seL4 & family:
fast, trustworthy, cheap, deployed

June Andronick & the Trustworthy Systems Group

January 2018

http://trustworthy.systems/

We offer:
VERIFIED *and* FAST

We're adding:
***and* CHEAP**

We need ***all 3*** to get
DEPLOYED

Overview

Making verified software a **reality** in real-world systems

Remaining challenges to **mainstream** verified software

Approach:
- ➔ minimal & verified TCB
- ➔ ecosystem: seL4&co

Deployment
- ➔ projects
- ➔ community!

cheaper ➔ proofs for free

relevant ➔ more features

scale ➔ proof engineering





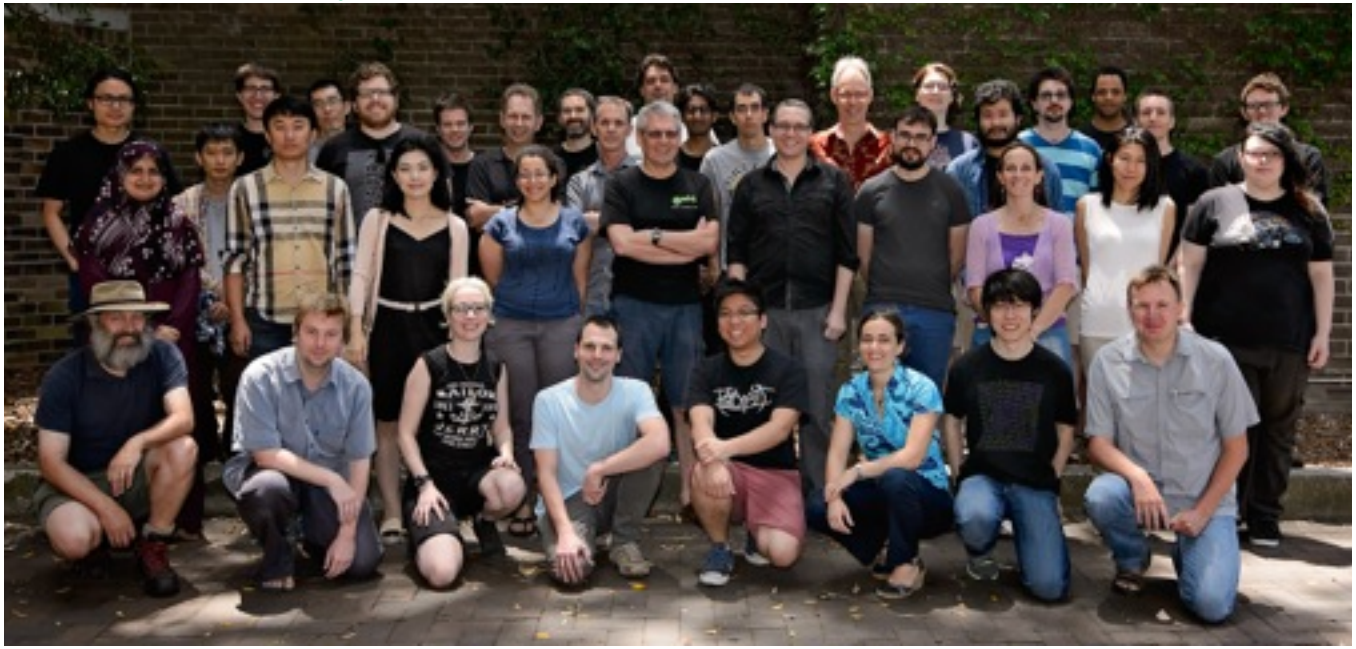

TS @ Data61

https://trustworthy.systems/

The Trustworthy Systems group is a set of people with a mission

experts in
formal methods,
operating systems,
programming languages,
security

provide the world with
deployable,
truly trustworthy
software systems



TS @ Data61

https://trustworthy.systems/

The Trustworthy Systems group is a set of people with a mission

experts in
formal methods,
operating systems,
programming languages,
security

provide the world with
deployable,
truly trustworthy
software systems

Key differentiator:

- combination of expertises
- combination of research and engineering
- critical mass

Key differentiator

- strength of mathematical proof, to highest standards
- high performance for real-world impact and deployment

Overview

Making verified software a **reality** in real-world systems

Remaining challenges to **mainstream** verified software

Approach:
- ➔ minimal & verified TCB
- ➔ ecosystem: seL4&co

Deployment
- ➔ projects
- ➔ community!

