NATIONAL ICT AUSTRALIA LIMITED

SECURE EMBEDDED SYSTEMS NEED MICROKERNELS

Gernot Heiser
Embedded, Real-Time and Operating Systems Program
National ICT Australia

February 2004

Australian Government
Department of Communications, Information Technology and the Arts
Australian Research Council

NICTA Members
THE AUSTRALIAN NATIONAL UNIVERSITY
UNSW
First for Business
Department of State and Regional Development
BusinessACT ACT GOVERNMENT
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
NICTA Partners

EMBEDDED SYSTEM ARE UBIQUITOUS

NATIONAL ICT AUSTRALIA LIMITED

EMBEDDED SYSTEM ARE UBIQUITOUS








But are they *Secure?*

SECURITY CHALLENGES

NATIONAL ICT AUSTRALIA LIMITED

- Growing functionality
- Wireless connectivity
- Downloaded contents (entertainment)

Security Challenges

NATIONAL ICT AUSTRALIA LIMITED

- Growing functionality
  - ➜ increasing software complexity
  - ➜ increased number of faults
  - ➜ increased likelihood of security faults
- Wireless connectivity
- Downloaded contents (entertainment)

Security Challenges

NATIONAL ICT AUSTRALIA LIMITED

- Growing functionality
  - ➜ increasing software complexity
  - ➜ increased number of faults
  - ➜ increased likelihood of security faults
- Wireless connectivity
  - ➜ subject to attacks from outside (crackers)
- Downloaded contents (entertainment)

Security Challenges

NATIONAL ICT AUSTRALIA LIMITED

- Growing functionality
  - ➜ increasing software complexity
  - ➜ increased number of faults
  - ➜ increased likelihood of security faults
- Wireless connectivity
  - ➜ subject to attacks from outside (crackers)
- Downloaded contents (entertainment)
  - ➜ subject to attacks from inside (viruses, worms)

Security Challenges

- Growing functionality
  - ➜ increasing software complexity
  - ➜ increased number of faults
  - ➜ increased likelihood of security faults
- Wireless connectivity
  - ➜ subject to attacks from outside (crackers)
- Downloaded contents (entertainment)
  - ➜ subject to attacks from inside (viruses, worms)
- Increasing dependence on embedded systems
  - ➜ increased exposure to embedded-systems security weaknesses

Present Approaches

- Real-time executives

Present Approaches

- Real-time executives
  - ⋆ suitable for systems with very limited functionality
  - ⋆ no internal protection

Present Approaches

- Real-time executives
  - ⋆ suitable for systems with very limited functionality
  - ⋆ no internal protection
    - ➜ every small bug/failure is fatal
    - ➜ no defence against viruses, limited defence against crackers

Present Approaches

- Real-time executives
  - ⋆ suitable for systems with very limited functionality
  - ⋆ no internal protection
    - ➜ every small bug/failure is fatal
    - ➜ no defence against viruses, limited defence against crackers
- Linux, Windows Embedded, ...

Present Approaches

- Real-time executives
  - ⋆ suitable for systems with very limited functionality
  - ⋆ no internal protection
    - ➜ every small bug/failure is fatal
    - ➜ no defence against viruses, limited defence against crackers
- Linux, Windows Embedded, ...
  - ⋆ dubious or non-existent real-time capabilities
    - ➜ unsuitable for hard real-time systems

Present Approaches

- Real-time executives
  - ⋆ suitable for systems with very limited functionality
  - ⋆ no internal protection
    - ➜ every small bug/failure is fatal
    - ➜ no defence against viruses, limited defence against crackers
- Linux, Windows Embedded, ...
  - ⋆ dubious or non-existent real-time capabilities
    - ➜ unsuitable for hard real-time systems
  - ⋆ huge code base (millions of lines of code)
    - ➜ excessive for small embedded system
    - ➜ too much code on which security of system is dependent

NATIONAL ICT AUSTRALIA LIMITED

REQUIRES MINIMAL TRUSTED COMPUTING BASE (TCB)

TCB: The part of system that must *be relied on for the correct operation of the system*

NATIONAL ICT AUSTRALIA LIMITED

REQUIRES MINIMAL TRUSTED COMPUTING BASE (TCB)

TCB: The part of system that must *be relied on for the correct operation of the system*

... otherwise the security issue is intractable!

Embedded-Systems Security

NATIONAL ICT AUSTRALIA LIMITED

Requires minimal trusted computing base (TCB)

TCB: The part of system that must *be relied on for the correct operation of the system*

... otherwise the security issue is intractable!

Minimal TCB means:

1. Split software into trusted and untrusted part

EMBEDDED-SYSTEMS SECURITY

NATIONAL ICT AUSTRALIA LIMITED

REQUIRES MINIMAL TRUSTED COMPUTING BASE (TCB)

TCB: The part of system that must *be relied on for the correct operation of the system*

... otherwise the security issue is intractable!

MINIMAL TCB MEANS:

1. Split software into trusted and untrusted part
2. Minimise the trusted part

EMBEDDED-SYSTEMS SECURITY

NATIONAL ICT AUSTRALIA LIMITED

REQUIRES MINIMAL TRUSTED COMPUTING BASE (TCB)

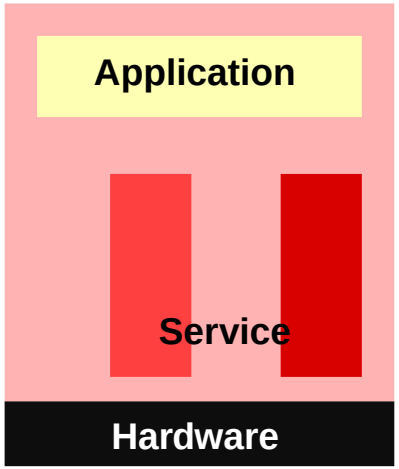
TCB: The part of system that must *be relied on for the correct operation of the system*

... otherwise the security issue is intractable!

