-end Driver
  - Native Driver

**Dom U**
- Application
  - Front-end Driver

**Communications via virtual I/O between domains**

**Route HID interrupts to the foreground domain**

**Xen on ARM**
- Ethernet
- Sound
- Flash

**HID Event Router**

**HID Devices**
- Keypad
- LCD
- Touch-Screen
- UART

**i.MX21 Platform**

---

**SW Laboratories, CTO, Samsung Electronics**
I/O Virtualization (2/2)

Mixed device driver architecture for devices shared among guest domains

- Consists of split device drivers and deterministically coordinated native device drivers
  - Split device driver model
    - Xen-compliant device driver architecture
      - E.g.: Network device, storage device, keypad device
  - Coordinated native device driver model
    - Foreground domain gets exclusive access rights to coordinated native devices
      - Coordinated native device drivers installed in each guest OS domain
      - One button in keypad is reserved to change between domains.
      - E.g.: Human Interaction Device (HID: LCD, touch screen) and UART
Xen and dom 0 kernel images are loaded at predefined memory location.

**System Boot Procedure**

- **Bootloader** Blob or u-boot
- **Xen/ARM**
  - Hardware Initialization
  - Load Kernel Image for Dom 0
  - Load and Jump to Xen Image
  - Initialize System Resources (Timer, UART, Memory, IRQ)
  - Create Dom 0
  - Execute Dom 0
  - Create / Load Guest Domains

**Load Address**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Load Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xen</td>
<td>0xC000000</td>
</tr>
<tr>
<td>Dom 0</td>
<td>0xC1C0000</td>
</tr>
</tbody>
</table>

**NOR Flash Partition for Dom 0**

- Partition 0: Xen
- Partition 1: Kernel Image
- Partition 2: File System (JFFS2)

Ex) Para-virtualized Linux

SW Laboratories, CTO, Samsung Electronics
Guest domains (dom U) are created and destroyed by a user level application, dom0_util.

Dom0_util supports only create and destroy functions.

Dom U kernel uses NAND flash memory as storage.

**VM Create / Destroy**

- Control guest domain
- Request Xen to create and execute / destroy dom U kernel, where this driver loads the kernel image.
- Create and execute dom U / destroy dom U

**Dom U kernel uses NAND flash memory as storage.**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Load Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.MX21</td>
<td>0xc3c00000</td>
</tr>
</tbody>
</table>

NAND Flash Partition for Dom 1

<table>
<thead>
<tr>
<th>Partition 0</th>
<th>Partition 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel Image</td>
<td>File System (JFFS2)</td>
</tr>
</tbody>
</table>
Agenda

- Requirements for Beyond 3G Mobile Device
- Goal and Approach
- Xen on ARM
  - Xen on ARM Architecture
  - System Virtualization
  - System Boot Operation

- Security
  - Security Architecture and Its Components
  - Implementation: Status
- Conclusions and Future Work
- Appendix
Security Architecture

Secure Domain (Dom 0)
- Secure App1
- Secure App2
- Secure App3
- AC Policy Manager
- Secure SW Installer
- Back-end device drivers
- Access Control

Open Domain (Dom U)
- App1
- App2
- App3
- App4
- App5
- Front-end device drivers
- OS

1. Hypercall
2. Access control query
3. Decision
- Decision Cache
- Hooks

Secure Xen
- Access Control Policy Conductor
- Domain Integrity Manager
- Access Control Decision Maker

Hardware Layer
- SoC
  - Secure ROM
    - Master Key (MK), Bootloader
  - CPU
- Flash Memory
  - $E_{\text{mk}}(\text{Access Control Policy})$
- Devices
Secure Boot

- **Domains**
  - Integrity check of each domain with $\text{Cert}_M$

- **Xen**
  - Integrity check of the Xen with $\text{Cert}_M$

- **SoC**
  - Secure ROM
    - Master Key (MK), Bootloader

- **Flash Memory**
  - $E_{\text{MK}}(\text{Cert}_M)$, signed Xen, signed domains ...

