Software-Enforced Isolation

The Key to Cyber-Secure Cars

Gernot Heiser | gernot.heiser@data61.csiro.au | @GernotHeiser
Trustworthy Systems | Data61

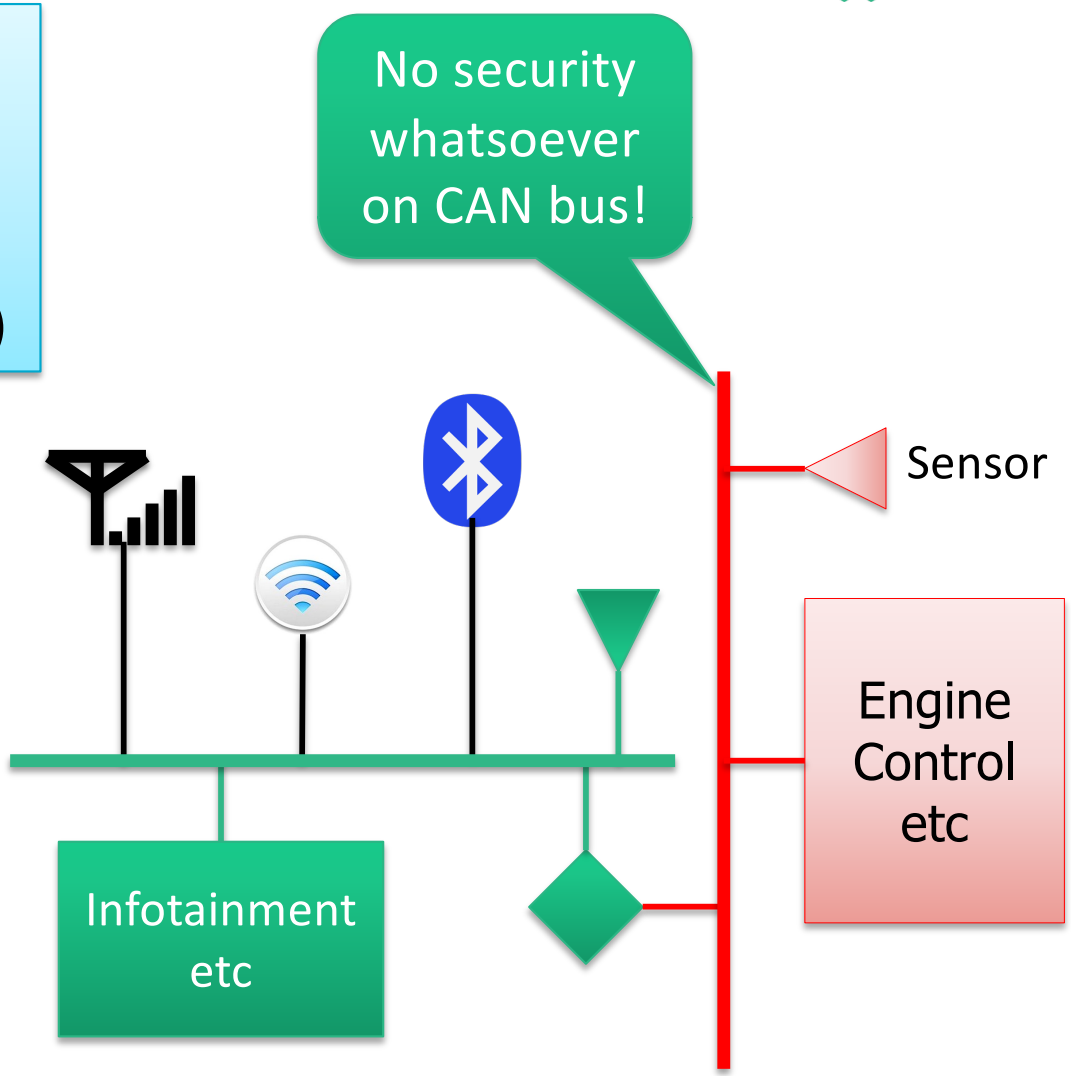
September 2017

https://trustworthy.systems

Car Hacking – What's Behind?

