Modern Systems: Security

PRESENTED BY ANDREW FERRAIUOLO

CORNELL UNIVERSITY CS 6510

9.20.2016

Two Secure Systems Approaches

Fabric (Myers)

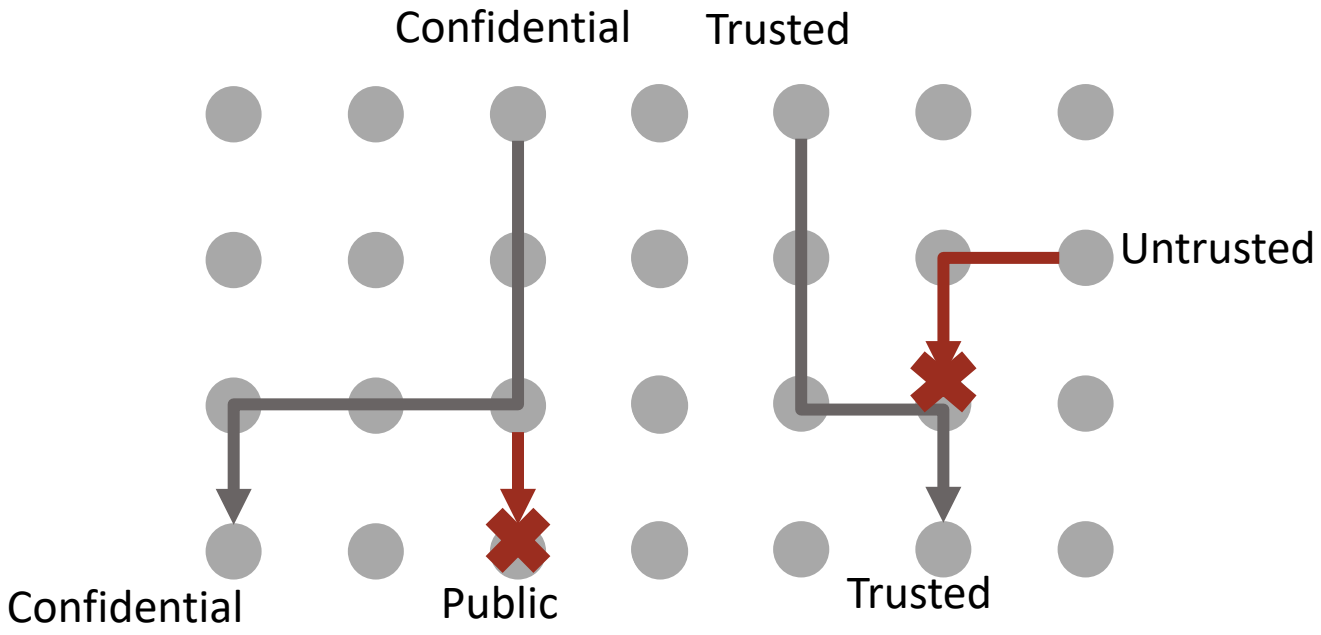
- Information flow

Nexus (Sirer & Schneider)

- Credentials-based authentication
- Hardware root-of-trust

Information Flow

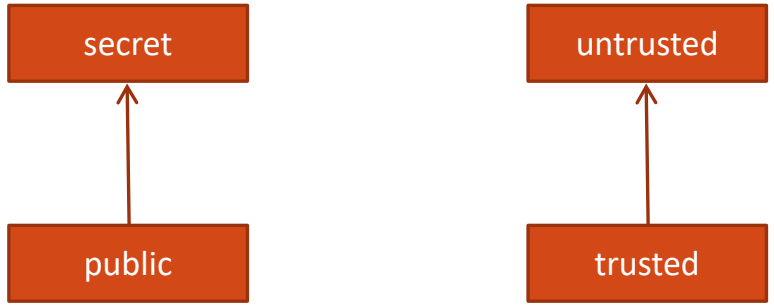
Track and constrain the propagation of data throughout the system



Information Flow Background

Lattice Model (Denning 1976)

$\texttt{public} \sqsubseteq \texttt{secret}$



Confidentiality and integrity are duals (Biba 1977)

Noninterference (Goguen 1982) – If an attacker can observe data in set L but not H, execution looks deterministic regardless of H.

