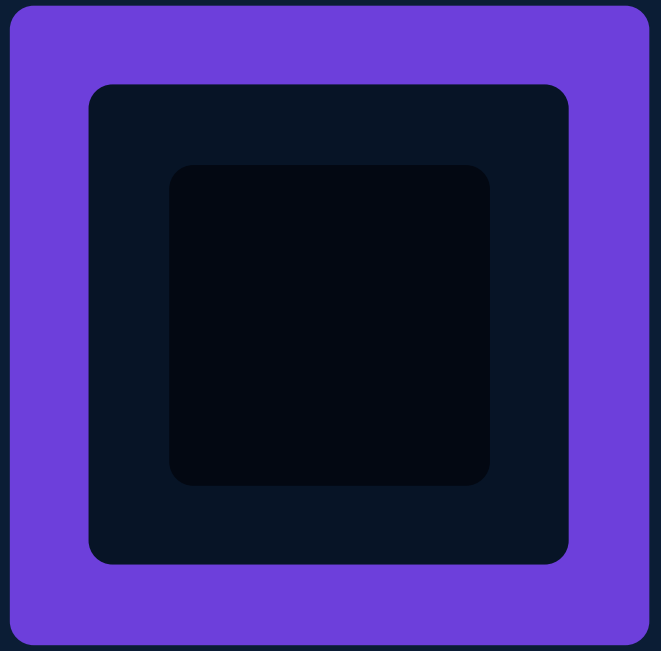
ENGINEERING FOR COMPROMISE

Designing systems that remain trustworthy secure under adversity



Dr. Ron Ross | CEO, RONROSSECURE, LLC
ISTS Fellow, Dartmouth College

Prevention is essential — but it cannot be the final assumption

THE REALITY

Modern systems are too interconnected, software-intensive, externally dependent, and continuously changing to assume every element will remain trustworthy.

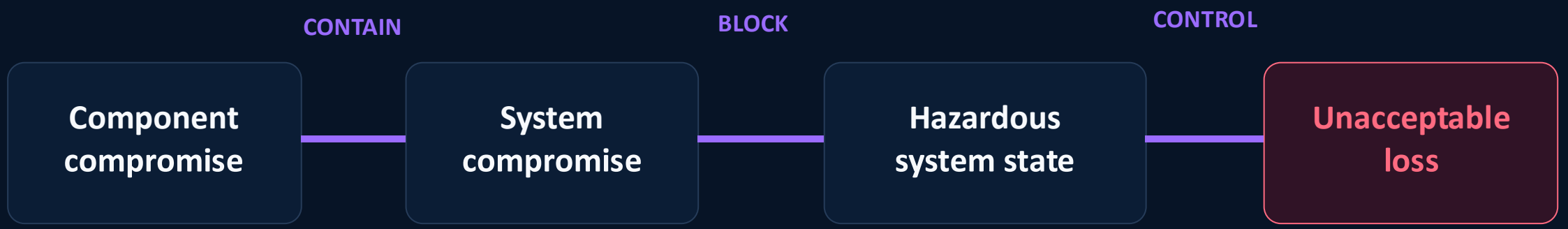
THE ENGINEERING QUESTION

If this system element were fully controlled by an adversary…

- What authority would it inherit?
- What information and functions could it reach?
- Could that reach produce unacceptable loss?

EfC controls consequence after prevention fails.

Architecture must break the causal chain before catastrophe



Component Compromise ≠ System Compromise ≠ Unacceptable Loss

The events become connected only when architecture permits the adversary's authority and reach to propagate.