cheaper

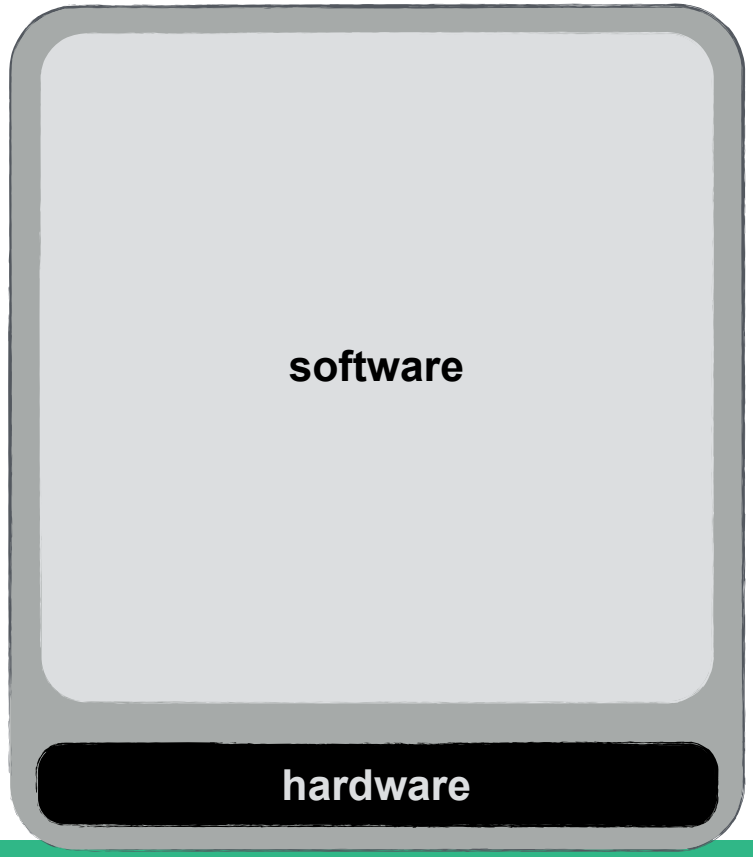
relevant

scale





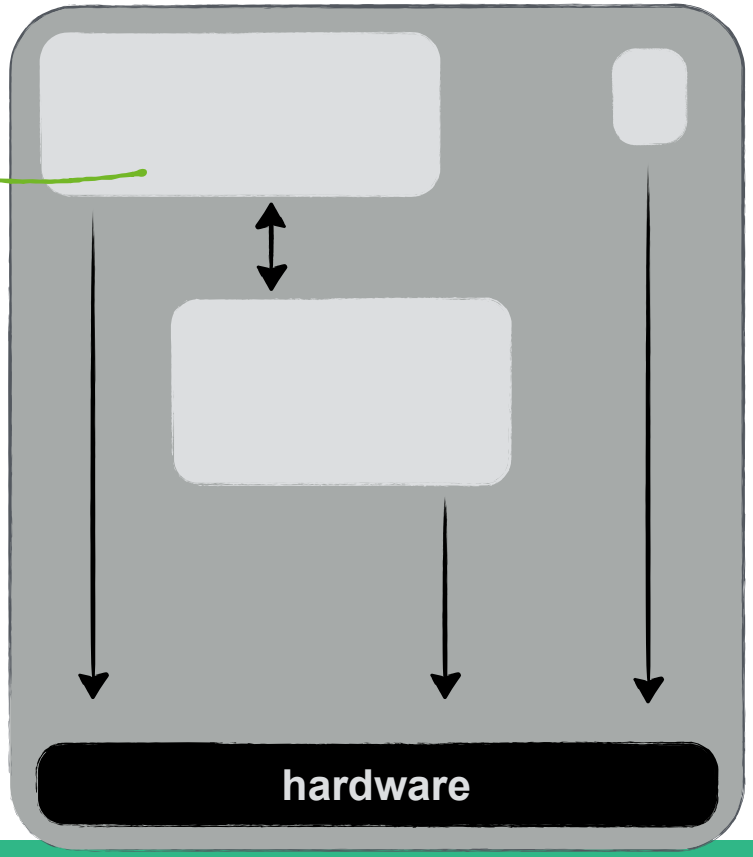

Our approach



Our approach

Componentized architecture

careful design



Our approach

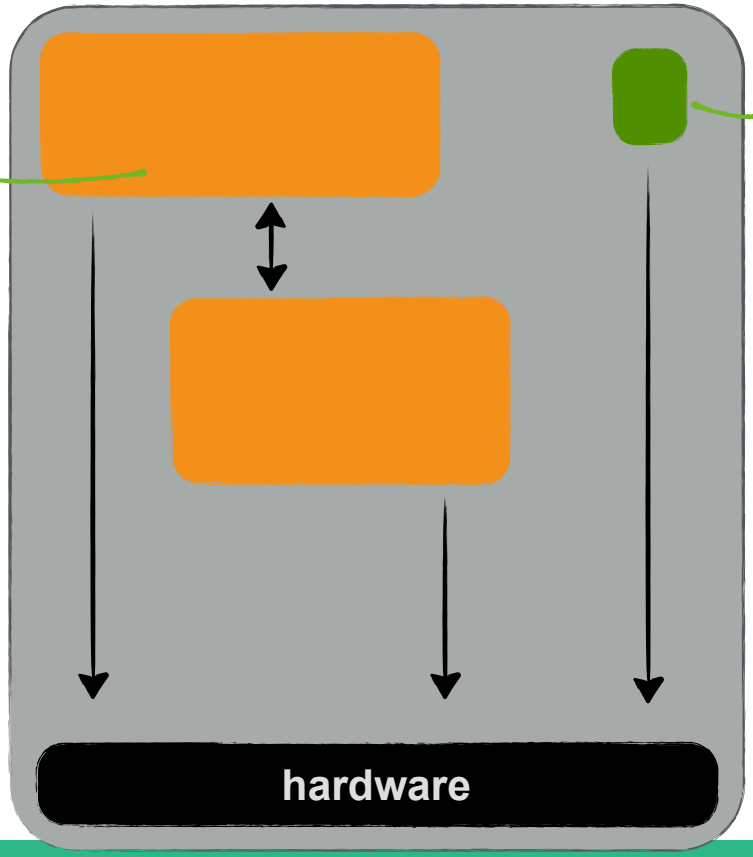
Componentized architecture

careful design isolating trusted and untrusted part of the system

Minimal TCB

(Trusted Computing Base)

limited number of trusted components



critical/trusted

uncritical/untrusted

Our approach

componentized architecture, with minimal TCB, on a trustworthy foundation

→ **seL4 & family (CAmkES, etc)**

Componentized architecture

careful design isolating trusted and untrusted part of the system

VERIFIED

Kernel-based system

enforcing access control and isolation

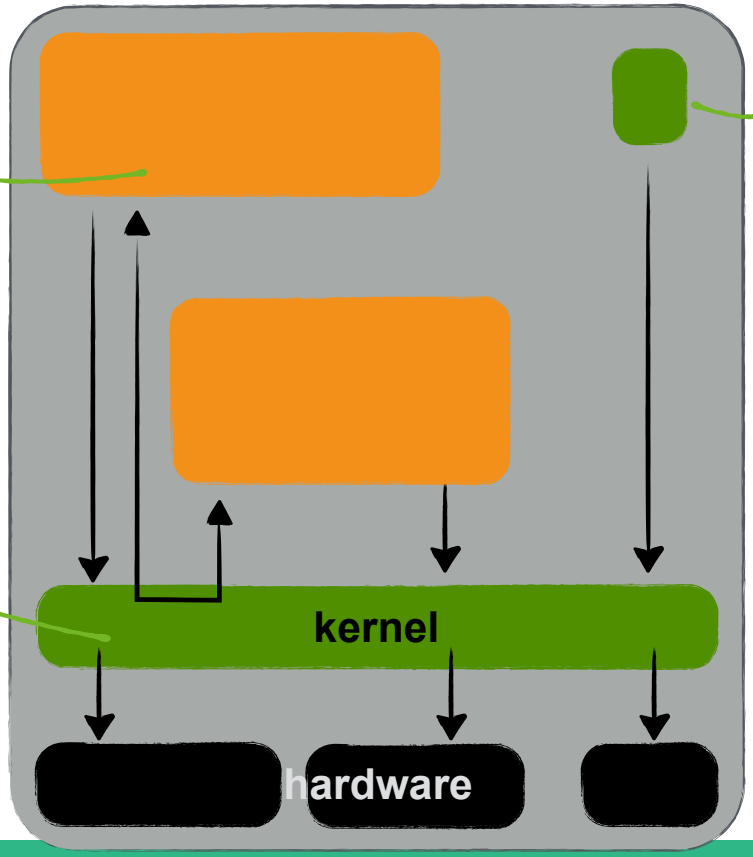
VERIFIED

Minimal TCB

(Trusted Computing Base)

limited number of trusted components

VERIFIED



seL4 in 1 slide

seL4

Security. Performance. Proof.

Small,

fast,

capability-based,

operating system kernel

~10,000 lines of C and ASM code

⬇

Small attack surface,
More amenable to full verification

Unprivileged code needs special permission (capability) to access resources and communicate

⬇

Kernel can confine damage from attacks in unprivileged code

World's fastest (5–10 faster) operating system designed for security/safety

⬇

Suitable for real-world deployment

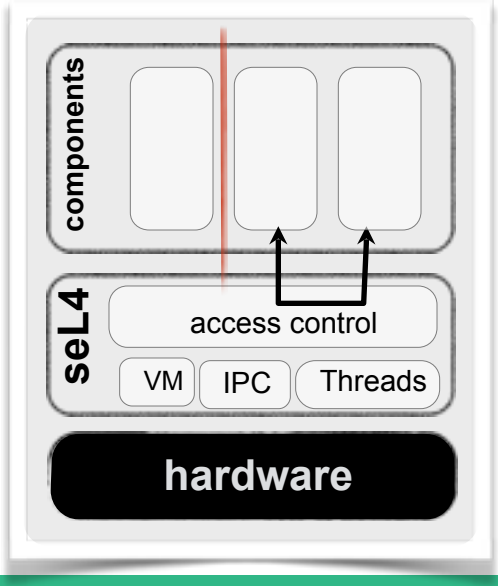
Code that runs in privileged mode of the hardware

⬇

Most critical part

Unprivileged mode

Privileged mode



seL4 in 1 slide

seL4

Security. Performance. Proof.