MINIMAL TCB MEANS:

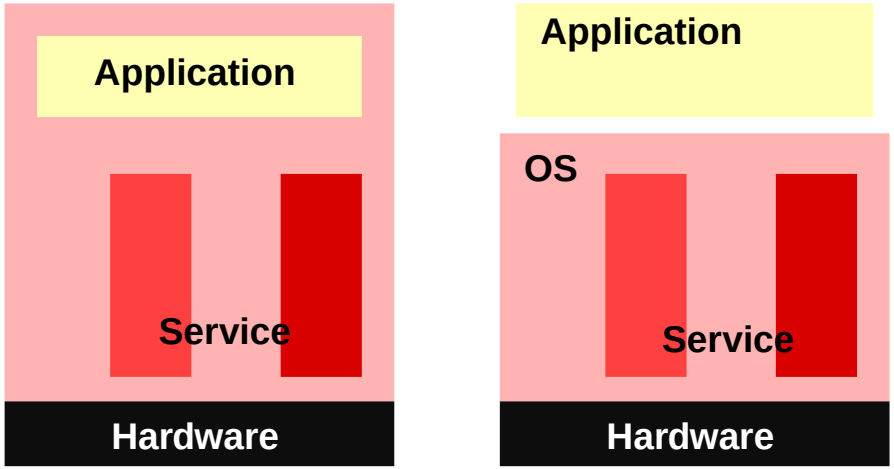
1. Split software into trusted and untrusted part
2. Minimise the trusted part
3. Make the trusted part *trustworthy*

MINIMISING THE TCB



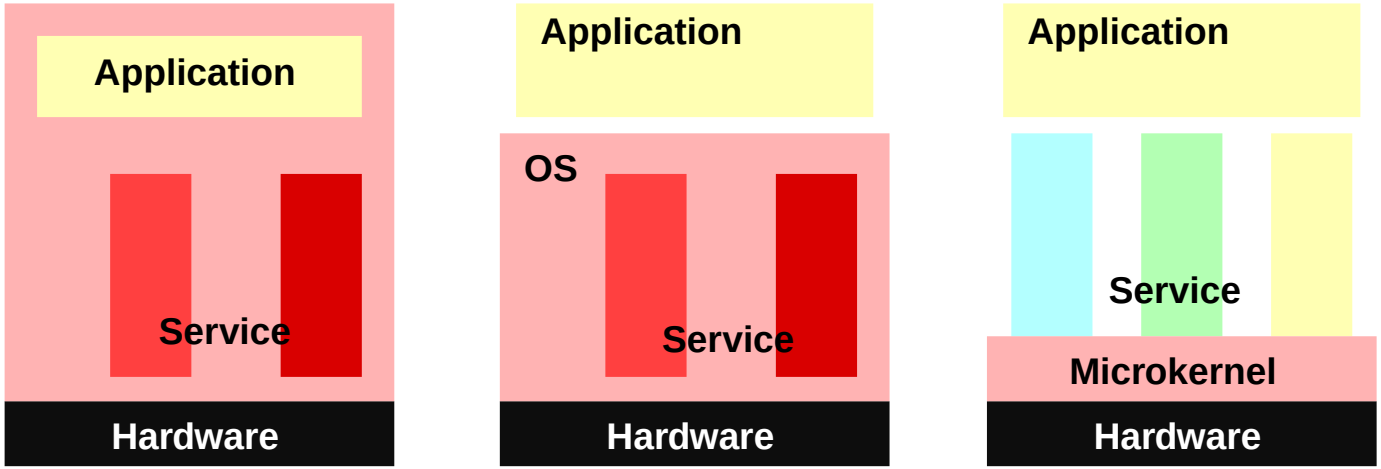
System: traditional
embedded

MINIMISING THE TCB



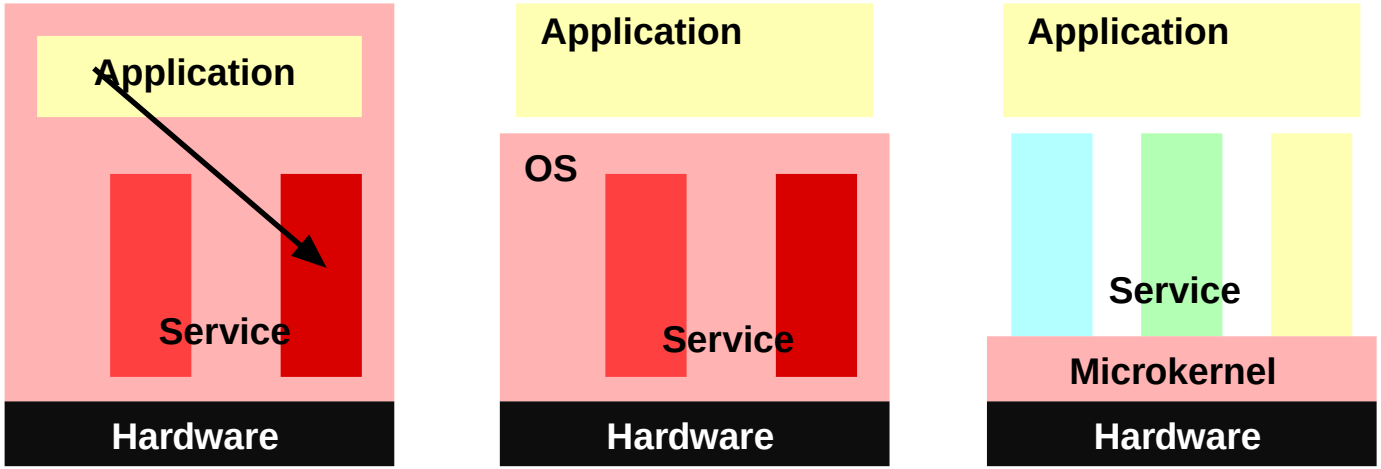
System:	traditional embedded	Linux/ Windows

MINIMISING THE TCB



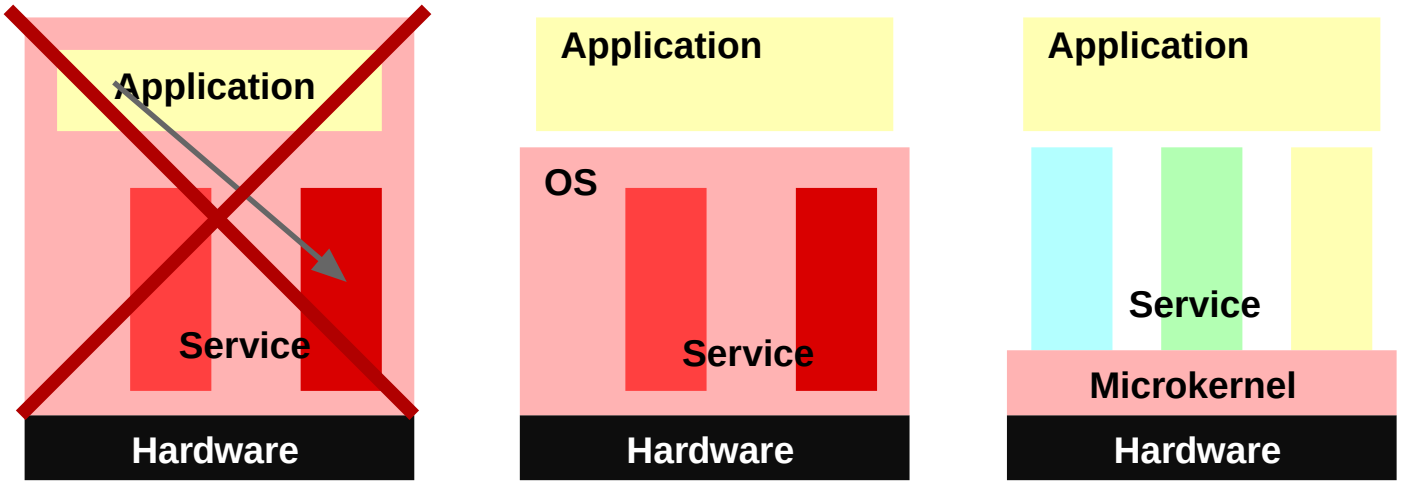
System:	traditional embedded	Linux/ Windows	Microkernel-based