$E_{\text{MK}}$: Encryption with the master key (MK)

$\text{Cert}_M$: Manufacturer’s public key certificate
Secure Storage

Offset
Flash memory

Secure partition

OS image partition

Data Partition

Encrypted data

Secure ROM

Bootloader Image, Master Key (MK)

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MK</td>
<td>Master key. Each mobile device has a unique MK to encrypt data stored in secure partitions (SPs).</td>
</tr>
<tr>
<td>Certₘ</td>
<td>Manufacturer’s public key certificate. It is used for integrity measurement of Xen or kernel images.</td>
</tr>
<tr>
<td>SP₁</td>
<td>A secure partition for Xen image and data for integrity measurement during a system boot.</td>
</tr>
<tr>
<td></td>
<td>$E_{MK}(Xen\ Image</td>
</tr>
<tr>
<td>SP₂</td>
<td>A secure partition for access control policies.</td>
</tr>
<tr>
<td></td>
<td>$E_{MK}(Access\ Control\ Policies))</td>
</tr>
<tr>
<td>SP₃</td>
<td>A secure partition for cryptographic keys which are used by secure domain.</td>
</tr>
<tr>
<td></td>
<td>$E_{MK}(Cryptographic\ keys))</td>
</tr>
<tr>
<td>DPₙ</td>
<td>Partitions for guest OS domains. Each OS is allowed to access its own partition.</td>
</tr>
</tbody>
</table>

SW Laboratories, CTO, Samsung Electronics
Access Control (1/2)

- Flexible architecture based on Flask
- Objects for access control
  - Physical resources
    - Memory, CPU, IO space, IRQ, DMA
  - Virtual resources
    - Event channel, grant table
  - Domain management
    - Creation and destroy of guest domains
- Multi-layered access control not to degrade Xen performance
Access Control (2/2)

Use case

Resources which are used badly due to DoS attacks are controlled by access control module (ACM) using our proprietary policy.

Resources: CPU, memory, DMA, the number of event channel, battery

E.g.:
- ACM can control CPU time allocated to a guest domain in order to keep malware on this domain from using CPU excessively.
- If battery stock is less than a threshold, ACM shuts a guest domain down.
Implementation: Status (1/2)

- **Access control**
  - 35 access control hooks in hypercalls used for access to physical resources or virtual resources, and domain management
  - Type Enforcement (TE) policy and proprietary policy to protect a mobile device from DoS attacks
  - **Performance**
    - About 20 micro sec. per access control hook

- **Secure boot**
  - Integrity measurement of a Xen and two domains
  - **Performance**
    - About 75 ms for the integrity measurement (digital signature verification) during a system boot
Implementation: Status (2/2)

Secure storage
- Secure partitioning applied to NAND/NOR flash memory
- Secure ROM simulated by using NOR flash memory
Agenda

- Requirements for Beyond 3G Mobile Devices
- Goal and Approach
- Xen on ARM
  - Xen on ARM Architecture
  - System Virtualization
  - System Boot Operation
- Security
  - Security Architecture and Its Components
  - Implementation: Status
- Conclusions and Future Work
- Appendix
Conclusions (1/2)  
Xen on ARM for Mobile Devices

- **Requires**
  - Virtualized three CPU modes
    - Modes: Xen, kernel and user process
  - Protection of virtual address spaces for Xen, kernel and user process through domain access control
- Mixed device driver architecture for shared devices works well
  - Split device drivers and deterministically coordinated native device drivers
Conclusions (2/2)

Xen Security for Mobile Devices

**Requires**
- **Integrity measurement of core components**
- **Multi-layered access control**
  - Access control at Xen layer
    - Physical/virtual resources and domain management are enforced by ACM at Xen
  - Access control at domain layer
    - In order not to degrade Xen performance, detailed access control of the resources in each domain is individually enforced by ACM at each domain
Future Work