Networking for:

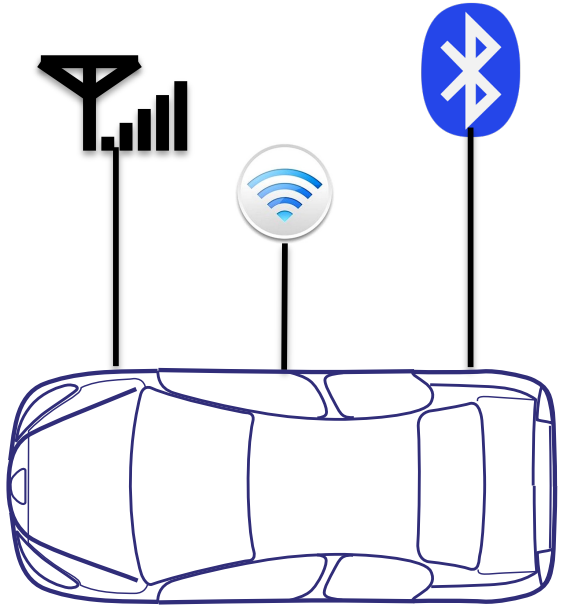
- Entertainment
- Connected car
- Safety (tire pressure…)
- Maintenance (OTA upgrades)



Challenge of Networking

Networking creates remote attack opportunities

- **from passengers (wifi, Bluetooth)**
- **from nearby cars (wifi, Bluetooth) – incl infected ones!**
- **from anywhere (cellular)**

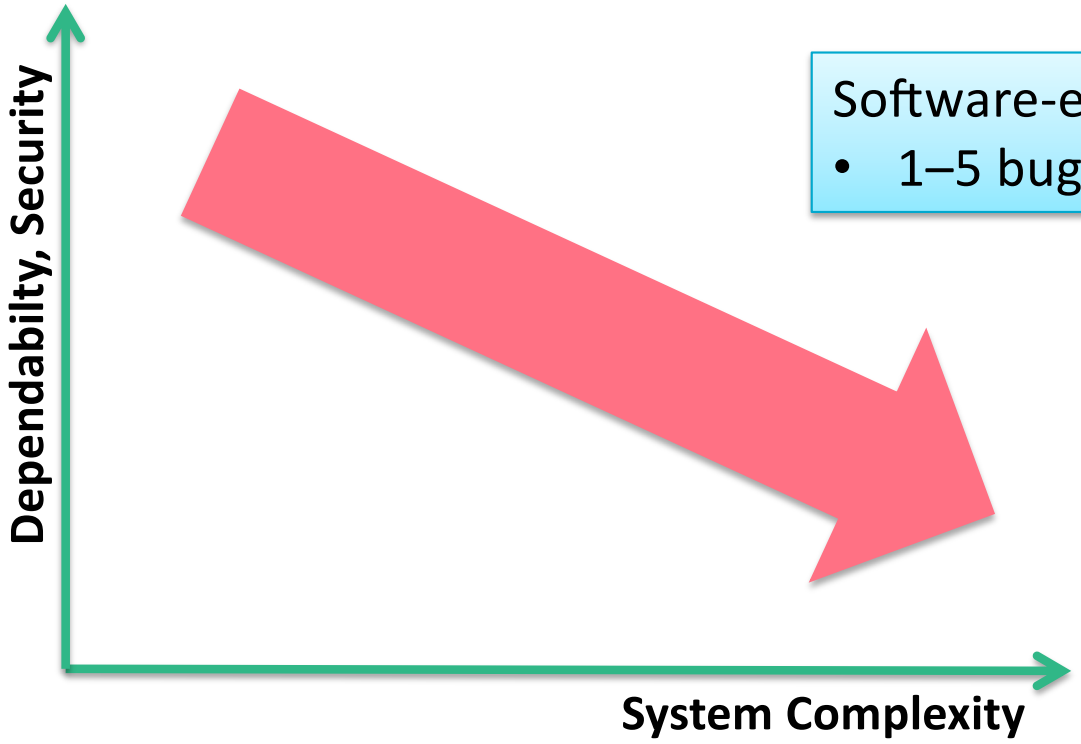


BlueBorne

Attack vectors:

- **Insecure protocols**
- **Reusing crypto keys**
- **Software vulnerabilities**

Software Vulnerabilities



Software-engineering rule of thumb:
- 1–5 bugs per 1,000 lines of ***quality*** code

