NICTA

NICTA

Provable Security: How feasible is it?

Gerwin Klein, Toby Murray, Peter Gammie, Thomas Sewell and Simon Winwood

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

ANU THE AUSTRALIAN NATIONAL UNIVERSITY
UNSW THE UNIVERSITY OF NEW SOUTH WALES
NSW GOVERNMENT Industry & Investment
Victoria The Place To Be
THE UNIVERSITY OF MELBOURNE
THE UNIVERSITY OF SYDNEY
Queensland Government
Griffith UNIVERSITY
QUT Queensland University of Technology
THE UNIVERSITY OF QUEENSLAND AUSTRALIA

In Short

- Very feasible
  - For certain systems and security properties
  - But feasible does not mean easy
- Let’s stop being lame, and start doing
  - **real proofs** of
  - **real security properties** of
  - **real code** of
  - **real systems**

Real proofs

NICTA

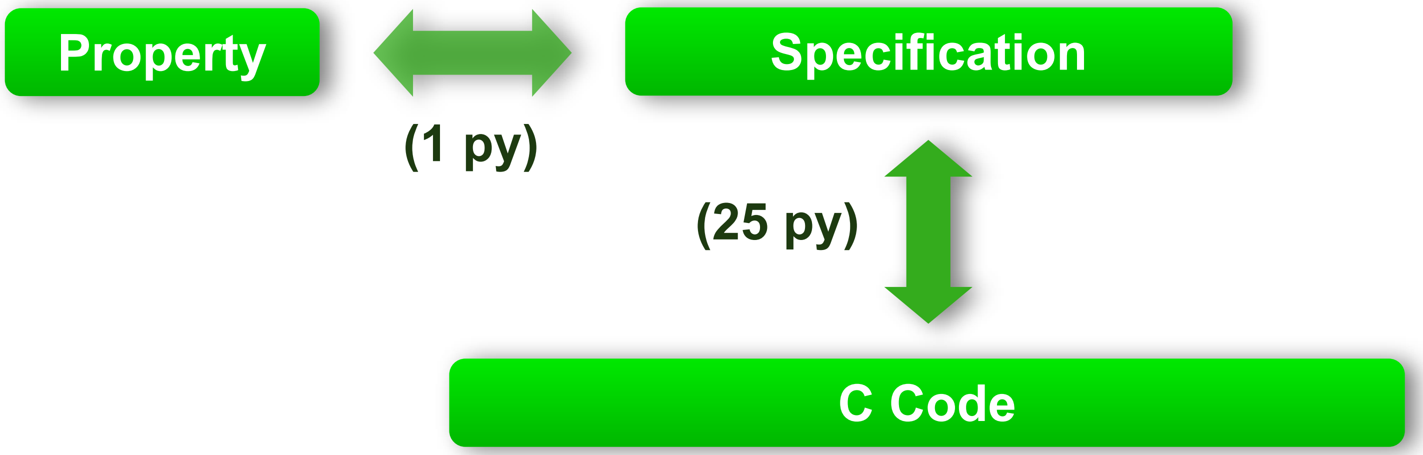
- Are not done with pen and paper
- Are machine-checked
- Turn up unexpected things you didn't know about your system or property
  - When the proof fails
  - Usually, in the more complicated parts of the API

Real security properties

- Are not absence of buffer-overflows etc.
  – (these should be trivially implied)
- Are specific to the purpose of each system
  – Are properties of whole systems
- Include high-level security goals, like:
  – Integrity, Confidentiality
- Reflect the complexities of real systems
  – e.g. authority encoded in non-cap state in seL4

Real Code

- Is not a high-level logic or language
  - Is C or assembler
- Is written to be run, not to be proved
  - Often trades-off clarity for performance
- Can be reasoned about via abstraction
  - But you have to prove the abstraction is sound



Real Systems

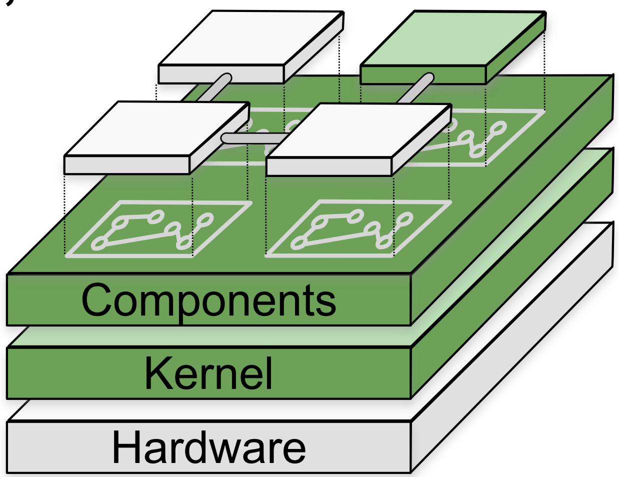
- Are deployed in the wild
- Are big (> MLOC)
- Are the imperfect results of balancing many (competing) tadeoffs
  - Performance, security, usability, simplicity
- Contain design- and implementation-quirks
  - Inevitably reflected in proofs and properties
  - May not adhere to “textbook” security defns
- Require real security properties

Example: seL4 Enforces Integrity

- Machine-checked proof (~10,000 LOC)
  - took 12 person-months (atop 30 py FC proof)
- 2-part security property of the seL4 kernel:
  - write-authority enforcement, and
  - authority-propagation
- Applies to the kernel's source code
  - Reflects the curiosities of the seL4 API
- Is a general property about the kernel
  - not yet fully applied to a specific system

The Immediate Horizon

- Security Properties
  - Integrity
  - Confidentiality excluding timing channels (e.g. untimed noninterference)
- Systems
  - MILS architectures with few, small (~10,000 LOC each) trusted components,
  - built atop small, proven kernels



What Is Still Too Hard

- Proving the absence of timing channels
  – Requires very detailed model of hardware
  – Likely infeasible on high-performance, commodity hardware
  – Will have to live with mitigation only, or use custom hardware that allows OS to carefully control timing effects
- Systems with large trusted components
  – Linux, Windows

Conclusion

- Real kernels need real security properties
- Now feasible to prove for small kernels
  - – And carefully architected whole-systems
- Not all properties are feasible
  - – e.g. absence of timing channels
  - – But this is still a huge step forward
- Security-critical systems demand real proofs of their code
  - – Not only necessary, but now feasible at reasonable cost

Thank You