Lightweight Virtualization Based Security Enforcement in Mobile Devices

Muhammad Abdulla

Joint work with:
Zhaohui Wang, Rahul Murmuria, and Angelos Stavrou

Center for Secure Information Systems
George Mason University
Overview

- Mobile platforms and security
- Virtualization under mobile devices
- Security enforcement
“Smart” Phones

• Better resources:
  – Computing power
  – Memory
  – Storage
  – ...

• Better task handling:
  – Email, contacts, phone calls, …

• Increased day-to-day use
  – Should be considered as part of overall security infrastructure
## Mobile Device Platforms

<table>
<thead>
<tr>
<th>Mobile Platform →</th>
<th>Android</th>
<th>iPhone</th>
<th>Palm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features ↓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development Platform</td>
<td>Android SDK</td>
<td>iPhone SDK</td>
<td>Palm WebOS</td>
</tr>
<tr>
<td>Open Source?</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Supported Languages</td>
<td>Java (with extensible C/C++ libraries)</td>
<td>Objective-C</td>
<td>Javascript/HTML</td>
</tr>
</tbody>
</table>
Security Risks from Applications

- Remote availability of applications, greater exposure to security risks:
  - Built-in applications
    - Susceptible to old and new vulnerabilities
  - Vendor-certified third-party apps
    - Behavior can be potentially time and location dependent
  - User-downloaded apps from any website or email
    - Greater risk
Got a Cool App!
(from an unknown source?)

- Pros:
  - App seems cool
  - It's free

- Cons:
  - Steal user data
  - Damage the phone
  - Spy on user
  - Attack others from user's phone
  - Attack PCs
  - Reveal user location
  - (Ab)Use phone sensors
How Do We Know App's Not Malicious?

- We don't! (at least not easily)
  - Behavior could be time / location dependent.
  - App could update itself later
  - User may not be able to detect malicious behavior
  - Not easy to test:
    - My display is broken or an app is blanking it out?
      - Hard to take out display for testing
- Even if app is not malicious, it can have security holes!
Hand-held Devices vs Conventional Platforms

- Limited resources, more weaknesses to target
  - Battery
    - Can be drained easily
  - CPU
    - High CPU usage to slow down other apps
  - Communication
    - Excessive use of network connections, etc.
Hand-held Devices vs Conventional Platforms (Cont'd)

- Tight integration of hardware:
  - Difficult to identify the source of an abnormal condition:
    - e.g. blank screen
- Weak built-in protection:
  - Directly connected to other hosts using USB or other channels: no firewalls
  - No anti-virus programs
  - …
Isolation through Virtualization

• Source of the problem:
  – Greater functionality versus greater exposure to security risks

• Proposed solution:
  – Isolate program execution through virtualization
    • Process virtual machines
      – JVM (hard to protect common resources)
    • System virtual machines
      – Xen, VMWare, VirtualBox, OpenVZ, …
  – Must be lightweight
# System-level Virtualization Solutions

<table>
<thead>
<tr>
<th></th>
<th>Paravirtualization</th>
<th>Virtual Machines</th>
<th>OS-level Virtualization</th>
<th>Process Control Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Examples</strong></td>
<td>Xen</td>
<td>VMWare, VirtualBox</td>
<td>BSD Jails, Solaris Containers, OpenVZ</td>
<td>Linux CGroups</td>
</tr>
<tr>
<td><strong>Overhead</strong></td>
<td>Execution: low</td>
<td>Execution: fair</td>
<td>Execution: low</td>
<td>Execution: low</td>
</tr>
<tr>
<td></td>
<td>Startup: high</td>
<td>Startup: high</td>
<td>Startup: fair</td>
<td>Startup: low</td>
</tr>
<tr>
<td><strong>Isolation</strong></td>
<td>Very good</td>
<td>Good</td>
<td>Good</td>
<td>Depends on implementation</td>
</tr>
</tbody>
</table>
Virtualization Goals

- Lightweight
- Process isolation
- User interaction support
- Application profiling and provisioning of system resources
- Security policy enforcement
Virtualization Solution: OpenVZ

- Platform: Google Android (open-source)

- Solutions:
  - OpenVZ
    - OS-level virtualization that provides isolation and resource management
    - Targeted towards x86-based platforms
  - Currently we have an OpenVZ Linux kernel that supports OS-level virtualization under Android/ARM platform
OpenVZ: Pros

- **Scalability**
  - As scale as the Linux kernel itself
  - 32-bit PAE: 4096 CPUs, 64GB RAM
- **Low overhead**
- **High density deployment**
  - Over 100 instances on a machine with less than 1GB RAM
- **Mass-management**
OpenVZ: Virtualization and Isolation

- Files
- Processes
- Network
- Devices
- Users
- ...

...
OpenVZ: Resource Management

- Two-level management:
  - Host decides which container should get a CPU share
  - Then a process is selected to run from that container
  - Different containers can have different priorities
  - Similar concept is used for disk quota management

- Beancounters: numfile, shmpages, ...
OpenVZ: Cons

- Not a full virtualization technology
  - Containers use the same kernel as the host (so cannot support any other OS)
- Only targeted towards the x86 platform
- Releases can fall behind mainstream Linux kernel
Virtualization Solution: CGroups

- Linux CGroups
  - Provide mechanisms for grouping and partitioning sets of tasks (and their future children)
  - Supports hierarchies with specialized behavior
    - CPU
    - Network
    - ...
  - Currently have CGroups kernel extension for battery
Cgroups: subsystems

- CPUset
  - cpuset.cpus: 0-2,16
  - cpuset.mems: 0,1,2,3
  - ...
- CPU
  - cpu.shares, cpu.rt_runtime_us, cpu.rt_period_us
- Memory: limit_in_bytes, failcnt, ...
- Network: classid (for traffic controller, `tc`)
- Freezer: freezer.state: FROZEN, FREEZING, THAWED
Resource Arrangement

Top cpuset

CPUSet1: cpu0, cpu1, cpu2
CPUSet2: cpu3

Professors
Students
Cgroups: Assigning Processes to Control Groups

- Assigning a process to a class:
  - `echo pid > /mnt/<restype>/<userclass>/tasks`

- Changing resource class:
  - `echo pid > /mnt/<restype>/<newclass>/tasks`
  - ...
  - `echo pid > /mnt/<restype>/<oldclass>/tasks`
Device Virtualization

Container 0

/\virt/dev0/fb0

graphics (/dev/fb0)

Container n

/\virt/devn/fb0

graphics (/dev/fb0)

device mapping

host graphics (/dev/fb0)
Security Enforcement Mechanisms

- **Resource Management:**
  - File system
  - CPU
  - Network
  - Battery, ...

- **Virtualization:**
  - Kernel namespaces
  - Device virtualization
  - Time virtualization
Security Enforcement Policies

- **Access control:**
  - e.g. “No app in 'game' category can access user's contacts database”
  - “No app is allowed to access display when it's not active”
- **Resource Control:**
  - “No app can use over 50% CPU at any 10 minute period”
- **Behavior:**
  - “If battery level is below 5%, only phone app is allowed to run”
Conclusions

- "Smart" devices – the new target
  - Remotely downloaded applications a security risk
  - User carries it with him/her
  - Carries Personal data (location, contacts, email)
  - Limited Defenses due to resource constraints