CTSRD

CRASH-worthy Trustworthy Systems Research and Development

Peter G. Neumann
Robert N. M. Watson

SRI International

UNIVERSITY OF CAMBRIDGE

CTSRD personnel

SRI International

Peter G. Neumann*
Hassen Saidi*
Patrick Lincoln*
John Rushby
Natarajan Shankar

University of Cambridge

Robert N. M. Watson*
Simon Moore*
Steven Murdoch*
Jonathan Anderson*
Ross Anderson
Steven Hand
Andrew Moore
Anil Madhavapeddy
Jonathan Woodruff
Philip Paeps
Tony Finch

Advisory board

TBD

* at PI meeting

Patrick Lincoln (SRI),
Peter Neumann (SRI), Jonathan Anderson (Cam), Hassen Saidi (SRI),
Robert Watson (Cam), Steven Murdoch (SRI), Simon Moore (Cam)

CTSRD intellectual history (partial)

SRI International

Multics (PGN)
PSOS and HDM (PGN, et al.)
Separation Kernels (Rushby)
Newcastle DSS (Rushby/Randell)
DARPA CHATS (PGN)
PVS/SAL/YICES (Rushby/Shankar)
LynuxWorks Separation Kernel (Rushby/DeLong)

University of Cambridge

CAP (Needham-Wilkes)
MAC Framework (Watson)
FreeBSD/TrustedBSD (Watson)
Multithreaded CPUs (S. Moore)
Xen (Hand)
Capsicum (Watson/Anderson)
MirageOS (Madhavapeddy)

CTSRD

CRASH-worthy Trustworthy Systems Research and Development

Peter G. Neumann
Robert N. M. Watson

UNIVERSITY OF CAMBRIDGE

Before CRASH

- Watson/Anderson (Cambridge), Laurie/Kennaway (Google); USENIX Security 2010
- Capsicum: practical capabilities for UNIX
  - Capabilities, capability mode
  - Delegation-oriented model
- Application/UI-centric security models
- Hybrid capability philosophy: runs UNIX, hybrid, and capability-only applications

If you could revise the fundamental principles of computer system design to improve security…

…what would you change?

Observations

- Software designs that employ the principle of least privilege are neither easily nor efficiently represented in current hardware
- Kernels and programming language runtimes (TCBs) build directly on hardware in C: enormous and unsound
- Implementations embody artifacts of security policies rather than design principles

Why now?

- New opportunities for hardware-software interface research created by FPGA soft cores and open source software stacks
- Trend towards exposing inherent hardware parallelism to the programmer: context switching can now be avoided
- Mature translations from type-safe language to expression-limited byte codes — security not assured, but at least possible

CTSRD agenda

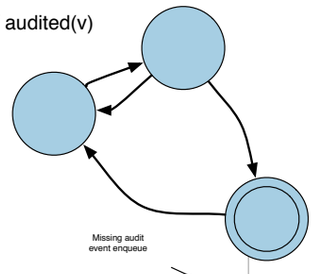
1. Develop hybrid capability HW architecture supporting efficient, easy decomposition
2. Decompose and minimise low-level TCBs: hypervisor, OS kernels, language runtimes
3. Map type-safe languages to new primitives
4. Pragmatically apply formal methods
5. Adapt runtime system to continuously validate complex security principles
6. Apply Capsicum's hybrid design philosophy

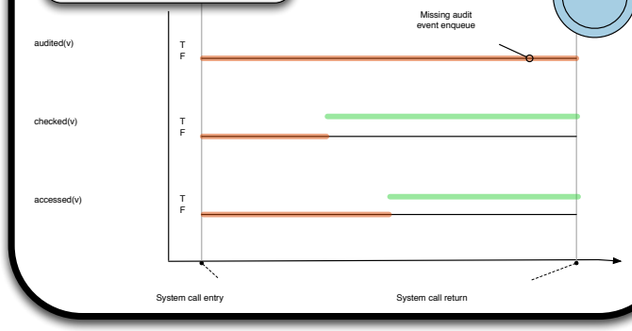
TESLA

accessed(v) → FINALLY audited(v)

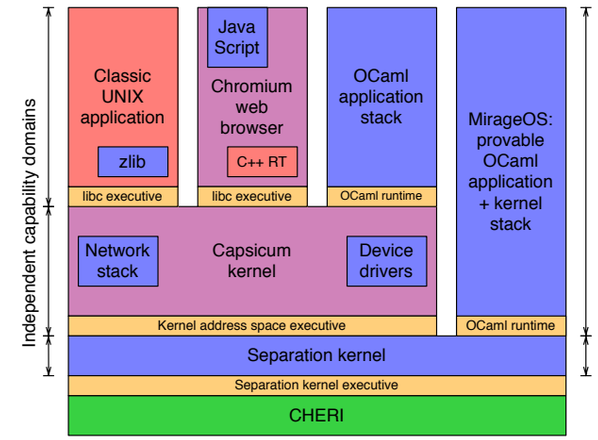
```
if (check_access(...))
{
    err = grant_access(...);
    if (err != OK)
        goto fail;

    audit_access(...);
}
```