Small,

fast,

capability-based,

operating system kernel

~10,000 lines of C and ASM code

⬇

Small attack surface,
More amenable to full verification

Unprivileged code needs special permission (capability) to access resources and communicate

⬇

Kernel can confine damage from attacks in unprivileged code

World's fastest (5–10 faster) operating system designed for security/safety

⬇

Suitable for real-world deployment

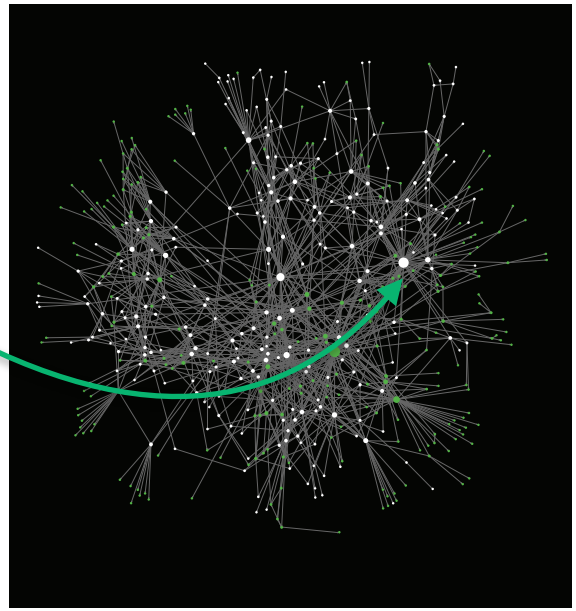
Code that runs in privileged mode of the hardware

⬇

Most critical part

```
void kernel_call ()
  {
     …
     …
     …
  }
```

(>500 functions)



What makes seL4 truly unique?

Security. Performance. Proof.

"World's most verified kernel"

Mathematical proof that code is **correct** w.r.t. specification,

Mathematical proof that it enforces strong **security** properties,

Proved safe upper bounds on their **worst-case execution times**

What it means:

Execution of kernel always defined:

- no null pointer de-reference
- no buffer overflows
- no code injection
- no memory leaks/out of kernel memory
- no div by zero, no undefined behavior
- no undefined execution
- no infinite loops/recursion

Even stronger:

- all the possible behaviours of the binary conform to spec
- security policies are enforced

Assumptions:

- Correct assembly code
- Correct hardware behaviours
- Correct hardware management (TLB and caches)
- Correct boot code
- DMA off or trusted
- Secure configuration

What makes seL4 truly unique?

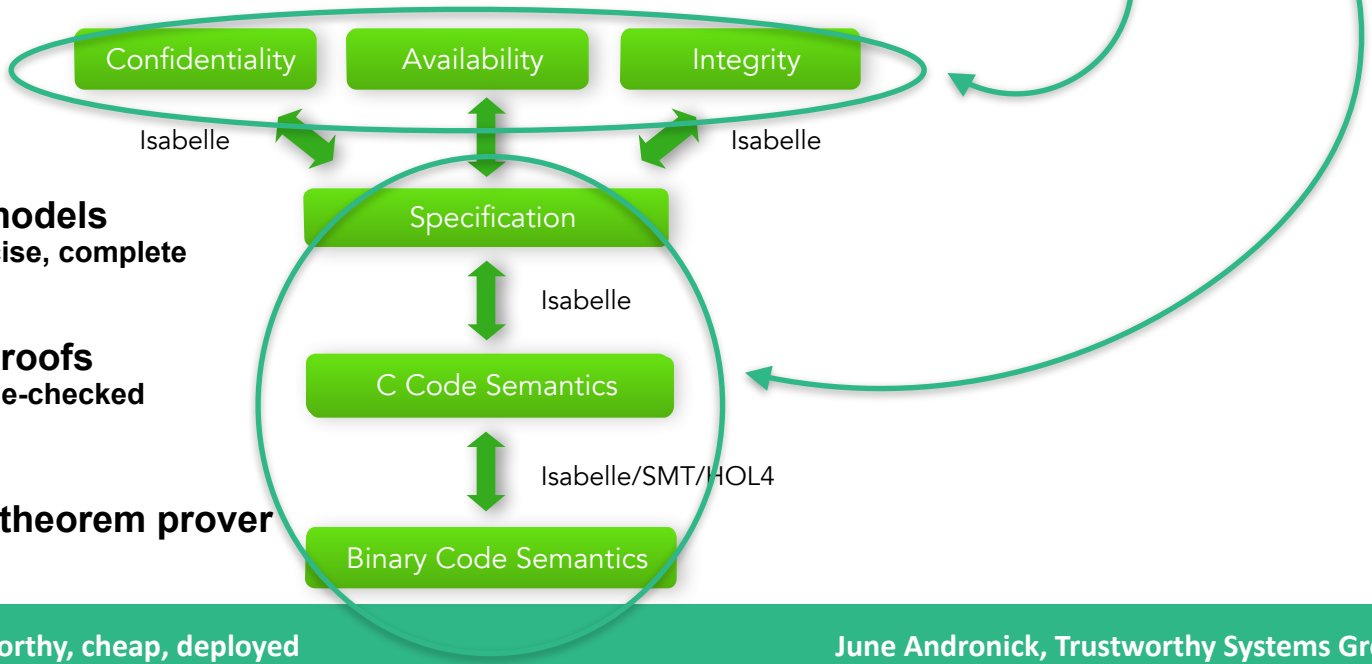
Security. Performance. Proof.

"World's most verified kernel"

Mathematical proof that code is **correct** w.r.t. specification,

Mathematical proof that it enforces strong **security** properties,

Proved safe upper bounds on their **worst-case execution times**



Mathematical models
unambiguous, precise, complete

Mathematical proofs
exhaustive, machine-checked

Using Isabelle theorem prover

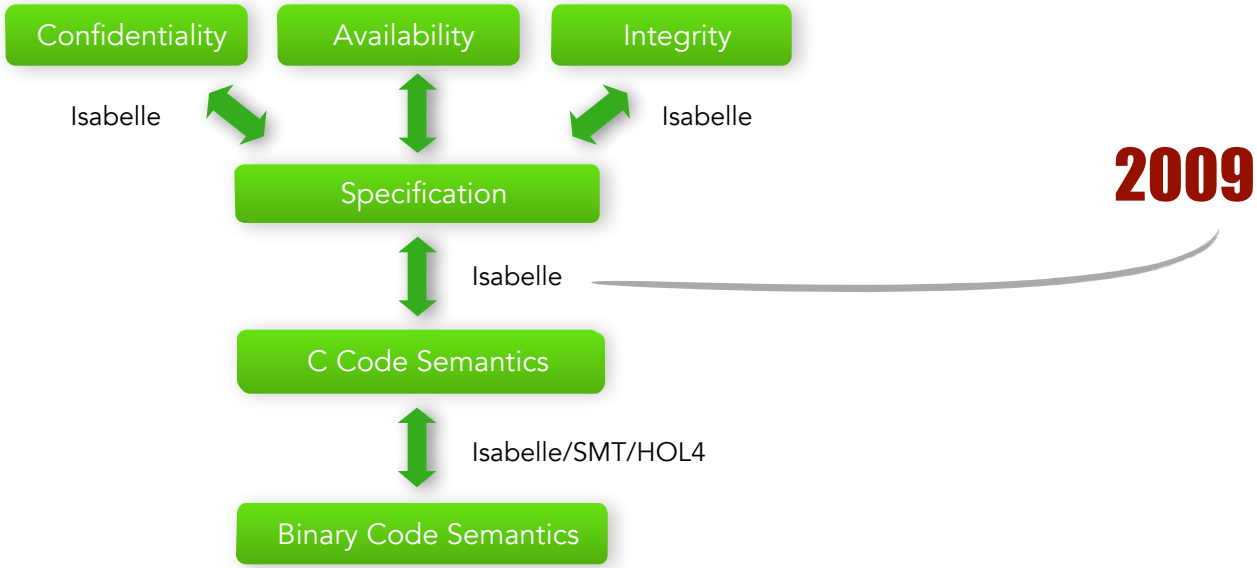
What makes seL4 truly unique?