MINIMISING THE TCB



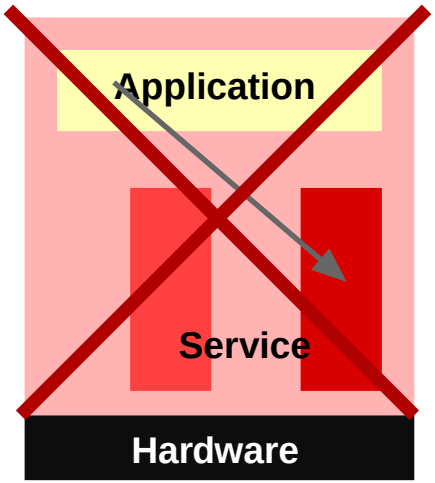
System:	traditional embedded	Linux/ Windows	Microkernel-based

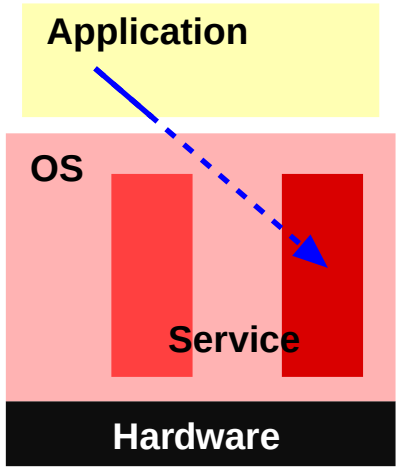
MINIMISING THE TCB

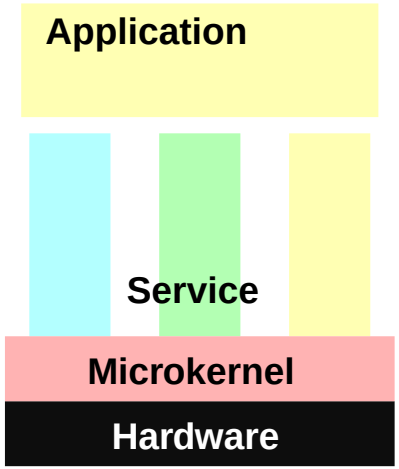


System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	**all** code		

MINIMISING THE TCB

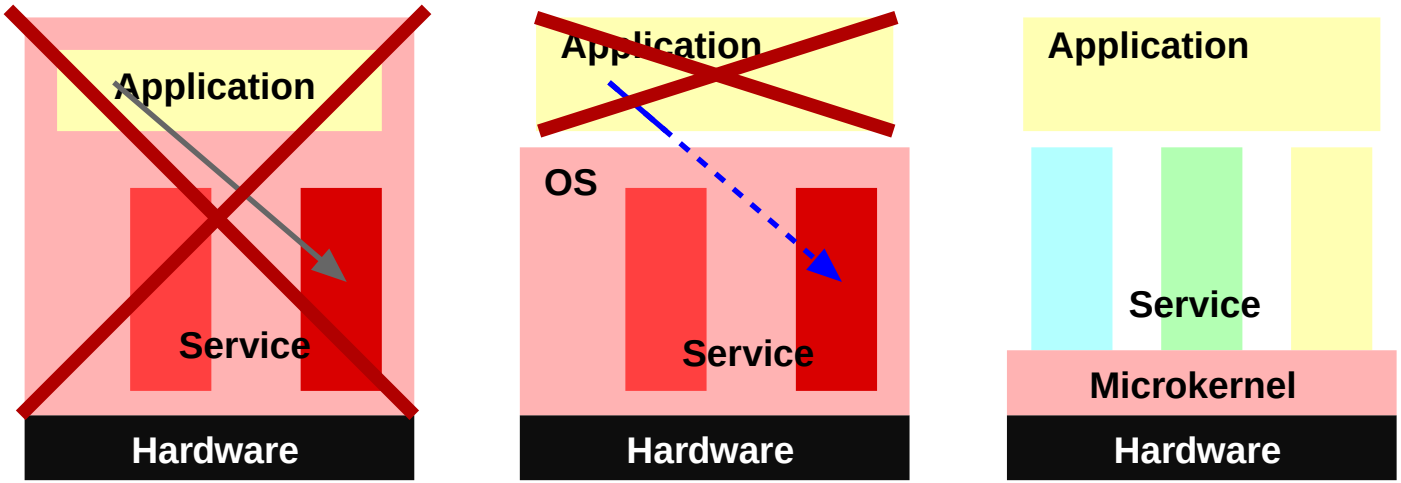






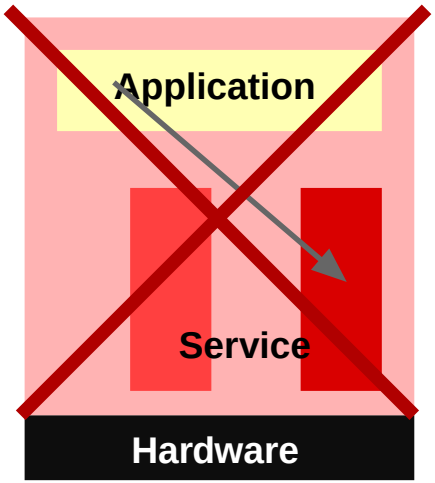
System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	**all** code		