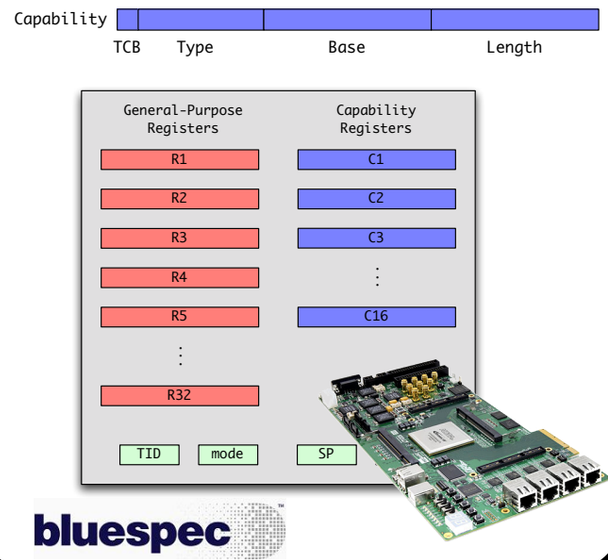


CTSRD



- Legacy application code compiled for general-purpose registers
- Hybrid code blending general-purpose registers and capabilities
- High-assurance capability-only code; stand-alone or "pools of capabilities"
- Per-address space memory management and capability executive

CHERI



CTSRD **validates security design principles** and **enforces application security structure** using a hybrid processor architecture.

CTSRD elements

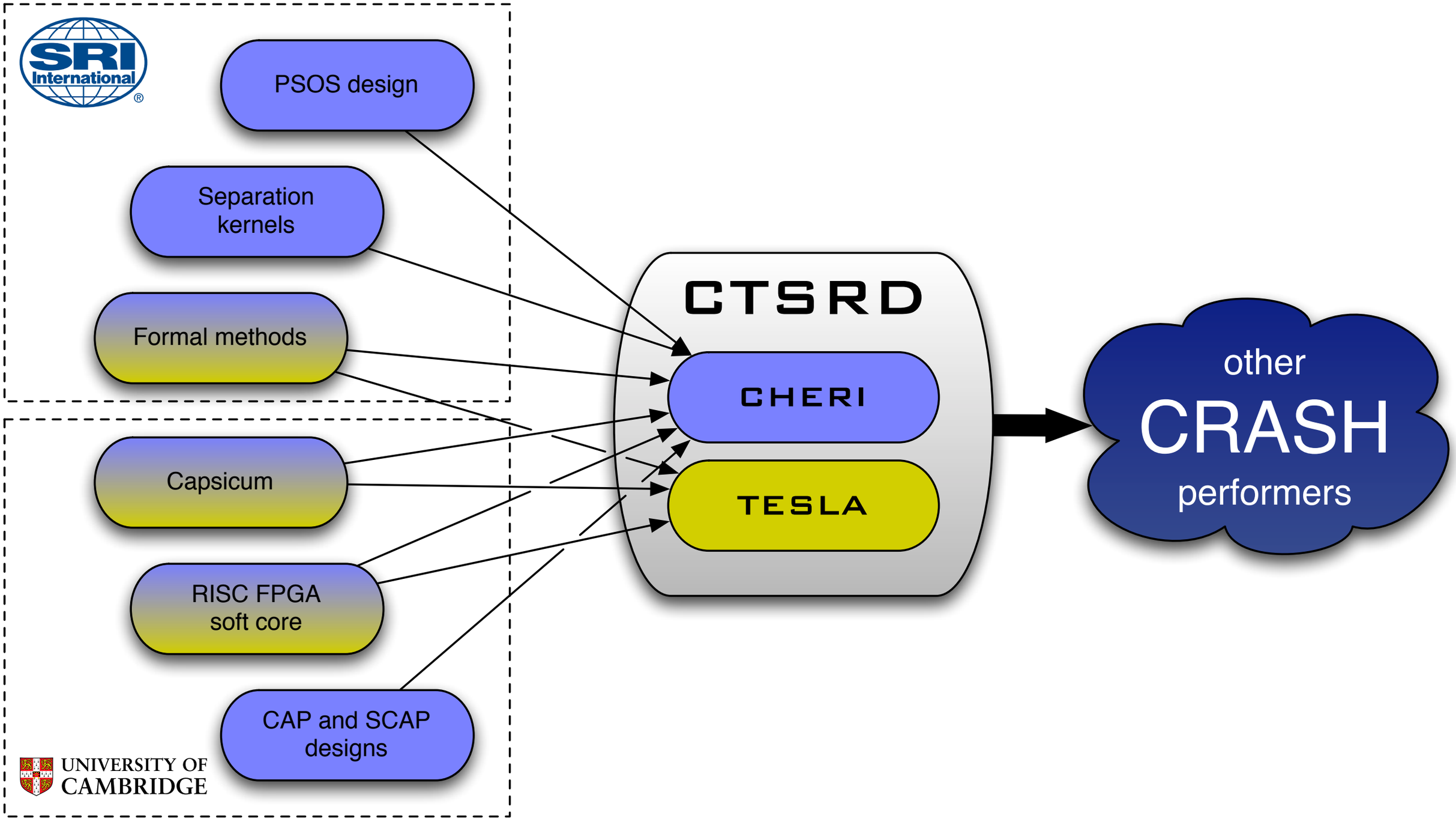
CHERI: capability hardware enhanced RISC instructions

Modify Cambridge RISC FPGA soft core to allow granular expression of program protection structure through a hybrid capability model.