Bluetooth protocol stack:
Multiple 100,000 lines

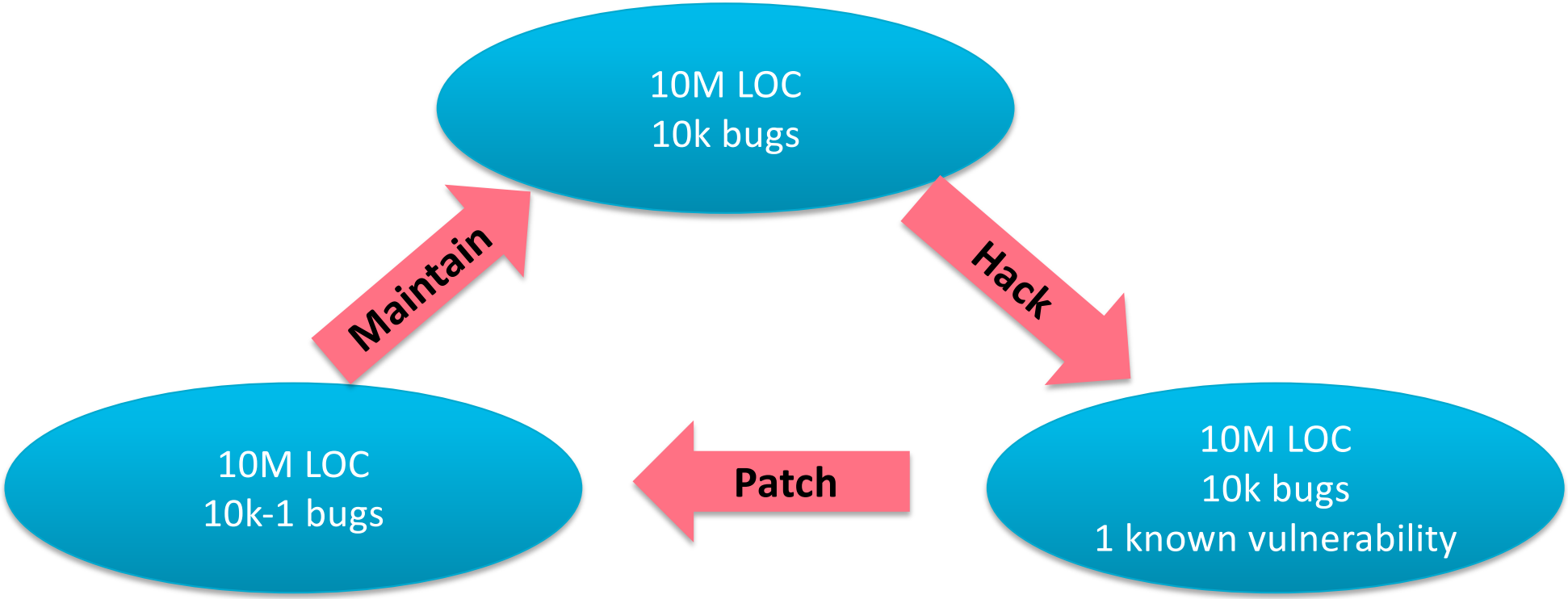
Linux kernel:
Tens of millions lines

Complexity Drivers

- Features/functionality
- Legacy reuse

Linux “Security”

