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)
  - 22,000 lop
- Executable Model (13,000)
  - 117,000 lop
  - 10,000 lop
- C Code (8,700)

High Performance Implementation
- Haskell Prototype (5,700)
  - 50,000 lop

Integrity (1,000)

seL4: microkernel with formal proof of functional correctness
What Games Can We Play?

Obvious ones: Security

- Eg. virtualization:

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

  - seL4

  - Critical Stuff

  - Untrusted Stuff

  - Virtualiz. Support

  - Critical Stuff

  - Untrusted Stuff
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
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?

![Diagram showing DBMS, Log, OS, Device Driver, and TPC-C (Postgres) throughput comparison between disk cache, no sync and disk cache, sync.]
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
Disk cache, sync
No disk cache, No sync, virtualized
Disk cache, sync virtualized
No disk cache, sync()
Thank You

mailto:gernot@nicta.com.au

Google: “ertos”