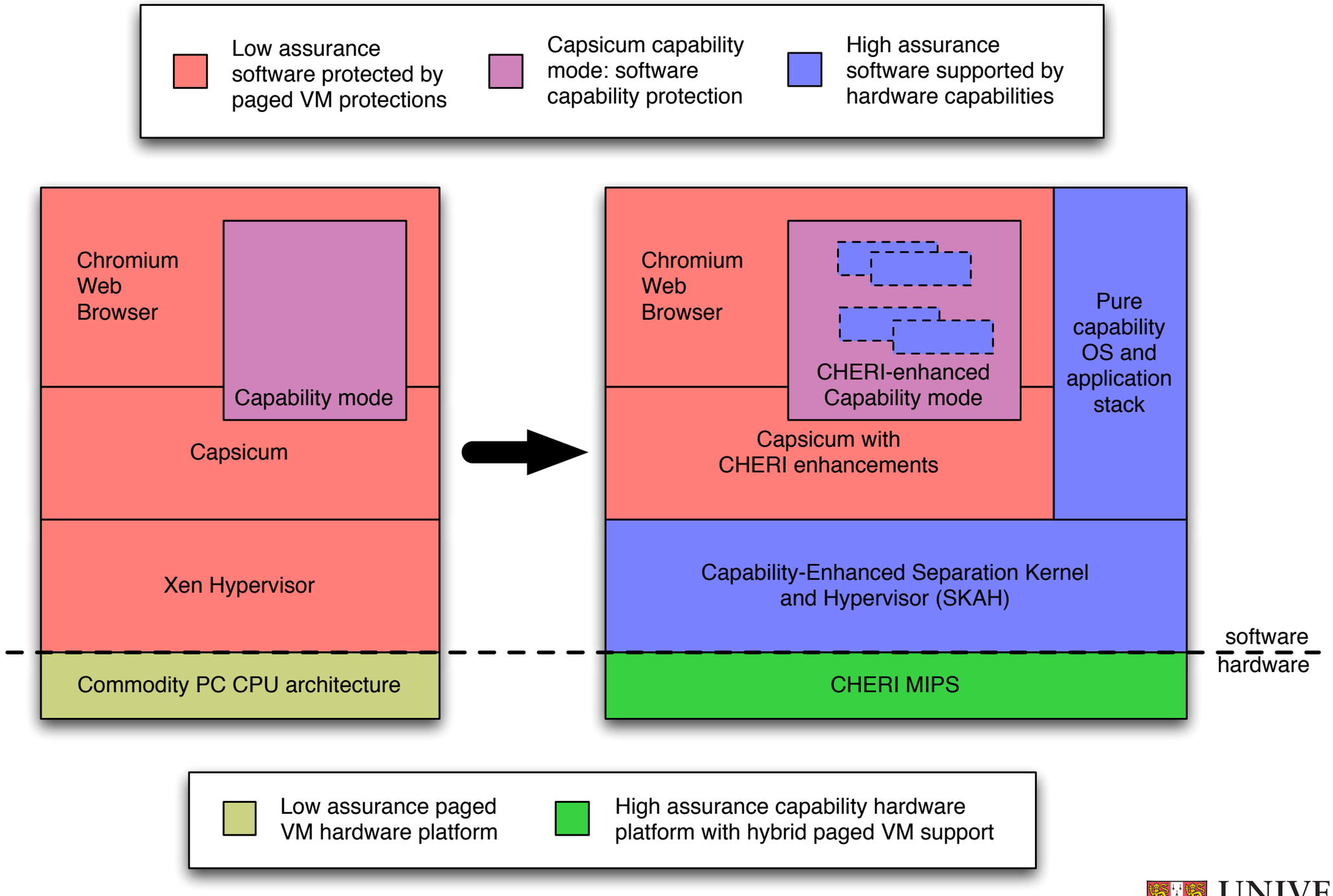
Develop separation kernel, hybrid Capsicum OS, capability-enabled OCaml runtime with MirageOS application stack.