OK, So Let's Patch Regularly

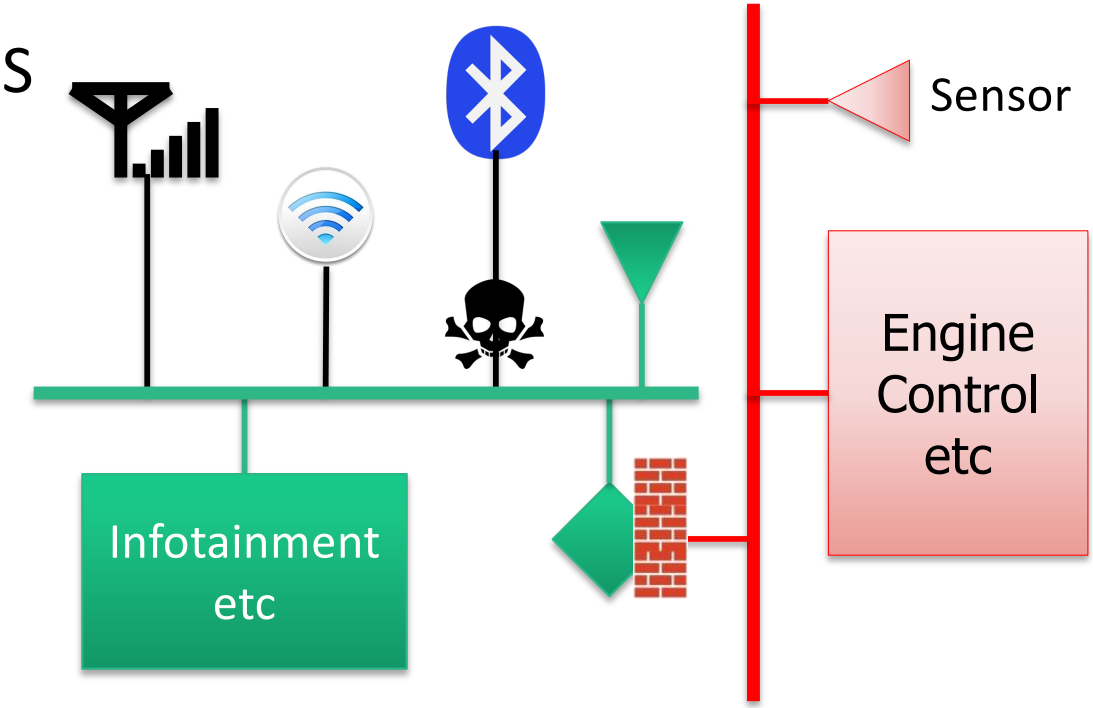


Patch-and-Pray: A losing proposition

So, Let's Use Firewalls!

- Imposes overhead (SWaP)
- Even more code –
  may *increase* attack surface
- No help for valid messages
  that trigger bugs in software
- Firewall runs on vulnerable OS

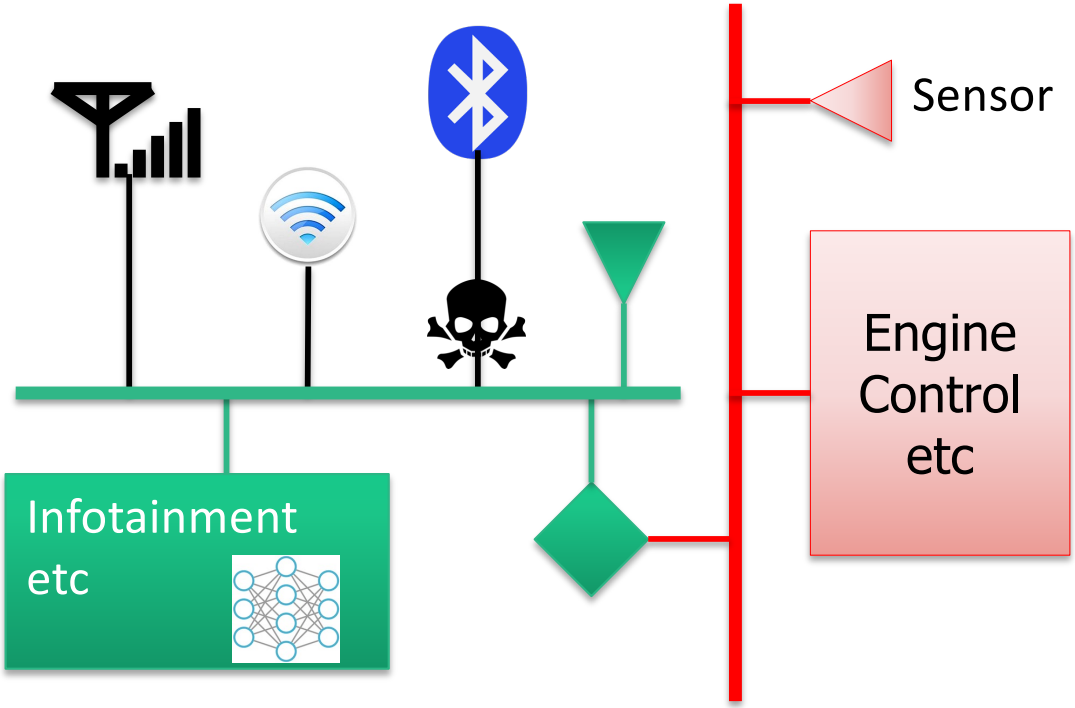
Firewalls treat symptoms, not causes of problems!