- Virtualization of DMA
- Merging Xenstore
- Dynamic memory allocation to guest domains
- Secure download protocol
- Study on separation of a device driver domain from guest OS kernel
- Performance analysis and optimization
Prototype Demo: Video

- **HW**: a smart phone development platform
  - CPU: ARM9, 266 MHz
  - System memory: 64 MB
  - HID: 3.5 inch LCD, touch screen, keypad
  - Storage: NAND/NOR flash memory
  - Network: Ethernet

- **SW**
  - VMM: secure Xen on ARM
  - OS: para-virtualized ARM Linux 2.6.11
  - GUI: Qtopia

- **Contents**: booting secure Xen and dom 0 (Linux), creating/destroying dom U (Linux), and etc.
References

# Comparison: Xen

<table>
<thead>
<tr>
<th>Feature</th>
<th>Xen/x86</th>
<th>Xen/ARM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booting guest domain U</td>
<td>XM</td>
<td>Lightweight version of XM</td>
</tr>
<tr>
<td>Memory allocation to domain</td>
<td>Dynamic</td>
<td>Static</td>
</tr>
<tr>
<td>Virtual Device Interface / Device Configuration</td>
<td>Xenbus / Xenstore</td>
<td>Modified Xenbus* / Proprietary (Xenstore to be implemented)</td>
</tr>
<tr>
<td>Console I/O</td>
<td>Xenconsole daemon and xenconsole client</td>
<td>Deterministically coordinated HID Device Driver</td>
</tr>
<tr>
<td>Virtual Block Device Support</td>
<td>IDE, SCSI HDD</td>
<td>NAND, NOR flash</td>
</tr>
</tbody>
</table>

Based on current status

* Modified Xenbus to support virtual I/O setup without xenstore
## Comparison: CPU

<table>
<thead>
<tr>
<th>Feature</th>
<th>x86</th>
<th>ARM v4/v5</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Privilege levels</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Software Interrupt Handling</td>
<td>Direct execution</td>
<td>Indirect execution through VMM</td>
</tr>
<tr>
<td># of sensitive instructions</td>
<td>Approx. 57 [KEV01]</td>
<td>18 [ARM01] (in case of ARM v5)</td>
</tr>
<tr>
<td>Cache Model</td>
<td>PIPT – No cache alias</td>
<td>VIVT – Cache Alias</td>
</tr>
</tbody>
</table>
## Comparison: Access Control

### sHype, XSM, and Our ACM

<table>
<thead>
<tr>
<th></th>
<th>sHype [SAI05]</th>
<th>XSM [COK06]</th>
<th>Our ACM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access Control Policies</strong></td>
<td>Flexible based on Flask (TE and Chinese Wall)</td>
<td>Flexible based on Flask (TE, Chinese Wall, RBAC, MLS, and MCS)</td>
<td>Flexible based on Flask (TE and proprietary policy)</td>
</tr>
<tr>
<td><strong>Objects of access control</strong></td>
<td>Virtual resources and domain management</td>
<td>Physical/virtual resources and domain management</td>
<td>Physical/virtual resources and domain management</td>
</tr>
<tr>
<td><strong>Protection against mobile malware-based DoS attacks</strong></td>
<td>N/A</td>
<td>N/A</td>
<td>Memory, battery, DMA, and event channels are controlled by ACM</td>
</tr>
<tr>
<td><strong>Access control to objects in each guest domain</strong></td>
<td>Enforced by ACM at VMM</td>
<td>Enforced by ACM at VMM</td>
<td>Enforced by ACM at each domain (for performance reason)</td>
</tr>
<tr>
<td><strong>Etc</strong></td>
<td></td>
<td></td>
<td>Xen on ARM specific hooks</td>
</tr>
</tbody>
</table>