TESLA: temporally enforced security logic assertions

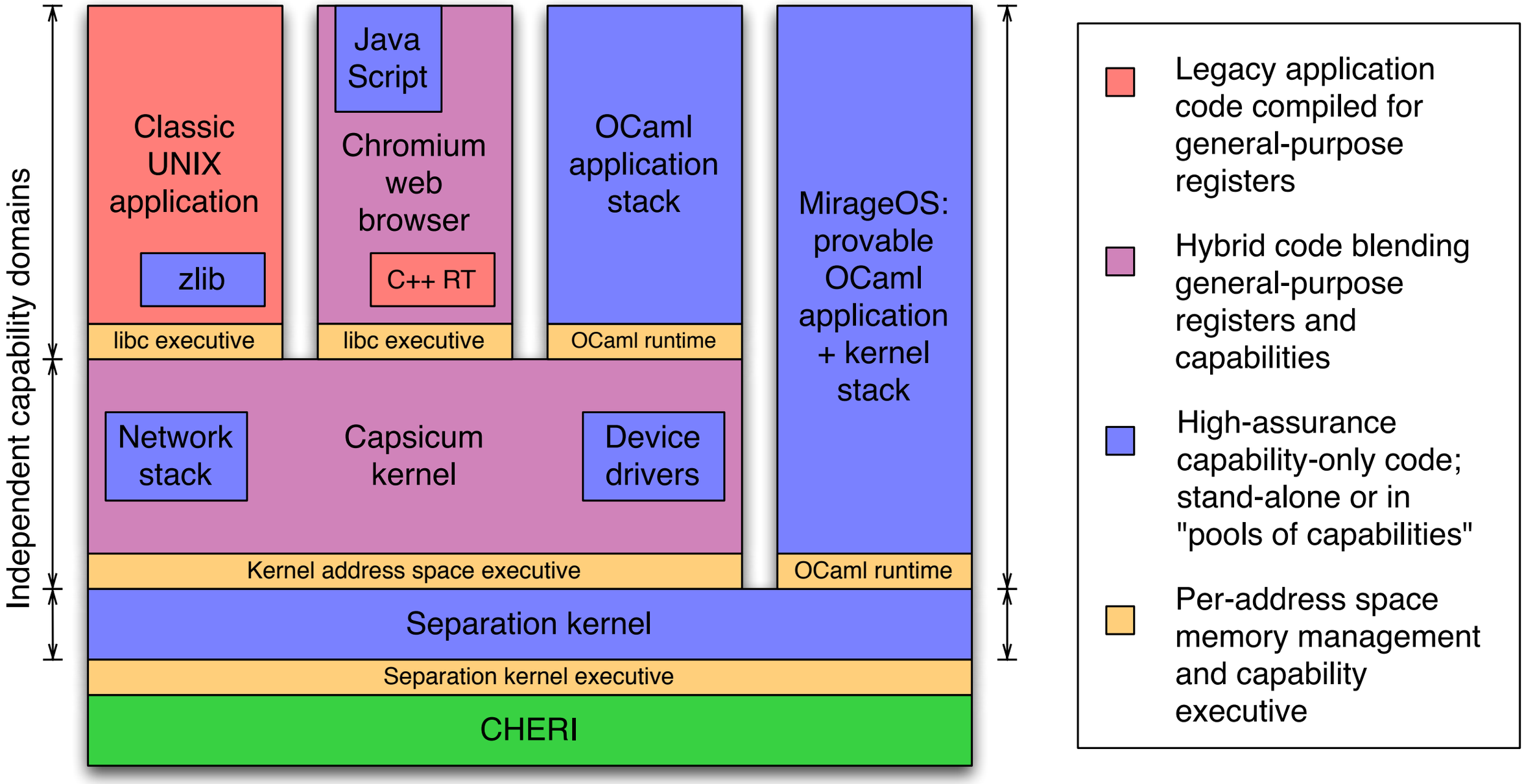
Translate security principles from design into software implementation by blending temporal logic expressions and software / hardware-assisted runtime validation


SRI International
PSOS design
Separation kernels
Formal methods
Capsicum
RISC FPGA soft core
CAP and SCAP designs
UNIVERSITY OF CAMBRIDGE
CTSRD
CHERI
TESLA
other
CRASH
performers