Compromise containment boundaries limit what hostility can reach

ASSUME THE ELEMENT IS HOSTILE

- Information read or inferred
- Data and configuration modified
- Privileges and identities exercised
- Functions and interfaces invoked
- Physical processes influenced
- Maintenance and recovery paths used

COMPROMISE CONTAINMENT BOUNDARY

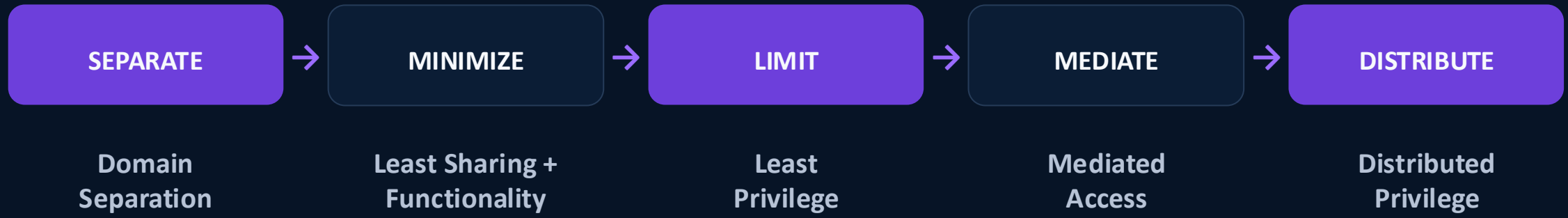
A system-level boundary that limits:

- **AUTHORITY**
- **INFORMATION**
- **CONTROL**
- **FUNCTIONALITY**
- **INFLUENCE**

A CCB may combine cyber, physical, procedural, human, and organizational constraints.

Trust is an architectural dependency — not a product label.

Bounded compromise emerges from principles working together



Protective Failure

Move toward safe states before loss thresholds are crossed.

Anomaly Detection

Observe when architectural invariants no longer hold.

Protective Recovery

Restore an authenticated, trustworthy basis—not just service.

Evidence

Substantiate the composed system claim.

Compromise margin makes tolerance — and hidden fragility — visible

COMPROMISE MARGIN

How much adversary success or loss of trust can occur before unacceptable loss becomes reachable?

1. Single compromise
2. Multiple independent compromise
3. **Correlated compromise**
4. **Common-mode compromise**

TEST INDEPENDENCE

Two barriers are not truly independent if they share:

- Identity provider
- Administrator or workstation
- Hypervisor or control plane
- Cryptographic root
- Build, update, or recovery path

When confidence declines, systems must fail and recover protectively

DETECT

Monitor the invariants that make the security argument true—not only attacker signatures.

PROTECT

Revoke authority, isolate domains, reduce functionality, or stop before crossing the loss threshold.

RECOVER

Reestablish trusted keys, identities, configuration, software, data, boundaries, and operator authority.

A system that sacrifices functionality to prevent catastrophic loss may be demonstrating successful security engineering.

Trust less — and demand evidence that local compromise stays local

MINIMIZE THE TRUSTED FOOTPRINT

Ask how few elements must remain trustworthy for the system-level claim to survive.

Concentrate assurance on critical mediators, roots of trust, authorization mechanisms, recovery authorities, and selected physical or procedural controls.

SUBSTANTIATE THE CLAIM

- **Architecture + information-flow analysis**
- Privilege and dependency analysis
- Formal methods + model checking
- Fault injection + adversarial testing
- Recovery exercises + operational monitoring

Evidence has an operational lifetime: mission, configuration, dependency, and adversary changes can invalidate it.

A repeatable methodology turns compromise assumptions into maintained claims

1. Define mission + unacceptable losses
2. Identify compromise-reachable hazards
3. Assume selected elements are hostile
4. Determine compromise reach
5. Establish containment boundaries
6. Minimize capability within boundaries
7. Mediate necessary interactions
8. Analyze propagation + margin
9. Engineer detection, failure + recovery
10. Substantiate and maintain the claim

Apply through mission analysis, requirements, architecture, design, implementation, integration, verification, transition, operation, maintenance, and disposal.

The strongest system can lose trust without losing control of the mission.

A trustworthy secure system is not necessarily one in which nothing can be compromised.

It is one engineered so that compromise does not automatically become catastrophe.

PREVENT • CONTAIN • DETECT • PROTECT • RECOVER • SUBSTANTIATE