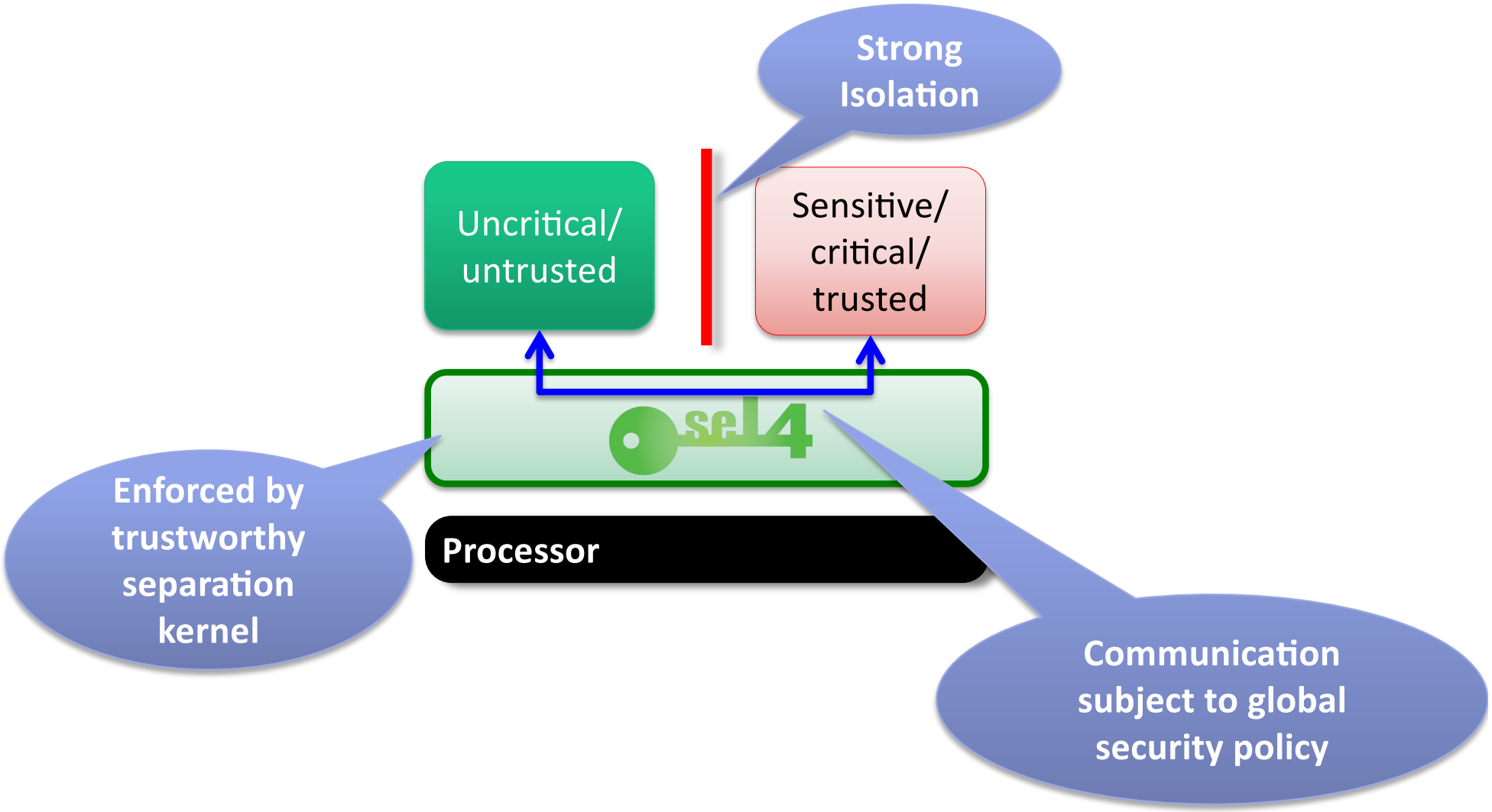
Let's Use AI to Detect Compromise!

- Can only detect that system is already compromised
- Even more code –
  may *increase* attack surface
- Runs on compromised OS!

**Intrusion detection –
admission of defeat**



Fundamental Security Requirement: Isolation



Trustworthiness: Can We Rely on Isolation?

A system is **trustworthy** if and only if:

- it behaves **exactly** as it is specified,
- in a **timely** manner, and
- while ensuring **secure** execution

***Claim*:**

A system must be considered *untrustworthy* unless *proved* otherwise!

Corollary [with apologies to Dijkstra]:

Testing, code inspection, etc. can only show *lack of trustworthiness*!