CTSRD proposal picture



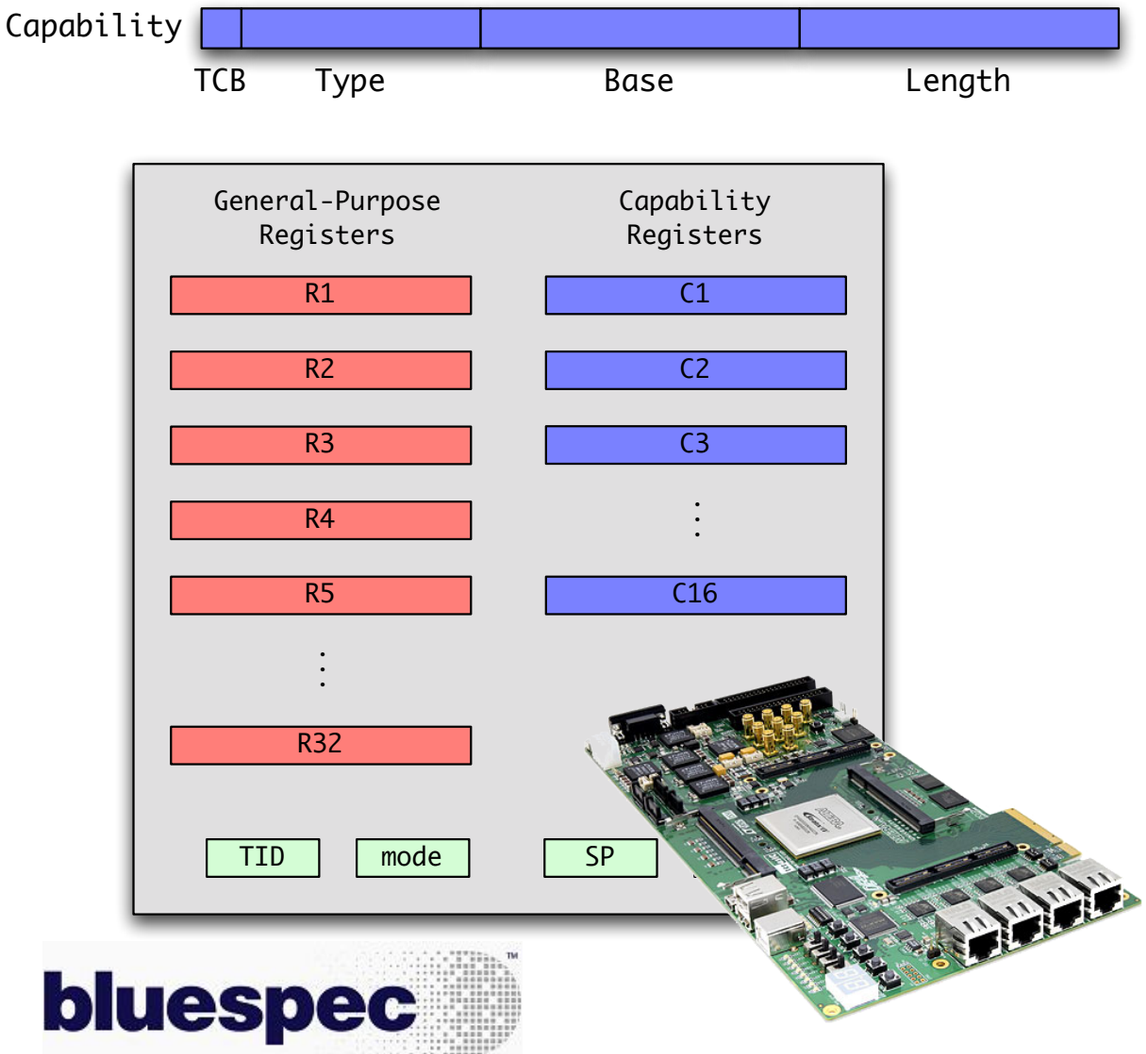
Pictures change!



CHERI

- Modify hardware platform to enforce elements of program protection structure
  - New capability registers, tagged memory
  - Replace context switches with hardware message passing within an address space
- Employ Capsicum's *hybrid design* principle
  - Can run legacy code, capability code, or even blends within a single process!
- Compiler changes for C to also support capabilities, OCaml to use only capabilities

CHERI capabilities



- 128-bit capability registers supplement general purpose registers
- New instructions load/store/load via/store via/manipulate capabilities
- Capabilities are interpreted relative to VM address space
- Data capabilities, user-defined typed procedure capabilities

