What If You Could Actually Trust Your Kernel?

Gernot Heiser, Leonid Ryzhyk, Michael von Tessin, Aleksander Budzynowski
NICTA and University of New South Wales, Sydney
We’ve Got a New Toy!

Initial state
- capDL Model (4,800)

Manual Spec (Isabelle/HOL)
- Abstract Model (4,900)

Executable Model (13,000)
- C Code (8,700)
- HW

Integrity (1,000)
- Haskell Prototype (5,700)

High Performance Implementation
- Hardware model

22,000 lop
117,000 lop
50,000 lop
10,000 lop

seL4: microkernel with formal proof of functional correctness

NICTA Copyright 2010
From imagination to impact
What Games Can We Play?

Obvious ones: Security
• Eg. virtualization:

Untrusted Stuff

Critical Stuff

VMM (Xen, KVM, VMware)
approx 1 Zillion LOC
approx 2–5 milli-Zillion bugs

Untrusted Stuff

Critical Stuff

Virtualiz. support

seL4
What Games Can We Play?

Obvious ones: Security

- Eg. web browsing:

- Strong isolation (like IBOS):
  - SOP enforcement
  - Minimal TCB

- ... but actual guarantees!

- More on this kind of stuff in next talk (Toby)
More Interesting: Make TPMs Useful

Trusted Platform Module (TPM)

- Provides (among others) *remote attestation*
  - Evidence of the software configuration of the machine
  - PCR register holds cumulative hashes ("measurements") of software

Diagram:
- Program
- RAM
- Log
- PCR
- Load
- Hash
- Attest
Problems with TPM

TPM asserts what has been loaded

• No protection against buggy software
  – Know what has been loaded, not that it is operating correctly!
  – Software could even be modified post-load

• Every piece of software loaded changes PCR
  – Server would need to keep hashes for every app user might load
    • Actually every distributed version of every app
  – Write your own app ⇒ attestation fails!

• Assumes no forgotten measurements
  – Eg buggy software loads code without measuring
Example: Home Banking

• Bank provides secure banking app
  – Uses remote attestation to confirm that this app is running
• But:
  – Unfeasible (and unhelpful) to allow for user’s arbitrary apps
  – Force user to boot into special banking configuration
  – User loses concurrent access to other machine features
    • Spreadsheets, address book, printer, …

⇒ Practically useless!
Late Launch / DRTM?

Dynamic root of trust, e.g. Intel TXT, AMD SVM:

- Suspends normal machine operation
- Loads specific kernel in clean environment
  - Untainted by previously loaded software
- Can remotely attest this state
- But:
  - No interrupts, DMA, multiprocessing!

⇒ Practically useless!
Practical TPM-based Solution

seL4 provides secure VM for banking app

- Runs verified loader
- Loads mini OS
  - Keyboard, mouse, display driver
  - Crypto, SSL endpoint management
  - Secure screen sharing
- Banking app runs concurrently with standard app environment
- Chain of trust for banking app:
  - seL4 (verified, changes rarely)
  - Loader (verified, no changes)
  - Mini OS (trusted)
  - Banking app (trusted)

Supports practicable and meaningful remote attestation

- Minimal and stable TCB ⇒ manageable set of measurements
Buying Performance with Reliability

Databases require durability guarantees

- In the presence of failures (OS crash, power)
- Ensured typically by write-ahead logging
  - Flush log before continuing processing
  - Disk writes on critical path
- What if you knew that your OS doesn’t crash?

![Graph showing TPC-C (Postgres) throughput with and without disk cache and sync]

Throughput (transactions / s)

<table>
<thead>
<tr>
<th>Number of clients</th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk cache, no sync()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2000</td>
<td>1500</td>
<td>1000</td>
<td>500</td>
</tr>
<tr>
<td>Disk cache, sync()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2500</td>
<td>2000</td>
<td>1500</td>
<td>1000</td>
</tr>
<tr>
<td>No disk cache, sync()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2000</td>
<td>1500</td>
<td>1000</td>
<td>500</td>
</tr>
</tbody>
</table>
DBMS with Crash-Proof OS?

Could port DBMS to run directly on seL4

Problem: costly, legacy issues, etc ⇒ not very attractive
Alternative: Use Virtualization

- No changes to DBMS or OS!
Performance

TPC-C (Postgres)

Throughput (transactions / s)

Number of clients

Disk cache, no sync
No disk cache, No sync, virtualized
No disk cache, sync(
Disk cache, sync
Disk cache, sync virtualized