"World's most verified kernel"

Mathematical proof that code is **correct** w.r.t. specification,

Mathematical proof that it enforces strong **security** properties,

Proved safe upper bounds on their **worst-case execution times**




10 BREAKTHROUGH TECHNOLOGIES
2011
DATA 61
CSIRO
AUSTRALIAN IT
Nicta could make billions
COMPUTERWORLD
NICTA wins race to secure L4
Dr.Dobb's Journal Presents:
Dr. Dobb's CodeTalk
100% Verifiable Bug-Free Code Is Possible
Slashdot
NEWS FOR NERDS. STUFF THAT MATTERS.
Technology: World's First Formally-Proven OS Kernel
The Register
Biting the hand that feeds IT
Researchers forge secure kernel from maths proofs
New Scientist
Saturday 29/8/2009
The ultimate way to keep your computer safe from harm
the ENGINEER online
Safer software
itnews
FOR AUSTRALIAN BUSINESS
Top secret trials for NICTA's kernel breakthrough
Sicherheits-Beweis für Betriebssystem-Kernel
17.08.2009
Тэги: seL4
Создан безошибочный код из 7500 строк
28.09.2009 [15:33], Денис Борн

What makes seL4 truly unique?

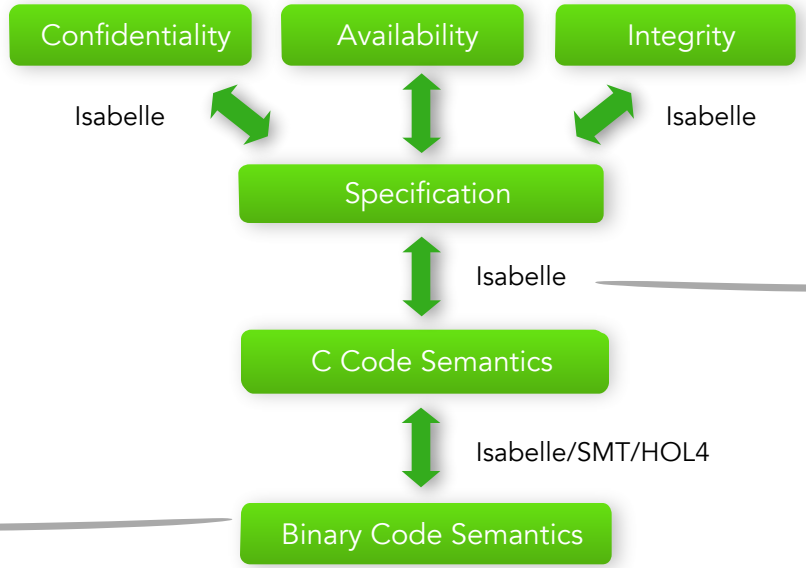
"World's most verified kernel"

Mathematical proof that code is **correct** w.r.t. specification,

Mathematical proof that it enforces strong **security** properties,

Proved safe upper bounds on their **worst-case execution times**

2012



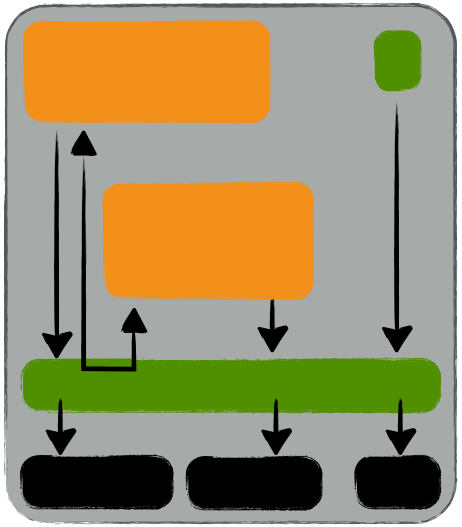
2009

2013

Building systems

Security. Performance. Proof.