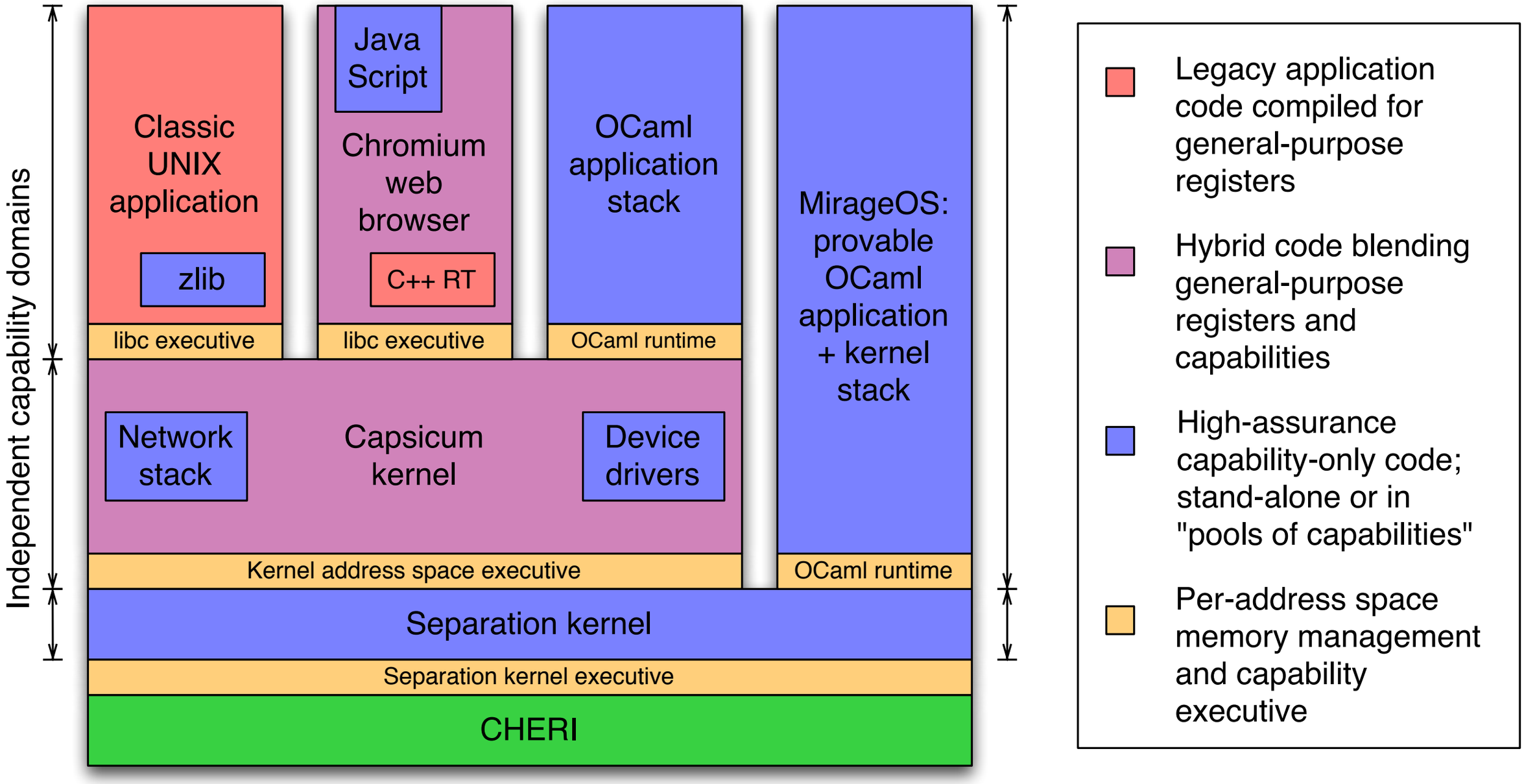
Processor modes

- **Hybrid mode**: Both general-purpose registers (GPRs) and capabilities may be used for load and store
- **Capability mode**: Only capabilities may be used for load and store; attempts to load or store via GPRs will trap
- Likely implementation through "capability 0", which constrains GPR loads and stores

Capabilities and VM

- Standard RISC software TLB approach to virtual memory: pages, processes, etc.
- Capabilities constrain memory use within an address space by limiting expression to permitted uses
- Each address space managed by an executive that defines memory model
- Different processes can use capabilities in different ways: kernels, libc, runtimes, ...

CHERI software stack



CHERI operating systems

- Capability-based separation kernel
- Hybrid Capsicum OS: hybrid executive, incremental adoption of capabilities: network stack, device drivers, etc.
- OCaml MirageOS: type-safe, provable operating system design that will build on the hardware capability model

CHERI applications

- Unmodified UNIX applications
- Hybrid applications
  - Legacy applications may transparently link capability-enabled libraries
  - Hybrid applications use a blend of capabilities and legacy code
- Capability-only applications and languages

TESLA

- Embed design-time security principles into executable code or code annotations
- Borrow ideas from model checking, but apply dynamically rather than statically:
  - Test assertions on experienced paths rather than all paths
- Continuous validation of principles by the runtime: fail-stop or exception handling
- Hardware assist via tightly coupled threads

TESLA assertions (I)

- C language assertions are instantaneous
  - Complex properties requires explicit kernel/application instrumentation
- However, many security design principles are fundamentally temporal
- New assertion representation and mechanism desirable!

TESLA assertions (2)

- Blend application/language-specific propositions and temporal quantifiers

  accessed(v) → FINALLY audited(v)
  ¬accessed(c, v) UNTIL checked(c, v)