Provably Secure Operating System

~10,000 lines of C and ASM code

Small attack surface,
Amenable to full verification

Non-kernel code can only access resources and communication channels if explicitly authorised with a per-object access token (capability)

Kernel can confine damage from attacks in unprivileged code

Small,

fast,

capability-based,

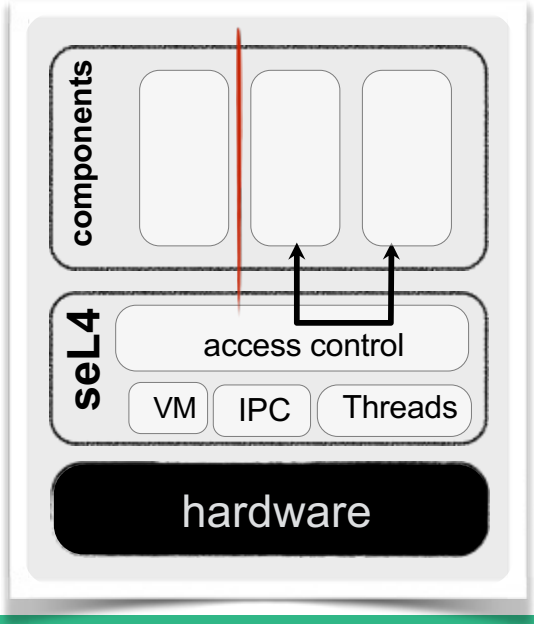