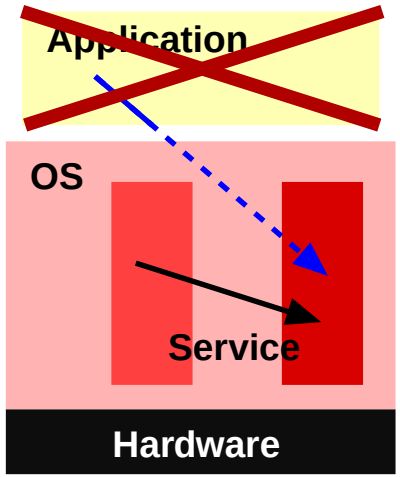
MINIMISING THE TCB

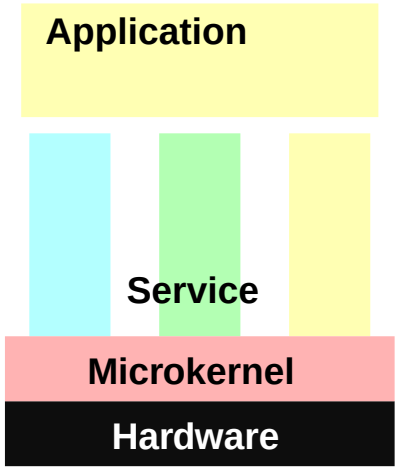


System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	**all** code		

MINIMISING THE TCB

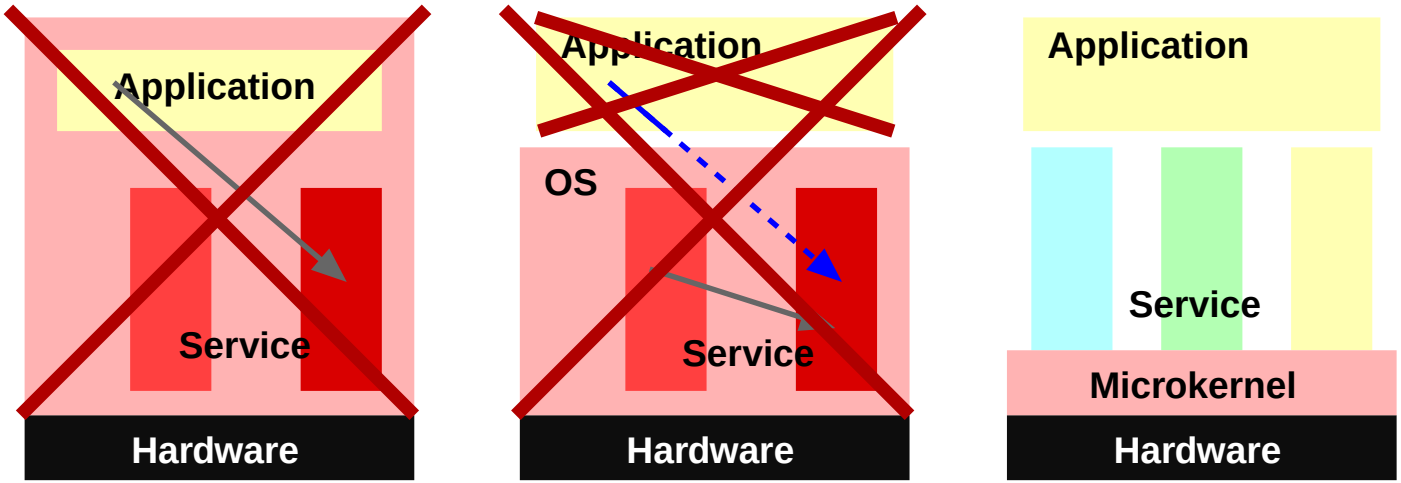






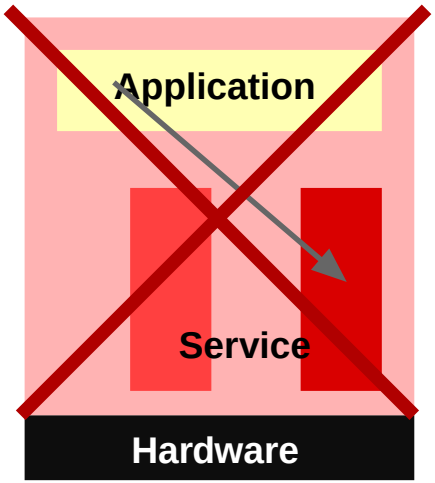
System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	**all** code		

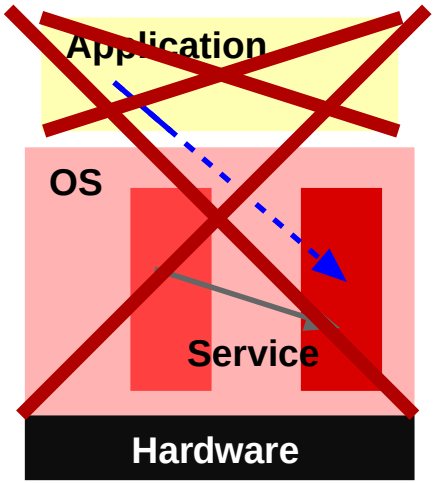
MINIMISING THE TCB

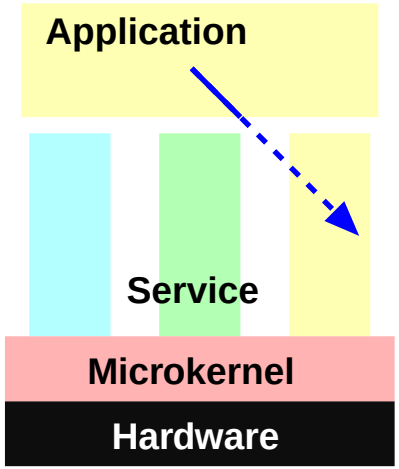


System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	**all** code	100,000's loc	

MINIMISING THE TCB

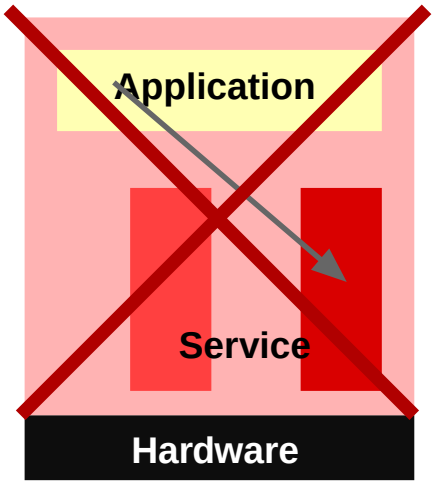


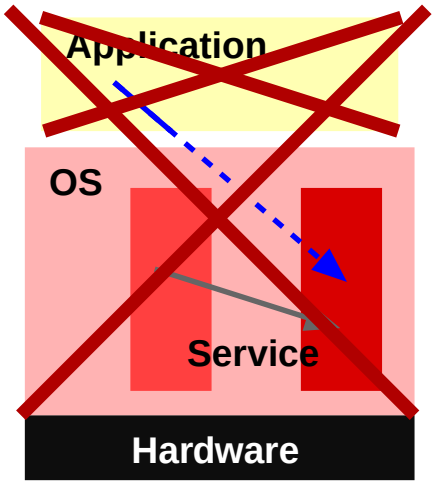


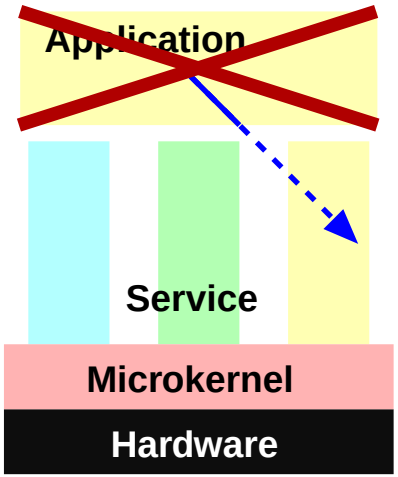


System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	**all** code	100,000's loc	