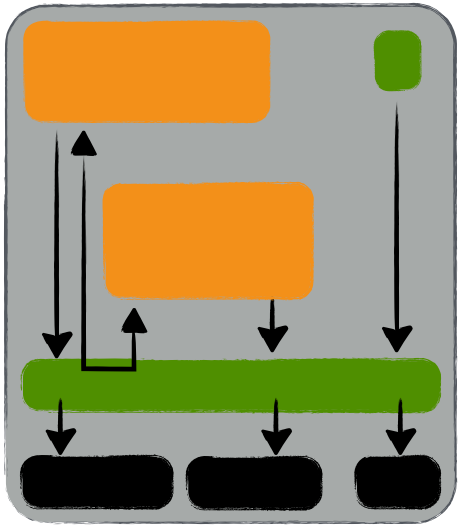
Key: proved **isolation**

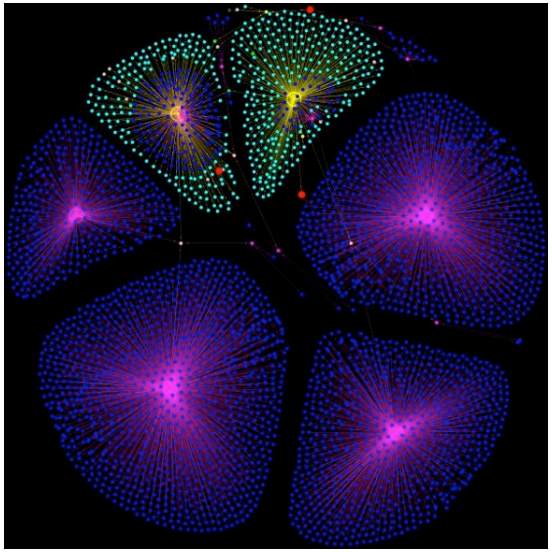


Building systems

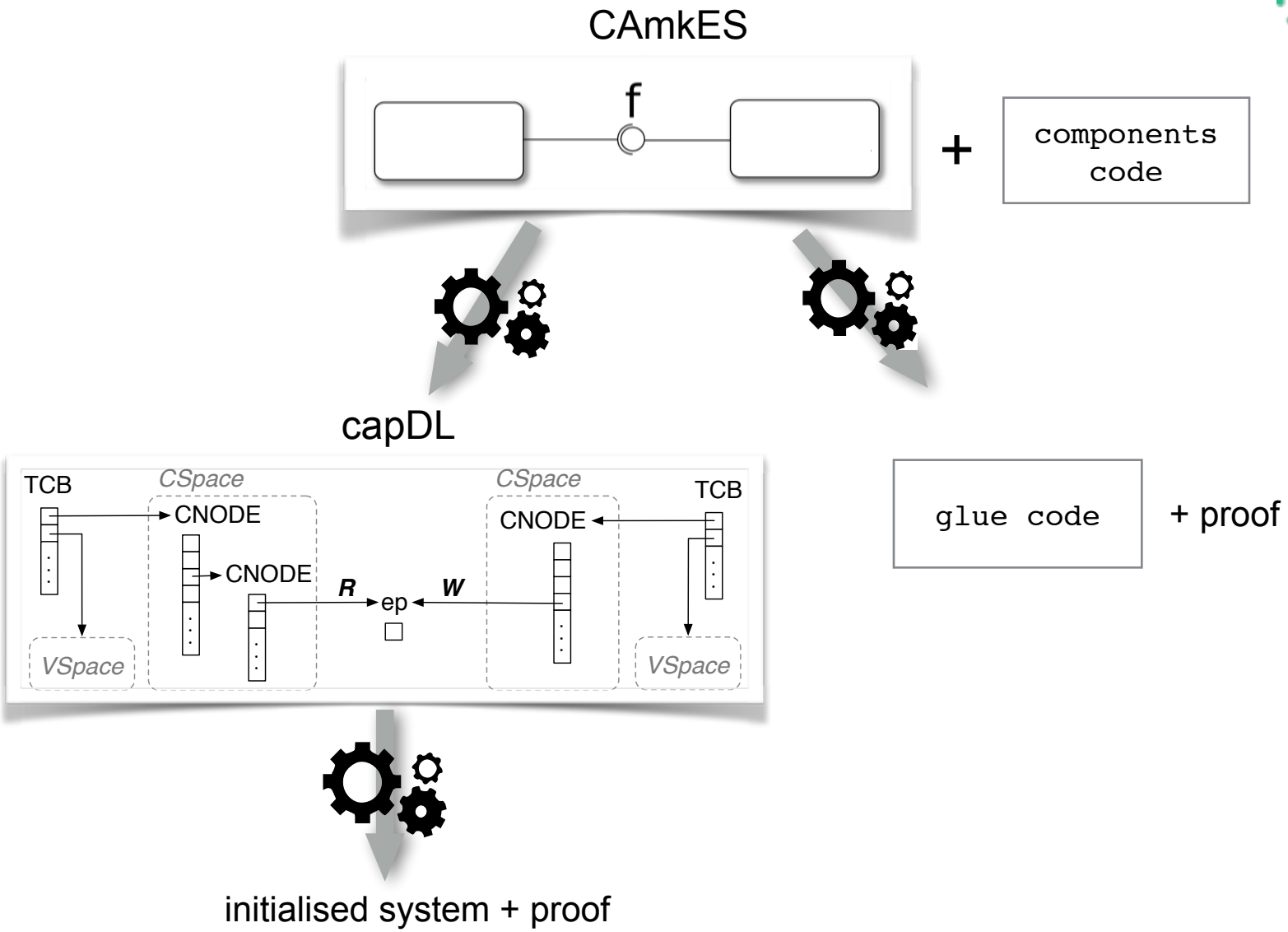
Security. Performance. Proof.

Key: proved **isolation**

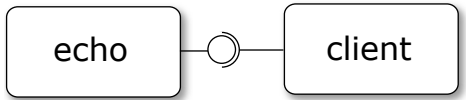


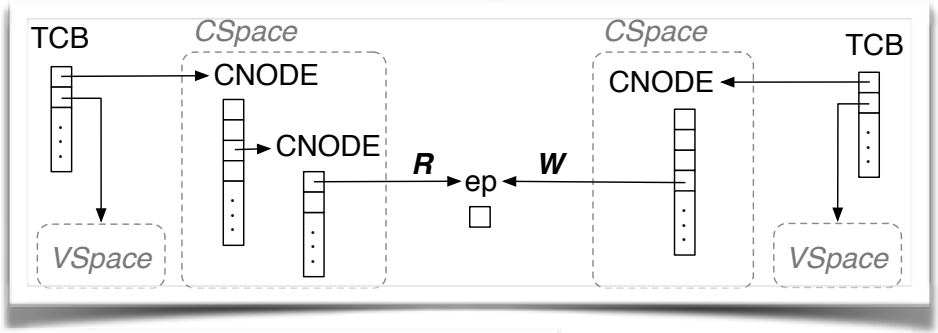


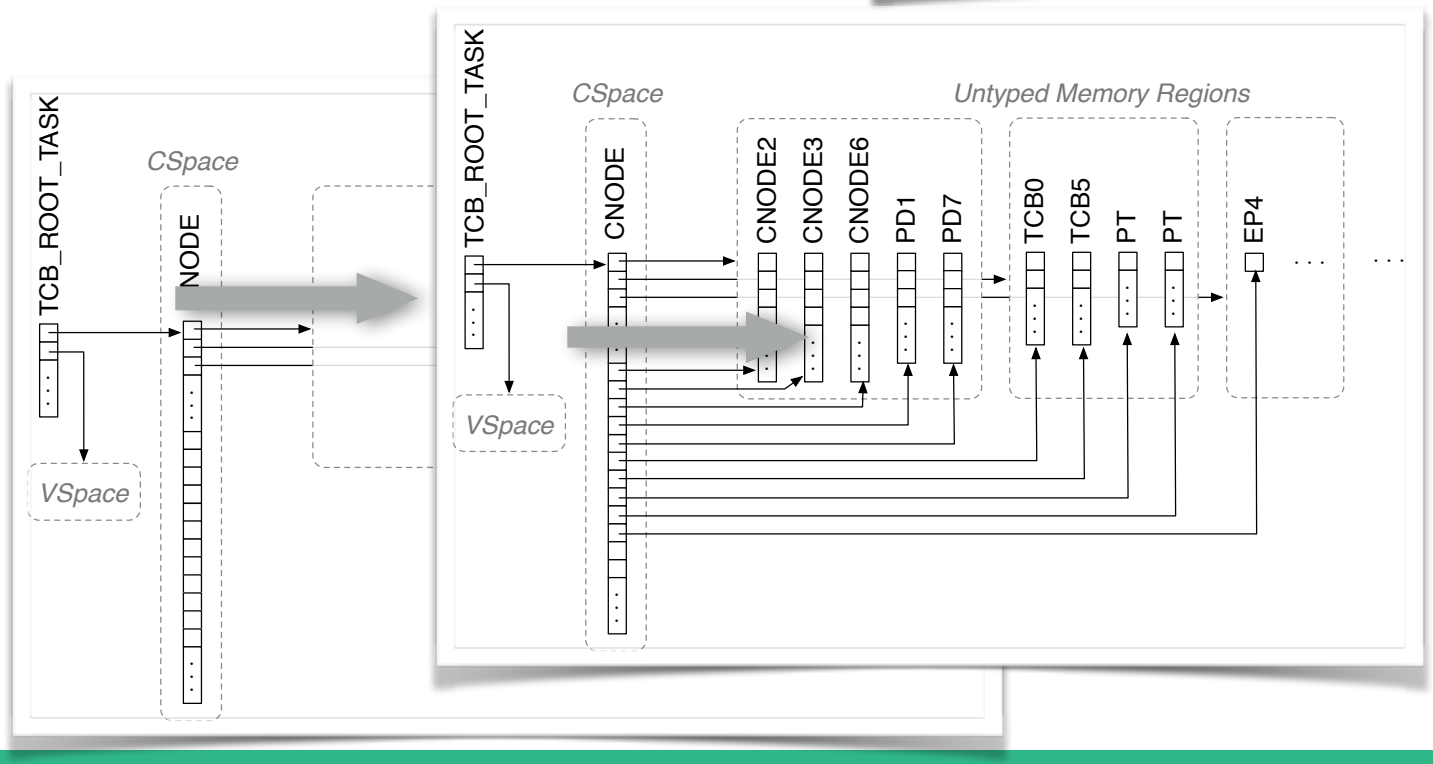
Building systems - component platform



Building systems - initialisation



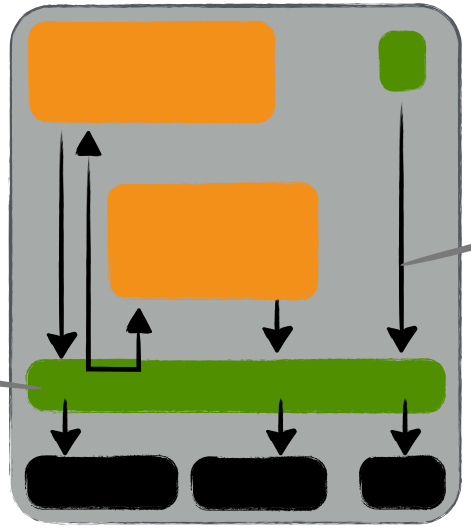




seL4 & family: an ecosystem

seL4 Kernel
- Platform ports
- Performance, debugging
- Proofs

seL4 Platform
- Libraries, CAmkES, Driver framework…
- OS system components, VMM…