Applications of Information Flow

Operating Systems

- DStar, HiStar (Zeldovich 2008)

Digital logic

- Gate-level information flow tracking – (Tiwari 2009)

Hardware architecture

- Dynamic information flow tracking – (Suh 2003)

Type Systems

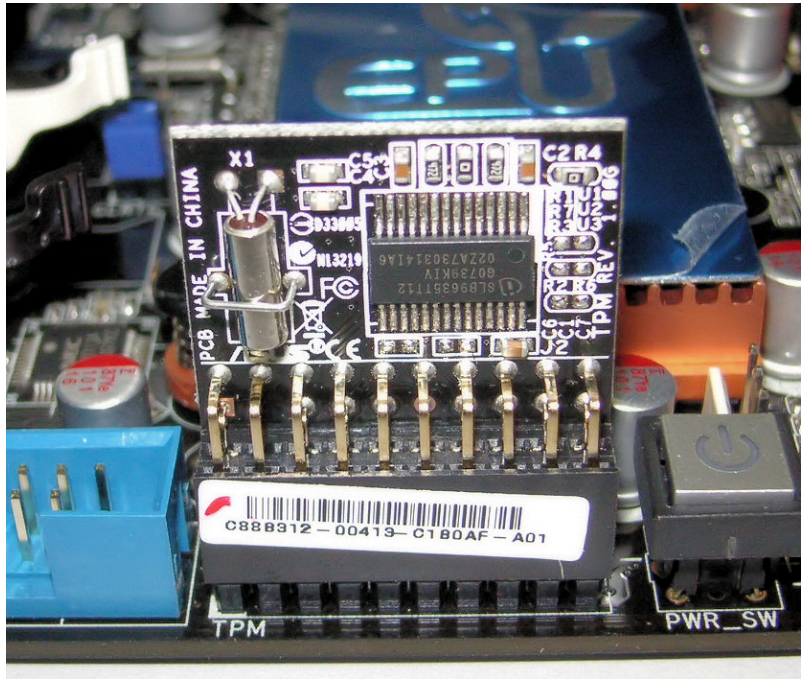
- A lot of work (Survey: Sabelfield 2003)

Hardware Authentication: TPM

Hardware support for providing remote attestation of software

Problems:

- Cannot describe state-dependent trust
- Compromises privacy
- Provides purely **axiomatic trust**



Trust Bases

Axiomatic – A principal is assumed to be trusted

- Trusted platform module

Synthetic – Transforms untrusted code into trustworthy code

- Reference monitors, sandboxing

Analytic – Predicts whether certain program behaviors are possible

- Proof-carrying code (Necula 1996), information flow!

Credentials Based Authorization

Permit access when a principal satisfies some property (Appel 1999)

$$\frac{\mathcal{P}_1,\ \mathcal{P}_2,\ \ldots,\ \mathcal{P}_n}{\mathcal{F}}$$

Goal Formula: conditions under which resource can be accessed

Credentials: a proof that satisfies a goal formula

Nexus adds features to support all three bases of trust.

- Can Nexus be used to perform information flow analysis? Would you want it to?

Fabric

Goal: Secure distributed computation

Approach:

- *Secure language* – information flow type system enforces noninterference
- *Decentralized security* – multiple distrusting parties, no central authority

Principals

Represent entities (e.g., users, groups, nodes) and privileges.

Acts-for relation: supports delegation, can be used for access control

```
principal user, priv;
    …
    if (user actsfor priv) {
        … do action …
    }
```

Labels

Express policies in the code

- labels are associated with data
- describe which principals can act on them and how

Policies are enforced mostly statically at compile-time

- Some checks, e.g. for remote procedure calls are checked dynamically

Labels

Decentralized label model (Myers 2000)

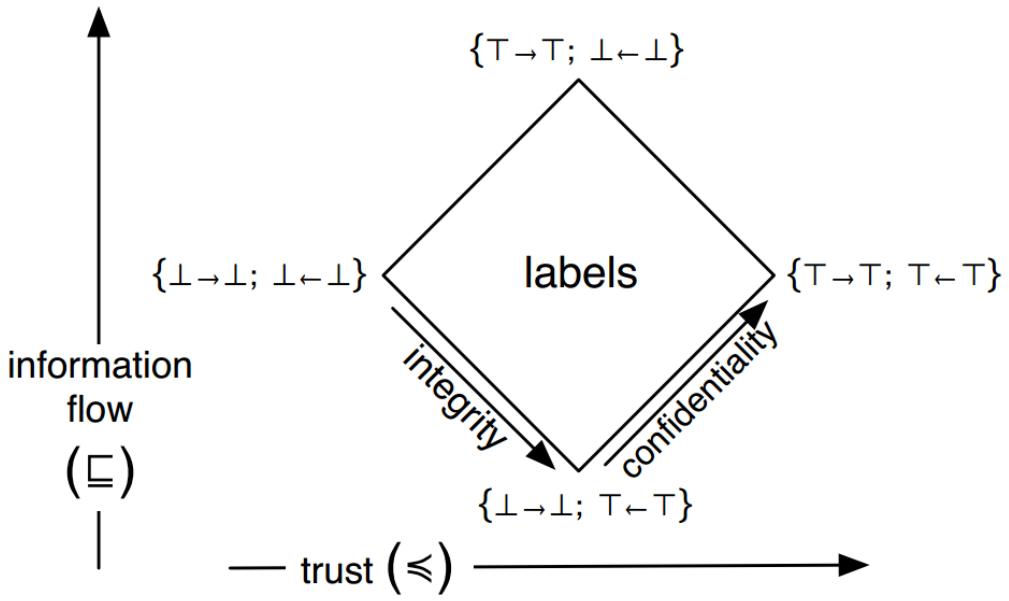
- Written in terms of principals
- Confidentiality { alice → bob}
- Integrity { alice ← bob }

