An Architectural Approach for Cost Effective Trustworthy Systems

Ihor Kuz, Liming Zhu, Len Bass, Mark Staples, Xiwei Xu
Trustworthy Systems

Deserving of Trust

- Availability
- Security
- Reliability
- Safety
Building Trustworthy Systems
Building Trustworthy Systems

Untrusted Service

Trusted Service

seL4

Hardware

Reason about a whole system without having to reason about the behaviour of every component

Controlled separation
Cost Effective Trustworthy Systems

• Verification is expensive
  ➡ make sure it works the first time
• Architecture-driven approach
Cost Effective Trustworthy Systems

• Verification is expensive
  ➜ make sure it works the first time
• Architecture-driven approach
Cost Effective Trustworthy Systems

[Diagram]

- Architectural model
- Untrusted component code
- Trusted component code
- seL4

Trustworthiness Property Analysis

**Architecture is realised as the policy for controlled separation**
Cost Effective Trustworthy Systems

Architectural model

Untrusted component code + Trusted component code + Framework code + seL4

Proof + Proof + Proof

Full system code + Whole system proof

Hardware

Trustworthiness Property Analysis
Working Example: SAC

• Secure Access Controller (SAC)
• Securely switch a terminal between two strictly separated networks
Working Example: SAC

- Secure Access Controller (SAC)
- Securely switch a terminal between two strictly separated networks
Working Example: SAC

- Secure Access Controller (SAC)
- Securely switch a terminal between two strictly separated networks
SAC Implementation

1. SAC Implementation
2. SAC controller
3. Router Manager
4. Router
5. Timer
6. seL4

Nic-C
Nic-A
Nic-B
Nic-D
Architectural Reconstruction in AADL

- Mode A
- Mode B
- Router Manager
- SAC Controller
- Nic-C
- Nic-A
- Nic-B
- Router A
- Router B
- Nic-D
Confidentiality Analysis with SPIN

PROMELA model

```pseudocode
active proctype DataSourceA()
{
    int data;
    idle:   ctrl_CM_A?connect; goto connected;
    connected:
        do
            ctrl_CM_A?disconnect -> goto idle;
            data_DM_A?data; data_A = data;
            assert(data_A!=b);
            data_A_DM!a;
        od
}...
```

Counterexample

Property Fulfilled?

Yes

No

SPIN Model Checker

Confidentiality property

Notification

Property Fulfilled?
Results

• Architecture analysis works
  – can reduce effort of whole system verification

• Helps spot problems early on
  – Terminal network card (NIC-D) can store data
    → storage channel unless flushed explicitly

• AADL and SPIN sufficient for SAC
  – **BUT**: other systems need more dynamism

• Next steps
  – code generation: glue code and framework
  – architecture support for verification
  – trusted patterns