Development support
- seL4test, continuous integration
- Debugging
- Benchmarking

Support
- Documentation
- Community support

Deployment: in DARPA HACMS project

→ Verified software does protect against cyber attacks

Quadcopter

Land robot

Unmanned Helicopter

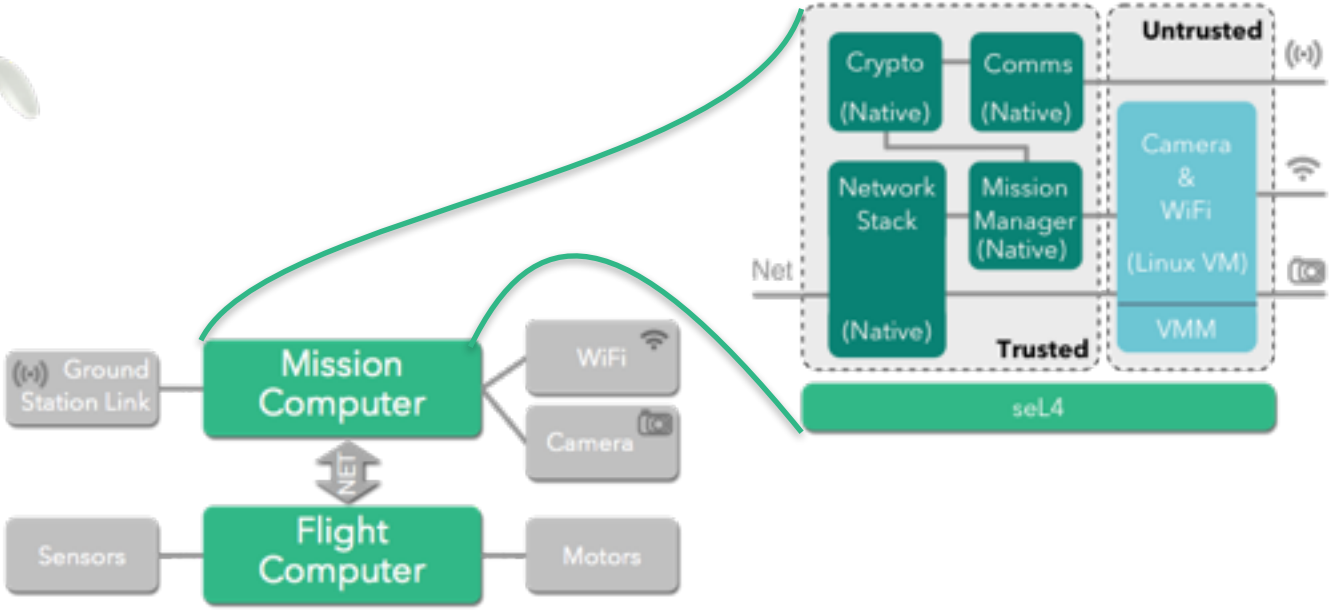
Autonomous Trucks

Deployment: in DARPA HACMS project

→ componentisation of unmanned air vehicles

Quadcopter

Unmanned Helicopter

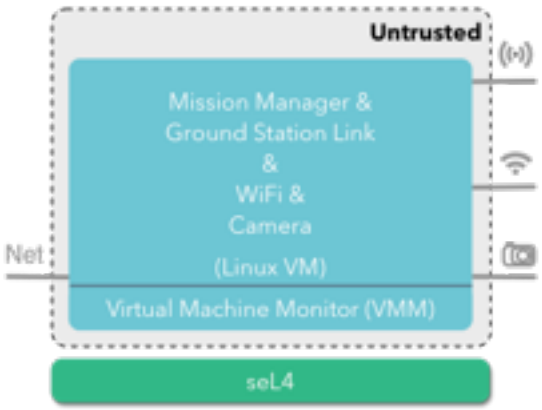


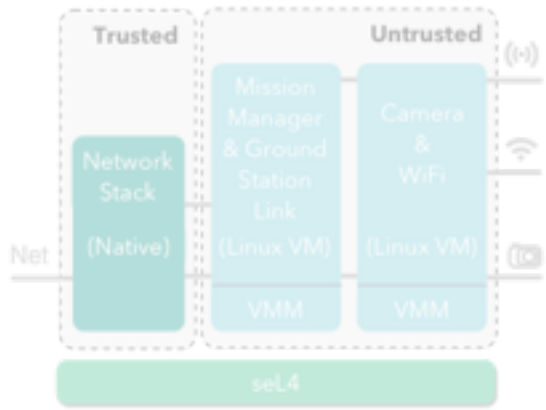
seL4 inside!

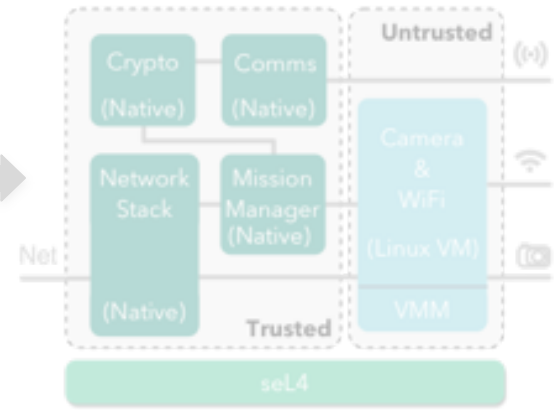
Security. Performance. Proof.

Deployment: in DARPA HACMS project

→ Retrofitting a system to be high-assurance







First put all of the existing software inside a VM running on seL4

⬇

No security benefit yet,

simply showing that seL4 runs on the target platform and that all the software can run virtualised

Then start pulling **some** trusted components out of the VM to run natively on seL4

⬇

Some security benefit:

compromise in VM cannot propagate to trusted component

Full security architecture, with **all** trusted components running as a seL4 component

⬇

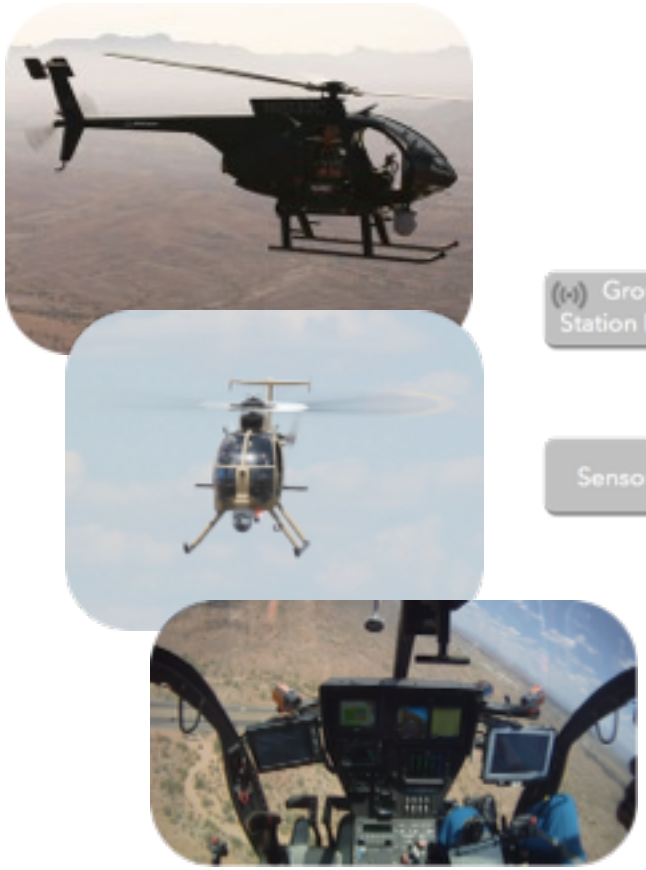
Important security benefit:

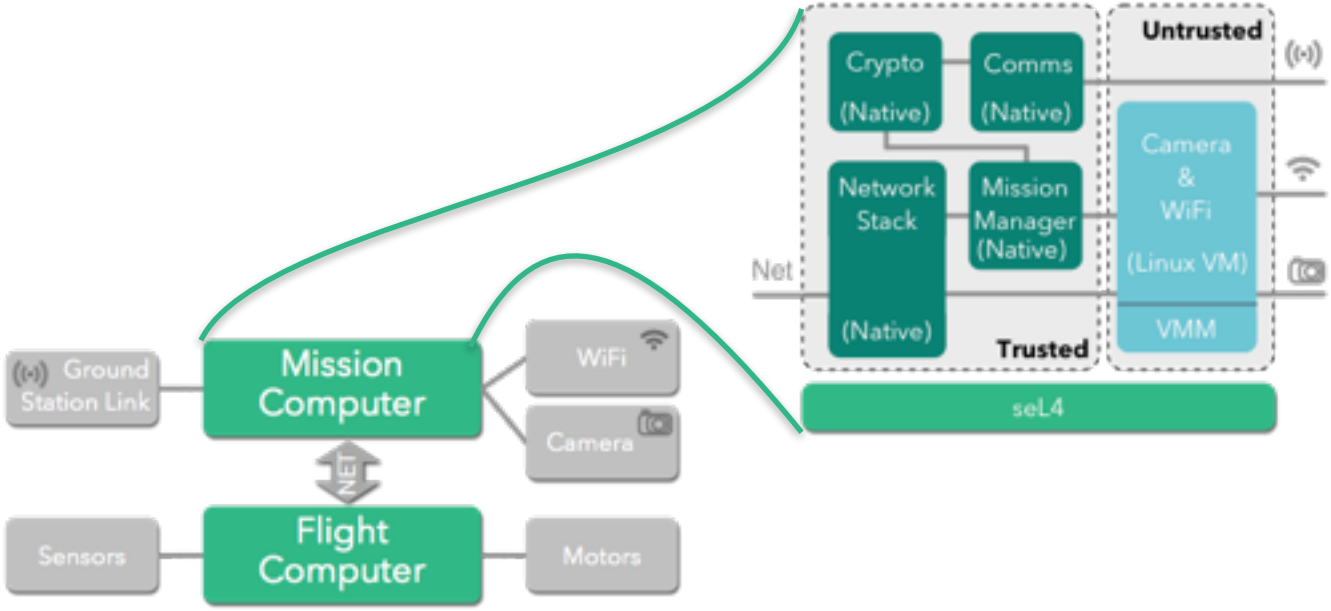
All components run isolated in a container, only the VM is still vulnerable

Deployment: in DARPA HACMS project

➔ ***Red-Team*** *could take control over the camera and Linux VM.*

➔ *Red-team could* ***not*** *send malicious commands to flight computer.*





seL4 inside!

Security. Performance. Proof.

Deployment - beyond

HACMS

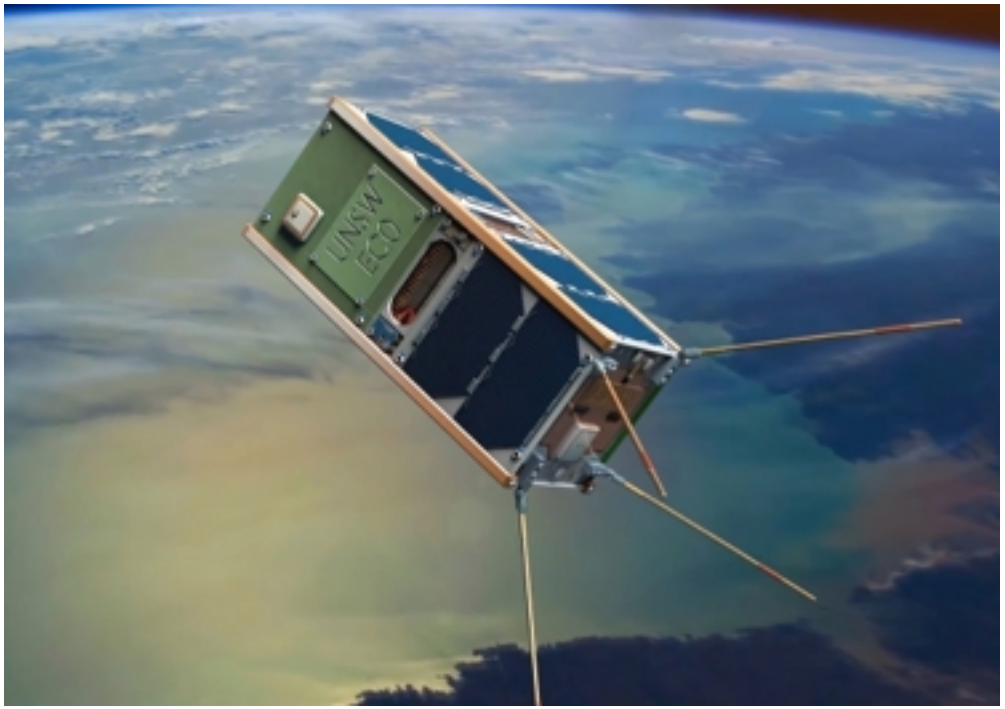



Deployment - beyond

HACMS	QB50







UNSW-ECO CUBESAT WAS LAUNCHED FROM CAPE CANAVERAL ON APRIL 19TH 2017 AT APPROXIMATELY 1:11AM SYDNEY TIME

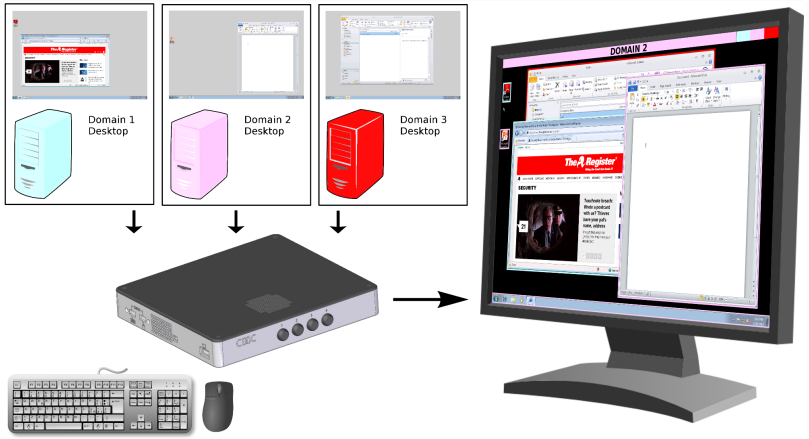
Deployment - beyond

HACMS	QB50	CDDC






CDDC: Cross-Domain Desktop Compositor



3 state iAwards!
2 national iAwards!