operating system kernel

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

Suitable for real-world deployment

Code that runs in privileged mode of the hardware

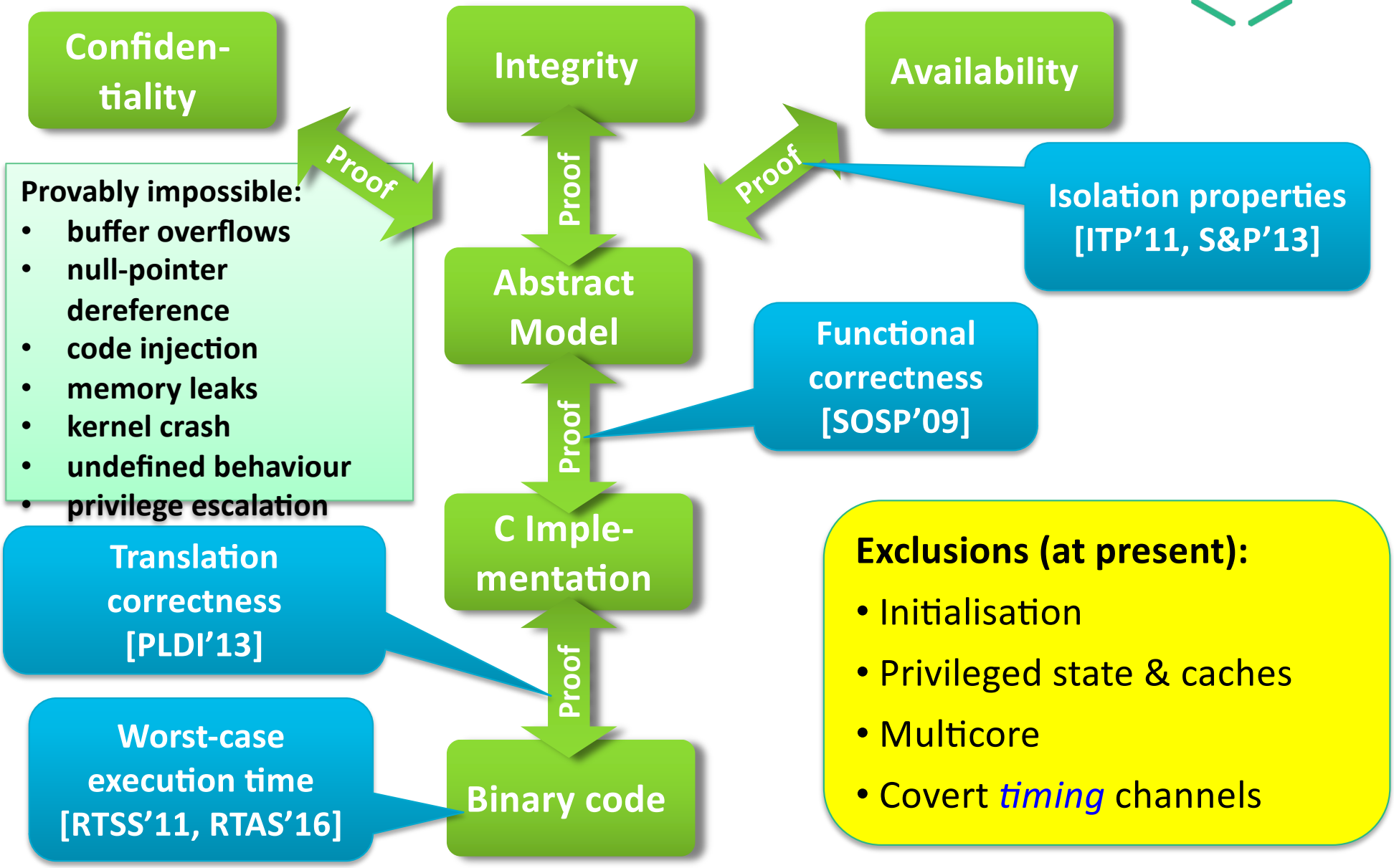
Most critical part



Unprivileged mode

Privileged mode

Proving Trustworthiness of seL4



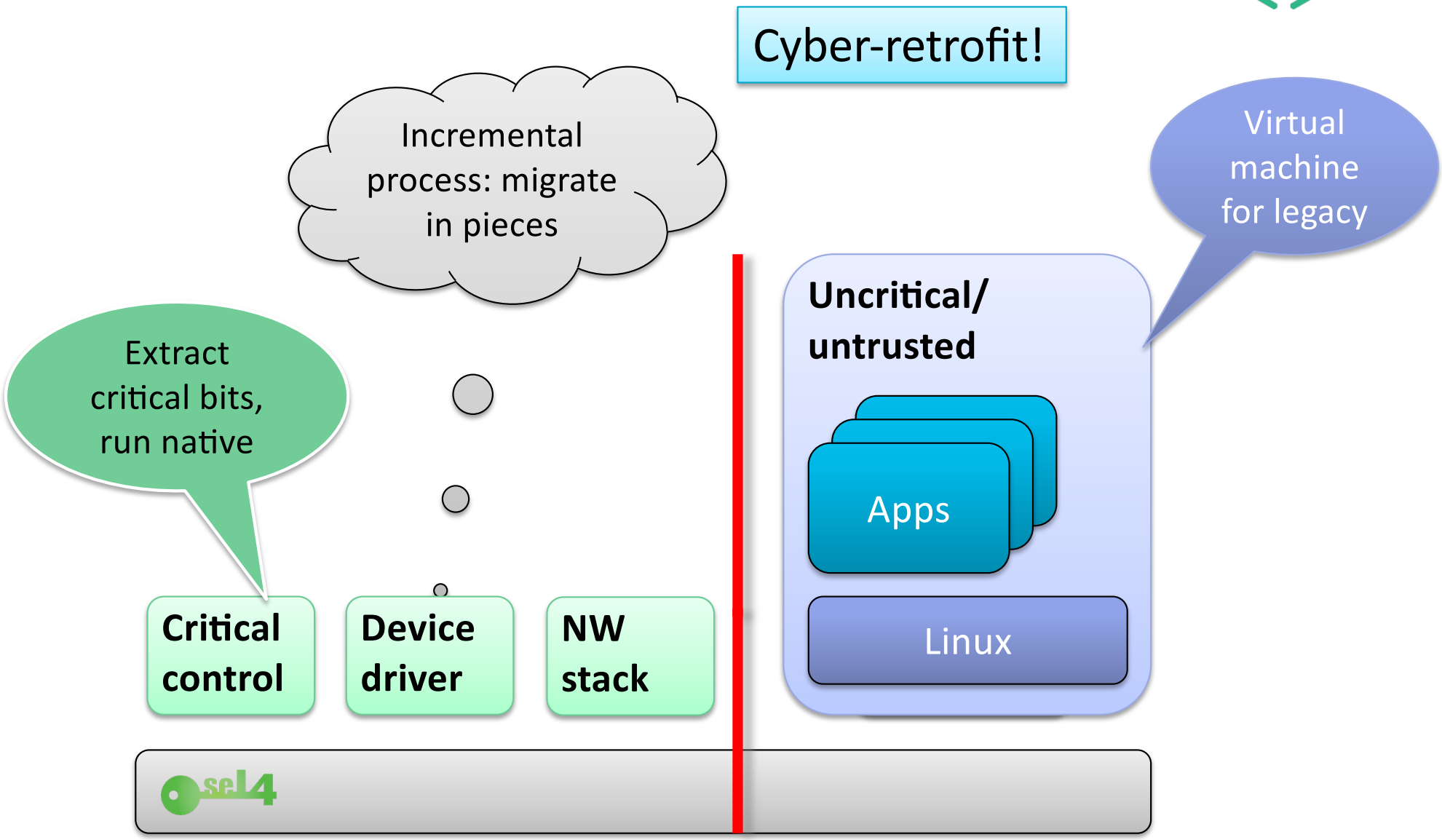
How Does seL4 Compare?