**Physical/virtual resources and domain management**

- Memory, battery, DMA, and event channels are controlled by ACM.
- Physical/virtual resources and domain management.
- Access control to objects in each guest domain.
- Enforced by ACM at each domain (for performance reason).
- Xen on ARM specific hooks.
Performance (1/2)

- Bandwidth Test (LMBench): Snapshot

Bar chart showing bandwidth test results for different operations:
- Pipe
- Memory Write
- File Write
- File Copy
- Block Zero Fill
- Block Copy

Comparison between:
- Native Linux
- Para-virtualized Linux

(Values in MB/sec)
Performance (2/2)

Latency Test (LMBench): Snapshot

- File lock / unlock
- Pipe
- Semaphore
- AF_UNIX socket
- Stream
- Signal(install)
- Signal(catch)
- System Call(null)
- System Call(read)
- System Call(write)
- Process creation(proced...)

Native Linux vs Para-virtualized Linux
Xen Tools

- Python packages are too big for small flash memory.
- Smaller size by removing unused Python modules.

<table>
<thead>
<tr>
<th></th>
<th>Full Python</th>
<th>Embedded Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total size</td>
<td>40MB</td>
<td>5.7MB</td>
</tr>
<tr>
<td># of modules</td>
<td>280</td>
<td>40</td>
</tr>
</tbody>
</table>

Python version: 2.4.3
Modified Xenbus

- Modified to support virtual I/O setup without xenstore.
  - Xenstore porting is in progress.
- All configuration data is maintained in shared configuration page.
  - E.g.:
    - Event Channel No.
    - Grant Table Ref. No.
**I/O Virtualization: example**

- **Virtual Memory Technology Device (MTD) Driver**
  - To share flash memory between guest domains

![Diagram showing I/O Virtualization]

- **Application**
  - JFFS2
  - Back-end MTD Driver
  - Native MTD
  - NOR Flash Chip Driver
  - NAND Flash Chip Driver

- **Modified Xenbus**
  - help to setup
  - Virtual I/O

- **Application**
  - JFFS2
  - Front-end MTD Driver

- **Xen on ARM**
  - NOR Flash Chip
  - NAND Flash Chip

---

SW Laboratories, CTO, Samsung Electronics
I/O Virtualization: example

- **Virtual Network Driver**
  - Use synchronous I/O buffer instead of asynchronous I/O ring.
  - Transmit and receive data via shared pages
### Current Source Code Status (1/2)

**Xen/ARM (3.0.2)**

<table>
<thead>
<tr>
<th>Directory</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>security/access_control</td>
<td>2500</td>
</tr>
<tr>
<td>security/crypto</td>
<td>793</td>
</tr>
<tr>
<td>security/secure_boot</td>
<td>1500</td>
</tr>
<tr>
<td>security/secure_storage</td>
<td>720</td>
</tr>
<tr>
<td>arch/arm/xen</td>
<td>7455</td>
</tr>
<tr>
<td>arch/arm/arch-imx</td>
<td>1031</td>
</tr>
<tr>
<td>arch/arm/arch-omap</td>
<td>1127</td>
</tr>
<tr>
<td>arch/arm/lib</td>
<td>2695</td>
</tr>
<tr>
<td>include/asm-arm</td>
<td>4953</td>
</tr>
<tr>
<td>Include/asm-arm/arch-imx</td>
<td>2110</td>
</tr>
<tr>
<td>Include/asm-arm/arch-omap</td>
<td>4030</td>
</tr>
</tbody>
</table>
Para-virtualized Linux Kernel (2.6.11)

<table>
<thead>
<tr>
<th>Directory</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/arm/kernel</td>
<td>1134</td>
</tr>
<tr>
<td>arch/arm/mm</td>
<td>1730</td>
</tr>
<tr>
<td>arch/arm/mach-imx</td>
<td>1008</td>
</tr>
<tr>
<td>Include/asm-arm</td>
<td>646</td>
</tr>
</tbody>
</table>