Deployment - beyond

HACMS	QB50	CDDC





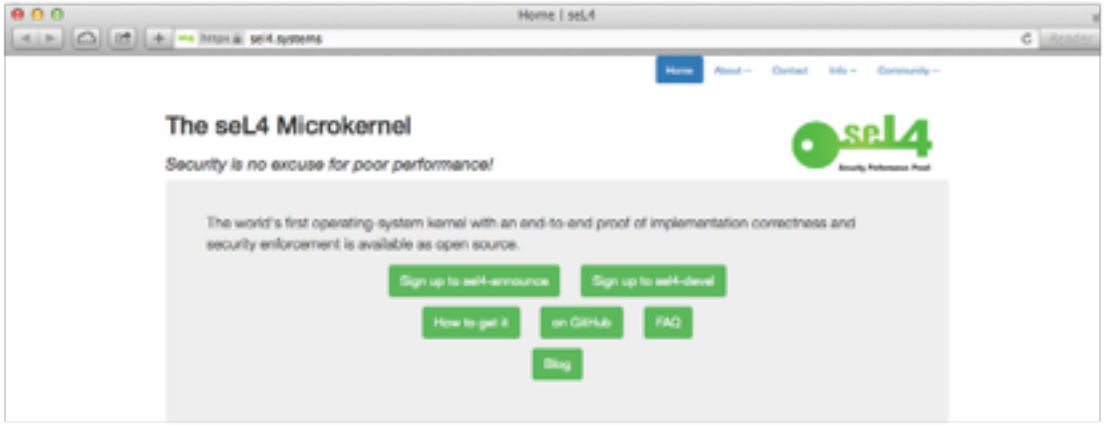

OPEN SOURCE!

https://github.com/sel4

Mailing list, IRC

Website, Wiki, Blog

Developer days



Deployment - beyond

HACMS	QB50	CDDC






OPEN SOURCE!

- https://github.com/sel4
- Mailing list, IRC
- Website, Wiki, Blog
- Developer days

SBIR+community

- helmets
- satellites
- submarines
- wireless storage
- communications dongle
- …

Deployment - beyond

HACMS	QB50	CDDC






OPEN SOURCE!

- https://github.com/sel4
- Mailing list, IRC
- Website, Wiki, Blog
- Developer days

SBIR+community

- helmets
- satellites
- submarines
- wireless storage
- communications dongle
- …

interest from

- IoT
- Automotive
- Defence
- …

Take away

Making verified software a
reality
in real-world systems

Approach:

➔ minimal & verified TCB

➔ ecosystem: seL4&co

Deployment

➔ projects

➔ community!

Repeat?






Overview

Making verified software a **reality** in real-world systems

Remaining challenges to **mainstream** verified software

Approach:

cheaper ➔ proofs for free

relevant ➔ more features

scale ➔ proof engineering

Deployment






Now/Next

Cheaper

More applications

seL4 lesson:

- High assurance (Common Criteria EAL 6+):
  $1,000/LOC, model verification + testing, unoptimised
- Low assurance (traditional embedded kernels):
  $100–200/LOC, 1–5 faults/kLOC, optimised
- State of the Art — seL4:
  $400/LOC, binary-level formal proof, optimised

➔ Verified software getting close to traditional kernel development
➔ Still too expensive for large-scale user code development

➔ automation, verified generation, proof for free

Now/Next

Cheaper

More applications

high-level languages

Cogent CakeML, mVM

Cogent

✓ write in **high-level** restricted language

✓ **automatically** generate executable**+proof for free**

✓ works for **file systems**

➡ **usability**

➡ **different** application area (network stacks)

Now/Next

Cheaper

More applications

high-level languages

Cogent, CakeML, mVM

CakeML

✓ verified optimising compiler

✓ verified **binaries** from theorem provers

➡ increase **performance**

➡ connect to **kernel proofs**

Now/Next

Cheaper

More applications

high-level languages

Cogent, CakeML, mVM

microVM

- ➡ aim at minimal VM for **managed language**
- ➡ with **high-performance**
- ➡ amenable to **verification**

Now/Next

Relevant

More features/guarantees
Real-time, multicore
Side-channels, WCET

Cheaper

More applications
high-level languages
Cogent, CakeML

Temporal isolation — ***the Holy Grail of operating systems***

Integrity: ensuring timeliness

✓ high-assurance **WCET** analysis, with **verified** loop bounds
✓ world's first real-time OS suitable for **mixed-criticality**

Confidentiality: preventing timing channels

✓ first OS that can **prevent** info leakage through timing channels
➡ evaluation to show **effectiveness** and **minimal overhead**
✓ timing-channel analysis (Spectre & Meltdown, Yuval Yarom)

Now/Next

Relevant

More features/guarantees
- Real-time, multicore +verification!
- Side-channels, WCET

Cheaper

More applications
- high-level languages
  - Cogent, CakeML

multicore seL4

✓ **big-lock** design with **low performance impact** for small number of cores

➡ **optimisation** with execution outside the lock

✓ **high-level** model of interleaving

✓ verification framework for **concurrent C code**

➡ **refinement** to implementation, preserving most of existing proof

Now/Next

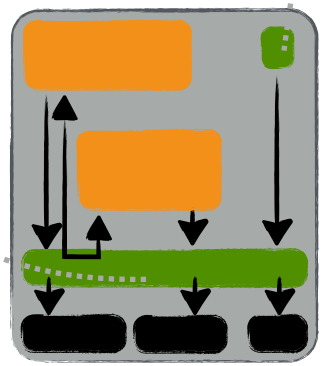
Relevant

More features/guarantees
- Real-time, multicore +verification!
- Side-channels, WCET

More usability
- platform support,
- platforms ports
  - +verification!

Cheaper

More applications
- high-level languages
  - Cogent, CakeML, mVM



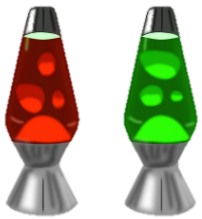
Scalable

Proof engineering!
- proof platform,
- proof development,
- proof maintenance

Now/Next

Proof engineering

Missions: guardians of **>500,000** lines of proofs;
verification of **new features, new ports, new properties**



Achieved: support for hardware **hypervisor extensions** on ARM,
verification of large number of kernel **patches**,
tools for **automation verification**, tactical language, regression testing, …

Now/Next: x64 correctness proof,
real-time seL4 verification, multicore seL4 verification,
security proofs for all of the above, …

Wishlist: **generic abstractions** for easier instantiation to new platforms,
increased verification automation and decreased refactoring effort,….

Proof engineering!

proof platform,
proof development,
proof maintenance

Now/Next

https://sel4.systems/Info/Roadmap/

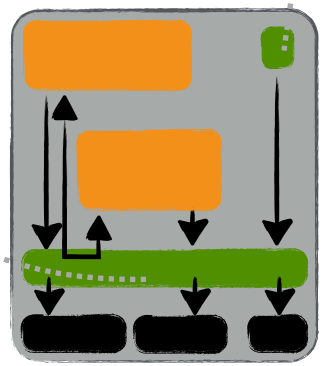
Relevant

More features/guarantees
- Real-time, multicore +verification!
- Side-channels, WCET

More usability
- platform support,
- platforms ports
  - +verification!

Cheaper

More applications
- high-level languages
  - Cogent, CakeML, mVM



Scalable

Proof engineering!
- proof platform,
- proof development,
- proof maintenance

Take-away message

We have produced **verified technology**
that is **high-performance**
and is now **in use** in various systems in the world

We aim to **radically change** the way the world builds secure systems,
and **mainstream** verified software
by increasing **automation**, **proof engineering**, **community support**

VERIFIED *and* FAST *and* CHEAP *and* DEPLOYED

TS @ Data61

https://trustworthy.systems/