"World's most verified kernel"

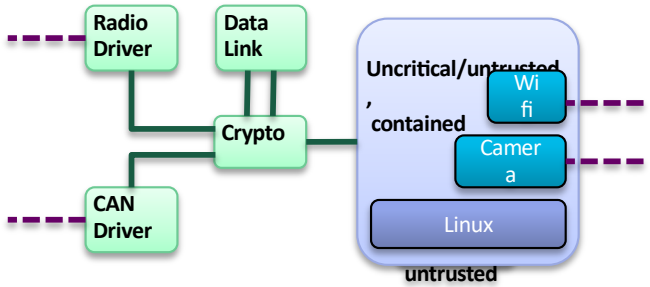
"Software you can depend on, data access you can trust"

Feature	**seL4**	**Others** (RTOSes, hypervisors, separation kernels)
Performance	Fast	5-10X slower
Functional Correctness	Guaranteed (Proved)	No Guarantee
Isolation	Guaranteed (Proved)	No Guarantee
Worst-case latency bounds	Sound and Complete	Estimates only
Storage Side Channel Freedom	Guaranteed (Proved)	No Guarantee
Timing Channel Prevention	Low overhead	None or High Overhead
Mixed Criticality Support	Fully supported, High Utilisation	Limited, resource-wastive

Security by Architecture

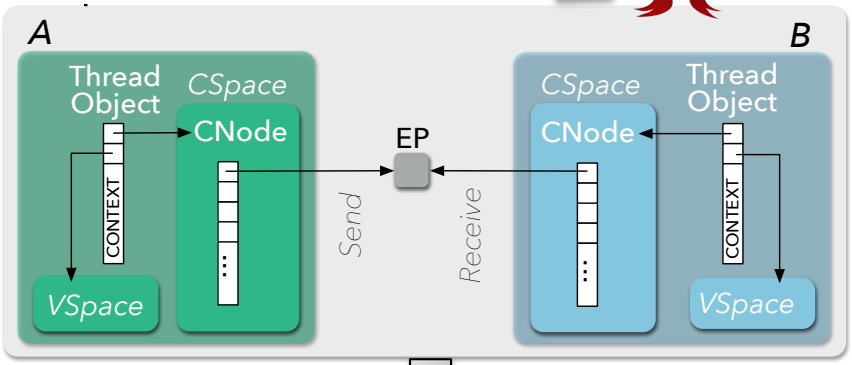


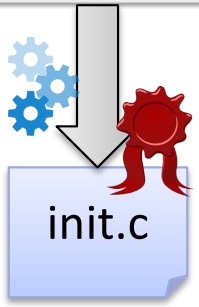
Enforcing the Architecture

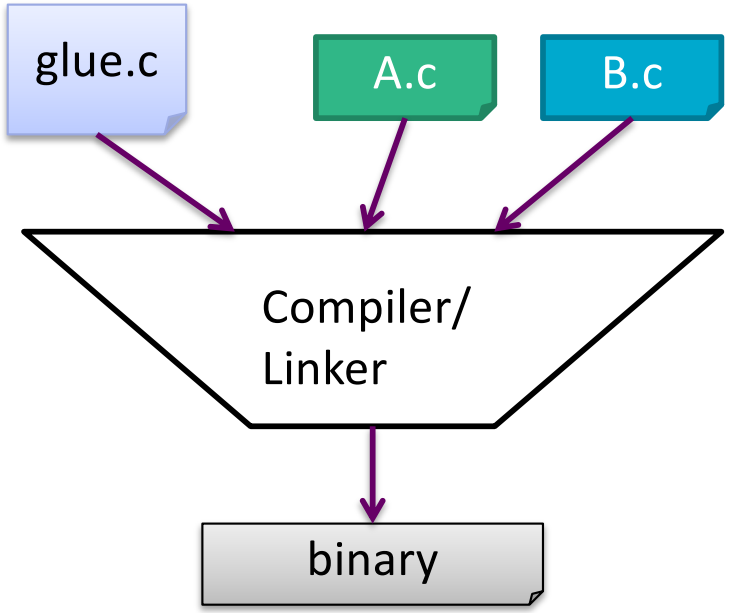


Architecture specification language

Low-level access rights







Real-World Use: DARPA HACMS

Boeing Unmanned Little Bird

Retrofit existing system!

US Army Autonomous Trucks

SMACCMcopter Research Vehicle

Develop technology

TARDEC GVR-Bot


DATA
61
Thank you
Security is no excuse for poor performance!
Gernot Heiser | gernot.heiser@data61.csiro.au | @GernotHeiser
APril 2017
http://sel4.systems
seL4
CSIRO