MINIMISING THE TCB

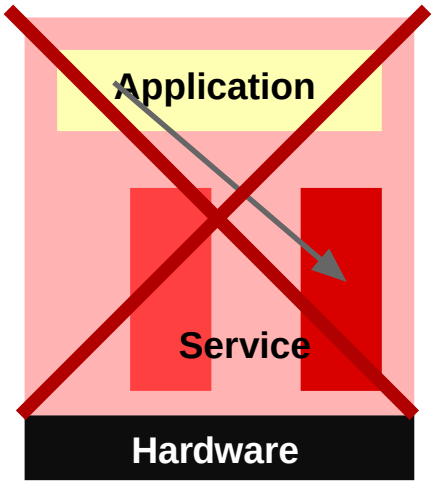


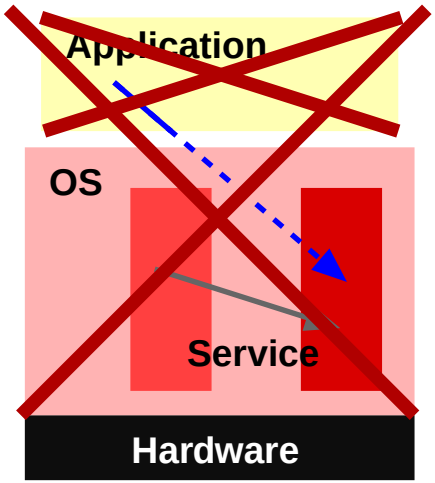


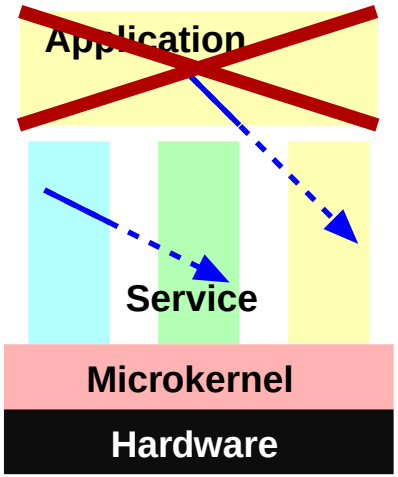


System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	**all** code	100,000's loc	

MINIMISING THE TCB

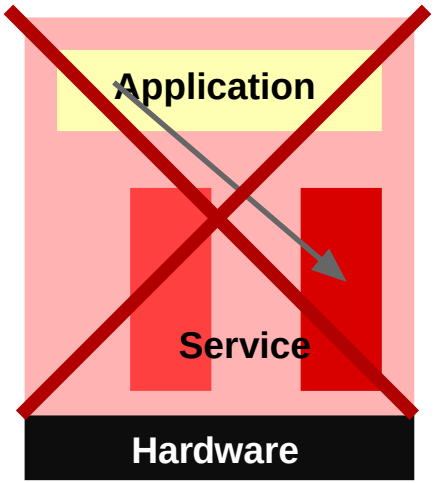


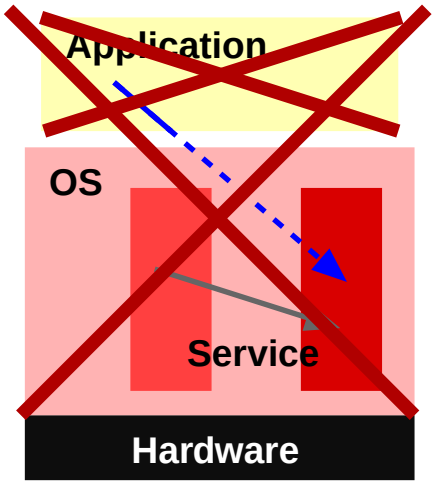


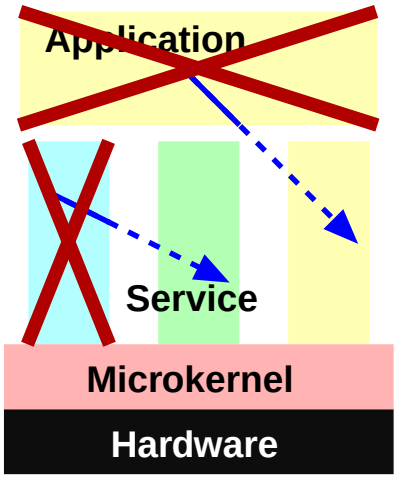


System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	**all** code	100,000's loc	

MINIMISING THE TCB

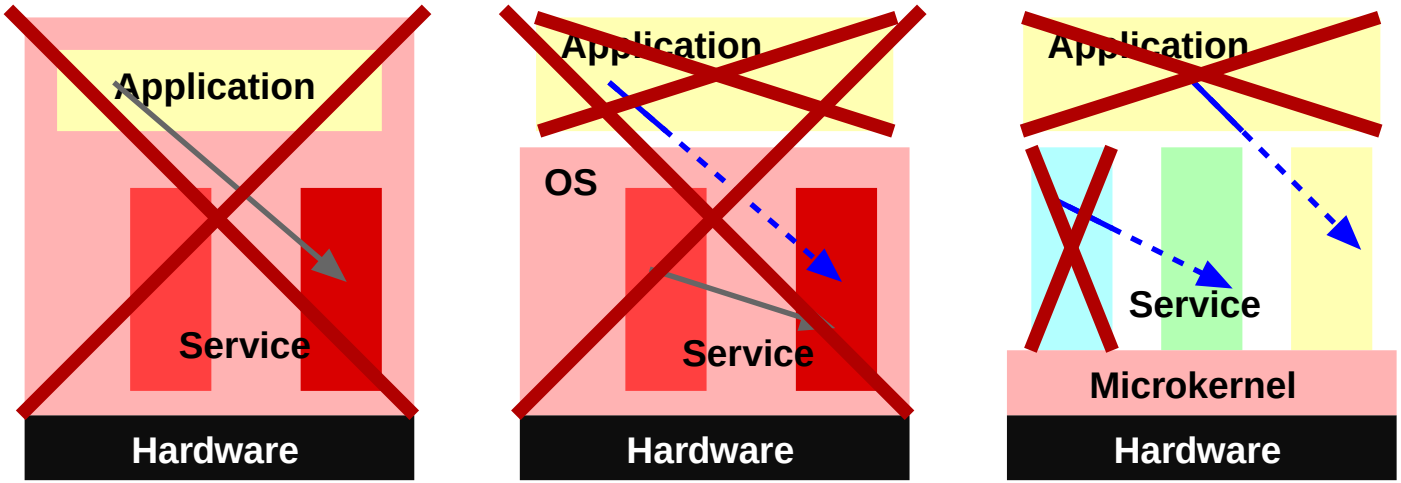






System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	**all** code	100,000's loc	10,000's loc

MINIMISING THE TCB



System:	traditional embedded	Linux/ Windows	Microkernel-based
TCB:	**all** code	100,000's loc	10,000's loc

Small kernel ⇒ small TCB ⇒ more trustworthy TCB!