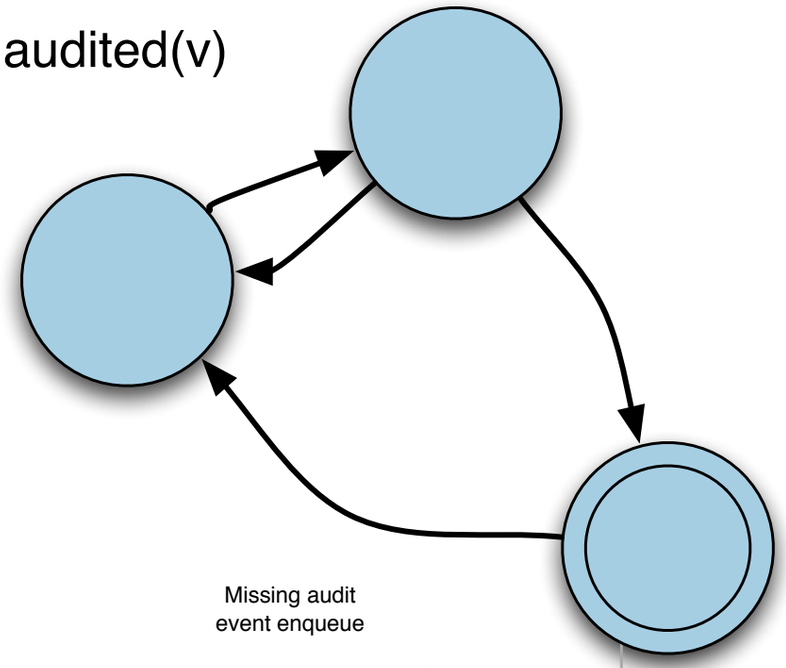
- Finite path semantics, sometimes truncated
- Certain properties can be evaluated continuously, other only on termination
- Typical finite path: system call enter-return
- Throw exception: panic? analyse? recover?

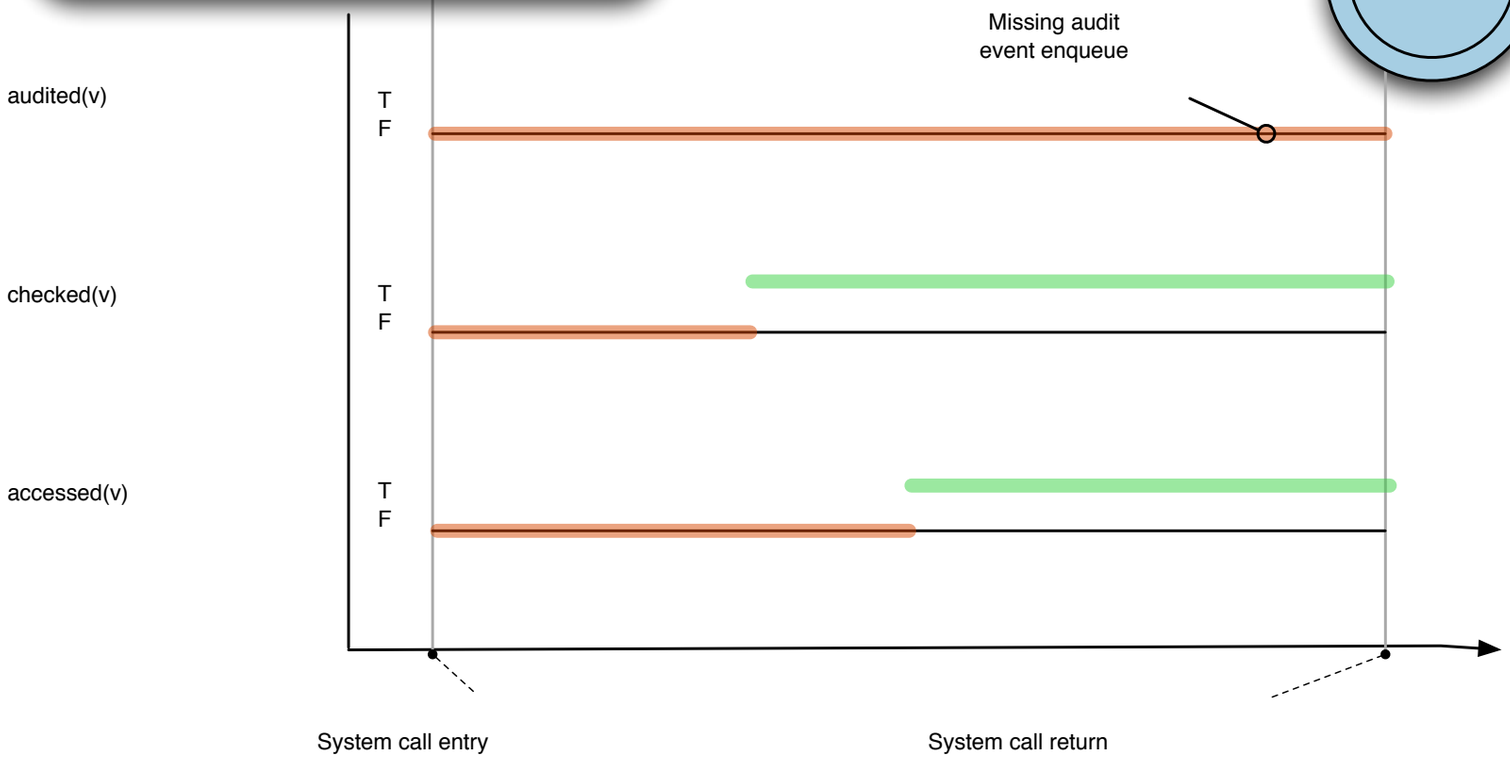
TESLA

accessed(v) → FINALLY audited(v)

```
if (check_access(...))
{
    err = grant_access(...);
    if (err != OK)
        goto fail;

    audit_access(...);
}
```