Making the TCB Trustworthy

Traditional approaches

Advanced approaches

Making the TCB Trustworthy

NATIONAL ICT AUSTRALIA LIMITED

Traditional approaches

- Testing
- Code inspection

Advanced approaches

Making the TCB Trustworthy

NATIONAL ICT AUSTRALIA LIMITED

Traditional approaches

- Testing
- Code inspection

☑ Smaller TCB ⇒ fewer remaining bugs

☒ ... but no *assurance* of absence of bugs!

Advanced approaches

Making the TCB Trustworthy

NATIONAL ICT AUSTRALIA LIMITED

Traditional approaches

- Testing
- Code inspection
- ☑ Smaller TCB ⇒ fewer remaining bugs
  - ☒ ... but no *assurance* of absence of bugs!

Advanced approaches

- *Prove* the correctness of the TCB
  - ☑ assurance of absence of bugs

Making the TCB Trustworthy

Traditional approaches

- Testing
- Code inspection
- ☑ Smaller TCB ⇒ fewer remaining bugs
  - ☒ ... but no *assurance* of absence of bugs!

Advanced approaches

- *Prove* the correctness of the TCB
  - ☑ assurance of absence of bugs
  - ☒ unfeasible for 100,000s loc
  - ☑ feasible for very small microkernel

MICROKERNEL VERIFICATION

MICROKERNEL VERIFICATION

PROOF OF CORRECTNESS OF MICROKERNEL

MICROKERNEL VERIFICATION

NATIONAL ICT AUSTRALIA LIMITED

PROOF OF CORRECTNESS OF MICROKERNEL

1. Formal specification of microkernel API

   ➜ prerequisite for all formal work on kernel

MICROKERNEL VERIFICATION

NATIONAL ICT AUSTRALIA LIMITED

PROOF OF CORRECTNESS OF MICROKERNEL

1. Formal specification of microkernel API

   ➜ prerequisite for all formal work on kernel

2. Prove security (isolation) properties of API

   ➜ confinement, data flow, interference, covert storage channels

Microkernel Verification

NATIONAL ICT AUSTRALIA LIMITED

Proof of correctness of microkernel

1. Formal specification of microkernel API
   - ➜ prerequisite for all formal work on kernel
2. Prove security (isolation) properties of API
   - ➜ confinement, data flow, interference, covert storage channels
3. Prove correctness of microkernel implementation
   - ➜ *refine* API spec into compilable source code

Microkernel Verification

NATIONAL ICT AUSTRALIA LIMITED

Proof of correctness of microkernel

1. Formal specification of microkernel API
   - ➜ prerequisite for all formal work on kernel
2. Prove security (isolation) properties of API
   - ➜ confinement, data flow, interference, covert storage channels
3. Prove correctness of microkernel implementation
   - ➜ *refine* API spec into compilable source code

Complete temporal model of microkernel

MICROKERNEL VERIFICATION

NATIONAL ICT AUSTRALIA LIMITED

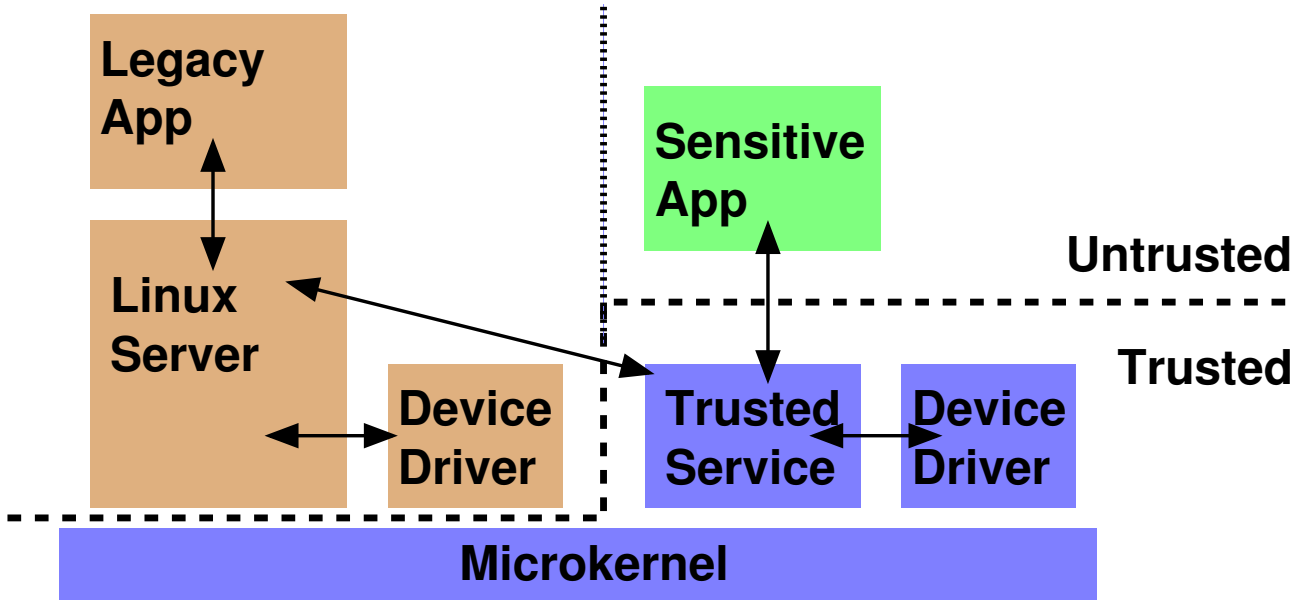
PROOF OF CORRECTNESS OF MICROKERNEL

1. Formal specification of microkernel API
   - ➜ prerequisite for all formal work on kernel
2. Prove security (isolation) properties of API
   - ➜ confinement, data flow, interference, covert storage channels
3. Prove correctness of microkernel implementation
   - ➜ *refine* API spec into compilable source code

COMPLETE TEMPORAL MODEL OF MICROKERNEL

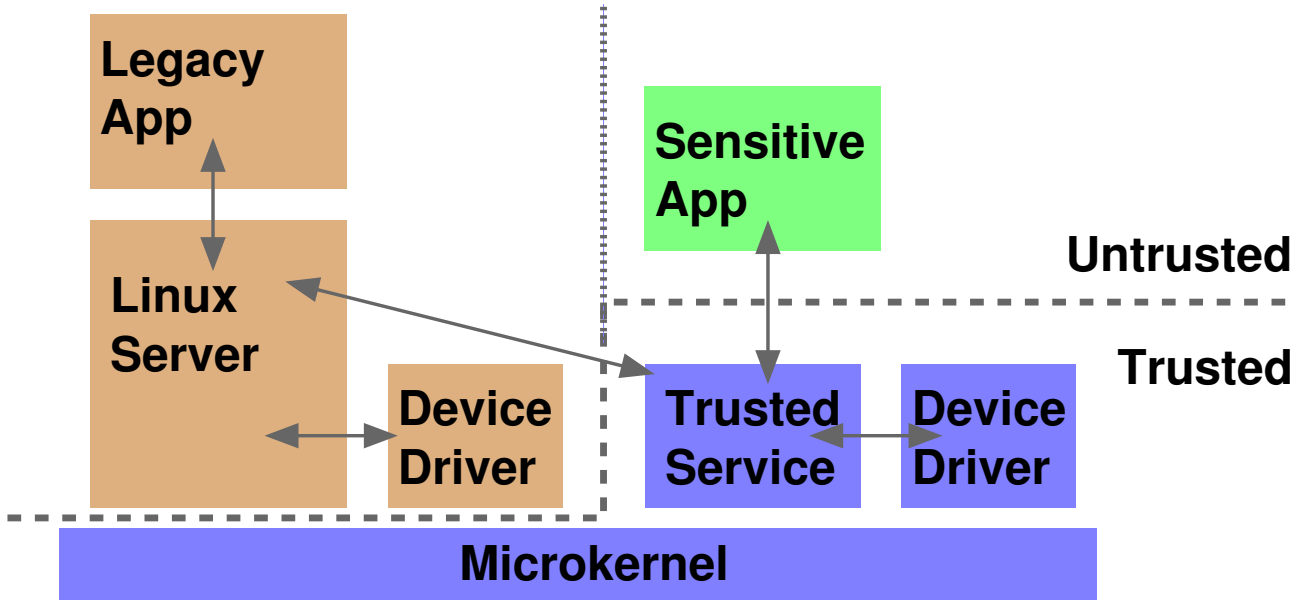
- exhaustive measurement of latencies of kernel operations
  - ➜ prerequisite for real-time analysis of whole system

A Sample System



- Sensitive part of system has small TCB

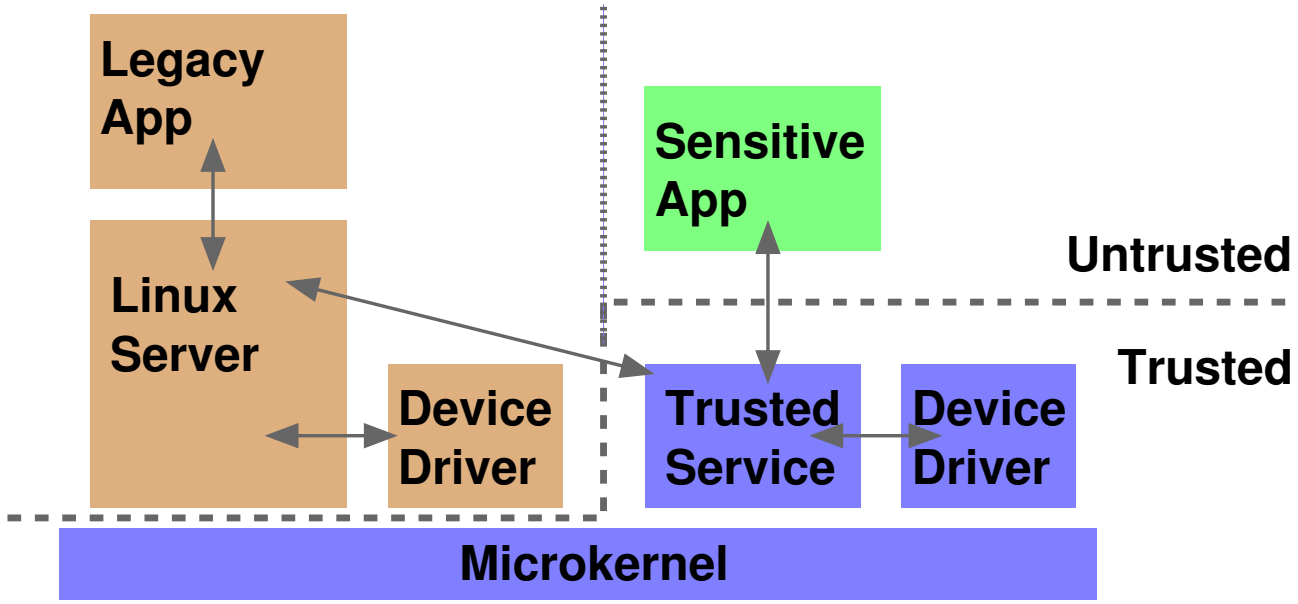
A Sample System



- Sensitive part of system has small TCB
- Standard API supported by de-privileged Linux server

➜ full binary compatibility with native Linux

A Sample System

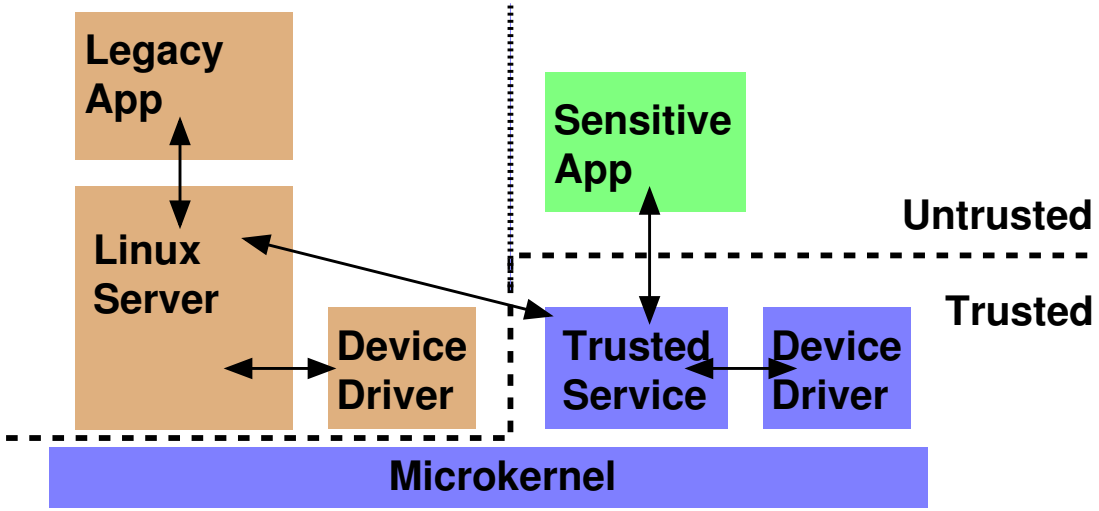


- Sensitive part of system has small TCB
- Standard API supported by de-privileged Linux server
  - ➜ full binary compatibility with native Linux
- Compromised legacy system cannot interfere with trusted part