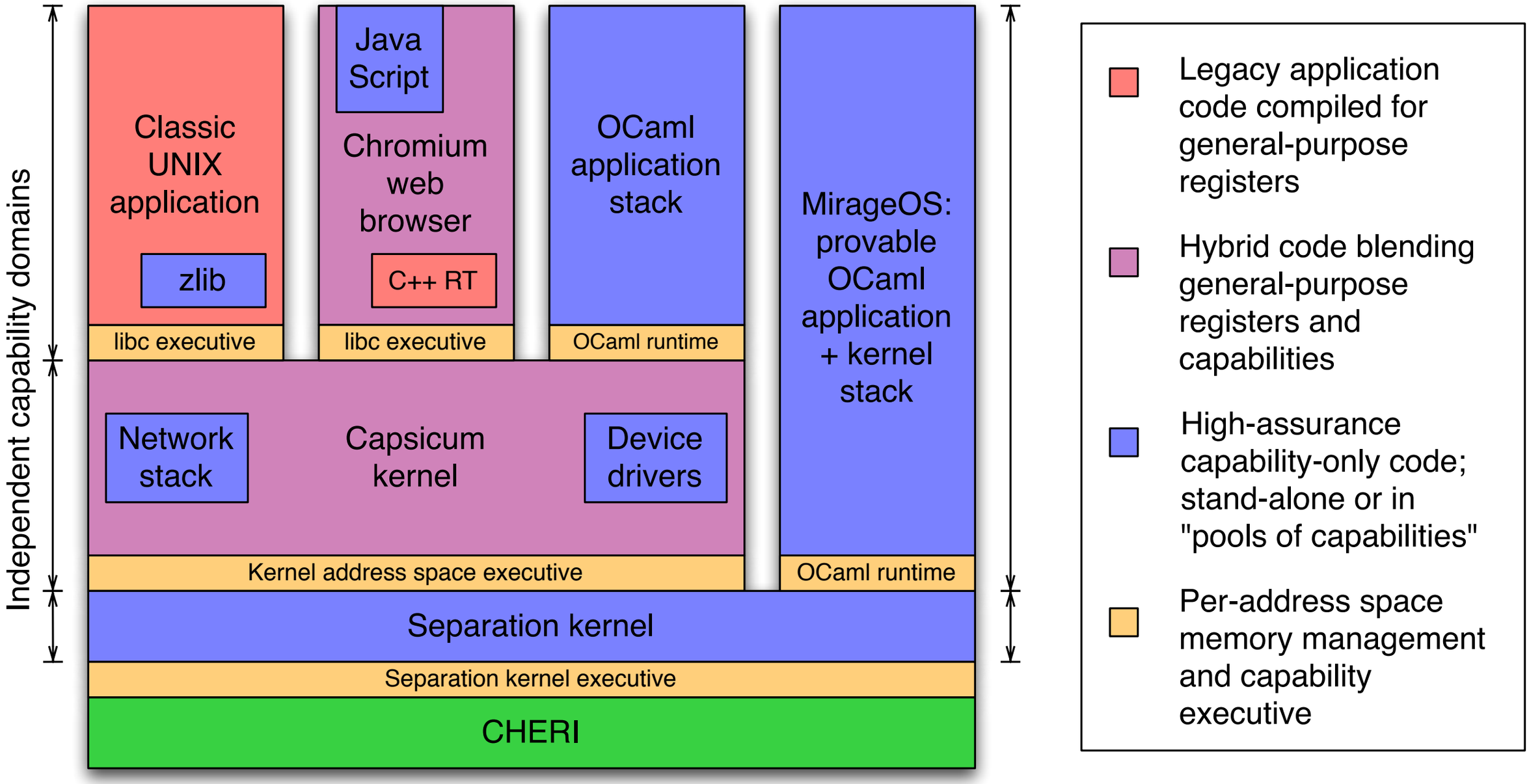




TESLA implementation

- Prototype based on Capsicum and DTrace
  - Blend of D + LTL compiled to automata
- Next: clang/LLVM compiler integration
  - Address soundness, completeness
- Then: hardware assist
  - Greater robustness, performance
  - Tightly coupled threads
- CHERI and TESLA turn out to be linked

TESLA integration



CTSRD products

- Open source CTSRD hardware-software research platform
- CHERI tagged hybrid capability architecture
- Bluespec-based FPGA soft core prototype
- Formally verified separation kernel
- Hybrid Capsicum OS, OCaml MirageOS
- TESLA temporal assertion system

Conclusion

- CHERI hybrid capability tagged architecture
- Separation kernel, MirageOS OCaml stack
- Hybrid Capsicum OS and applications
- TESLA continuous principle validation
- Formal grounding throughout

Q&A

CTSRD products

- Open source CTSRD hardware-software research platform
- CHERI tagged hybrid capability architecture
- Bluespec-based FPGA soft core prototype
- Formally verified separation kernel
- Hybrid Capsicum OS, OCaml MirageOS
- TESLA temporal assertion system