STATUS

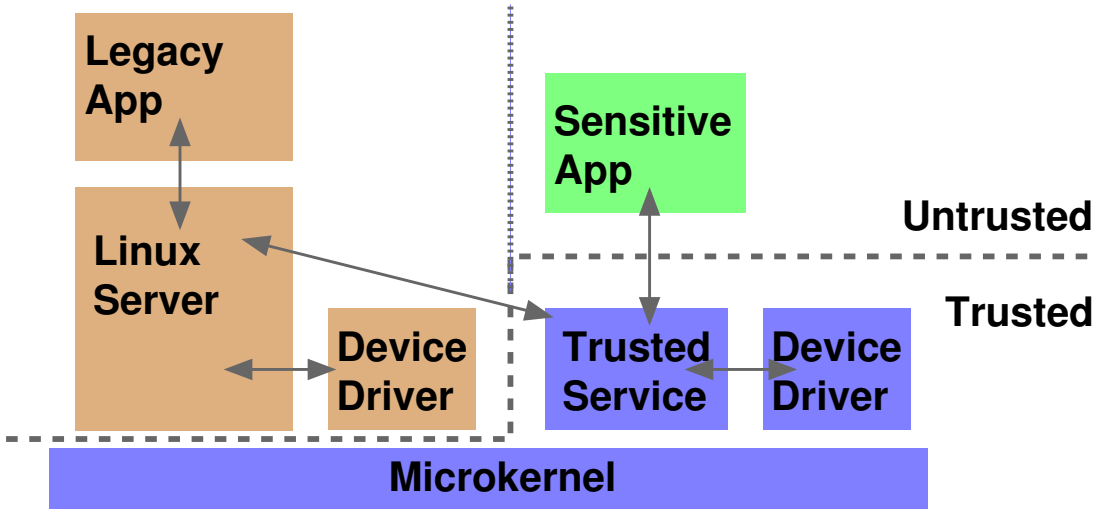
NATIONAL ICT AUSTRALIA LIMITED

- Microkernel operational
  - ➜ 10,000 lines of code
  - ➜ highly efficient



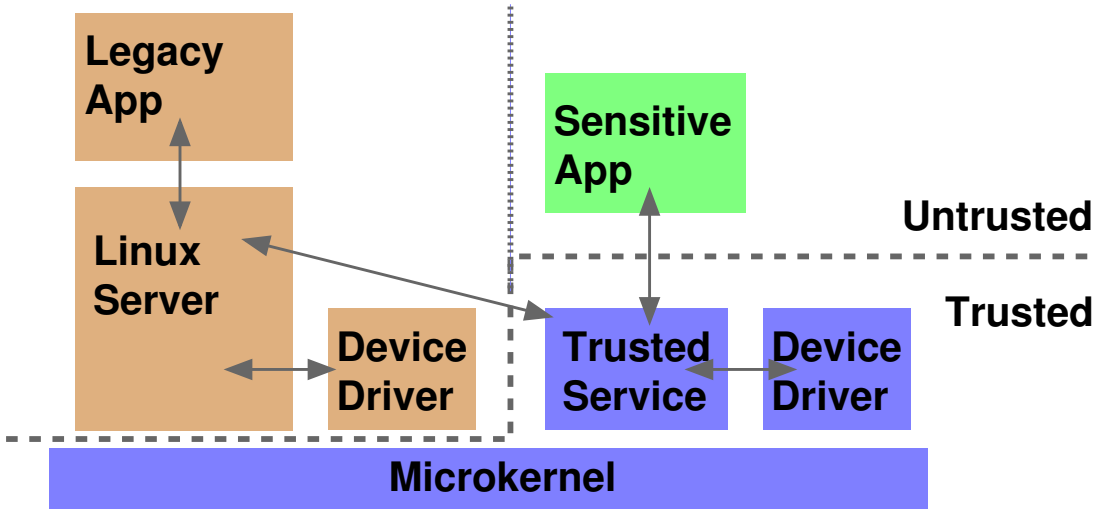
STATUS

- Microkernel operational
  - ➜ 10,000 lines of code
  - ➜ highly efficient
- Basic infrastructure exists
  - ➜ core services
  - ➜ Linux server, drivers



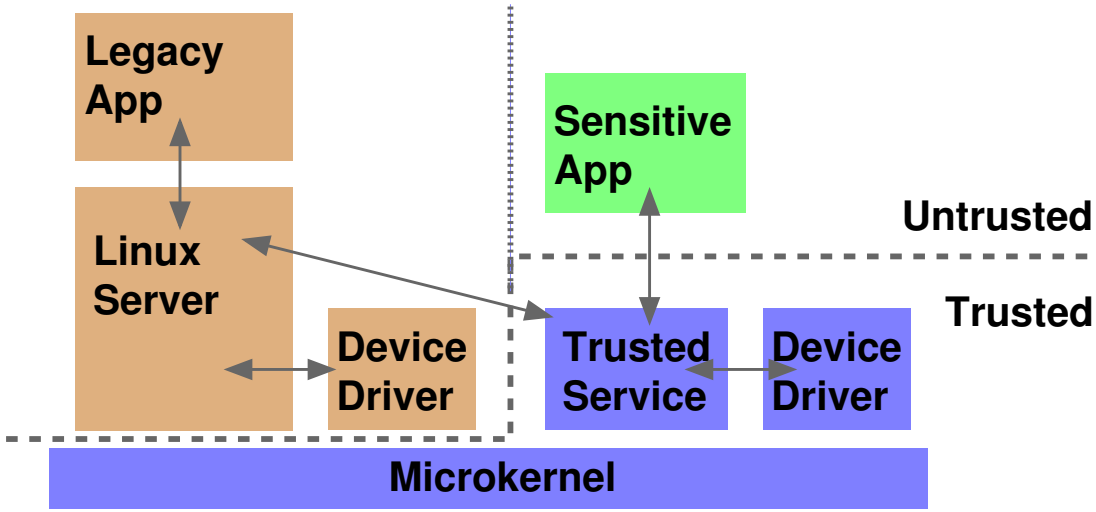
STATUS

- Microkernel operational
  - ➜ 10,000 lines of code
  - ➜ highly efficient
- Basic infrastructure exists
  - ➜ core services
  - ➜ Linux server, drivers
  - ➜ in commercial deployment



STATUS

- Microkernel operational
  - ➜ 10,000 lines of code
  - ➜ highly efficient
- Basic infrastructure exists
  - ➜ core services
  - ➜ Linux server, drivers
  - ➜ in commercial deployment



- Verification in progress
  - ➜ successful pilot project for verification of API “slice”
  - ➜ full-scale correctness proof project commencing
  - ➜ temporal analysis commencing