Flow ordering

- permitted flow among labels

Trust ordering

- restrictiveness of policy
- (new in this paper)



Label Example

```
int {alice→bob} x;
int {alice→bob, charlie} y;
x = y; // OK: bob ≽ (bob ∨ charlie)
y = x; // Invalid
if (charlie actsfor bob) {
    y = x; // OK: (bob ∨ charlie) ≽ bob

}
```

Remote Calls

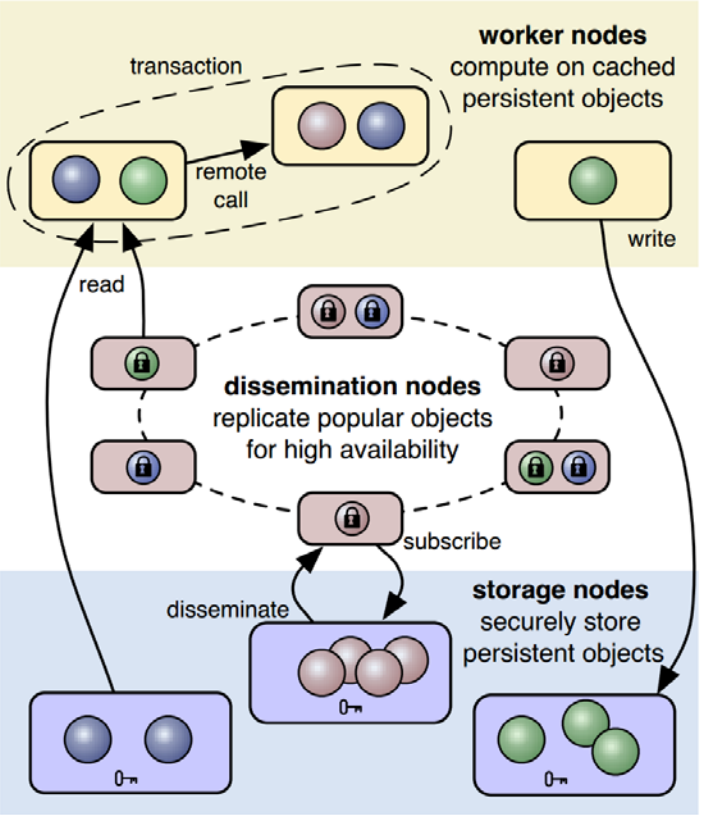
```
1  void m1{alice←} () {
2    Worker w = findWorker("bob.cs.cornell.edu");
3    if (w actsfor bob) {
4      int{alice→bob} data = 1;
5      int{alice→} y = m2@w(data);
6    }
7  }
8
9  int{alice→bob} m2{alice←} (int{alice→bob} x) {
10   return x+1;
11 }
```

System Architecture

Fabric nodes take one of three forms

- Storage nodes – persistent objects
- Worker nodes – perform computation
- Dissemination nodes – data replication

A single host can have multiple nodes



Evaluation: CMS

	Page Latency (ms)		
	Course	Students	Update
EJB	305	485	473
Hilda	432	309	431
FabIL	35	91	191
FabIL/memory	35	57	87
Java	19	21	21

Table 1: CMS page load times (ms) under continuous load.

2X-4X slowdown compared to regular Java

Evaluation: OO7 Benchmark

	total	app	tx	log	fetch	store
Cold	9153	10%	2%	12%	74%	2%
Warm	6043	27%	3%	6%	61%	3%
Hot	840	46%	14%	24%	0%	17%

Table 2: Breakdown of OO7 traversal time (times in ms)

The regular Java implementation is 10X faster

Nexus

Goal: Support authorization with all three trust bases.

Approach:

- Nexus authorization logic (NAL) – describes principals and their beliefs
- Reference monitor – general-purpose proof-checker enforces NAL
- OS and hardware features for each basis of trust.

Nexus & Trust Bases

Axiomatic

- *trusted platform module* – supports attested storage /w Merkle trees

Analytic

- *Introspection* – exposes view of kernel state
- *e.g. process lookup tables, IPC ports, application state*

Synthetic

- *Interpositioning* – reference monitor can respond to untrusted I/O

Nexus Authorization Logic

NAL formulas have form “P says S” (see Schneider 2009)

- Principal P believes S to be true
- Predicates have application-defined meanings

Delegation

- “A *speaksfor* B” means that if A says S then B says S.

Examples:

“Typechecker says isTypeSafe(PGM)”

“Server says NTP speaksfor Server on TimeNow”

Caching and State

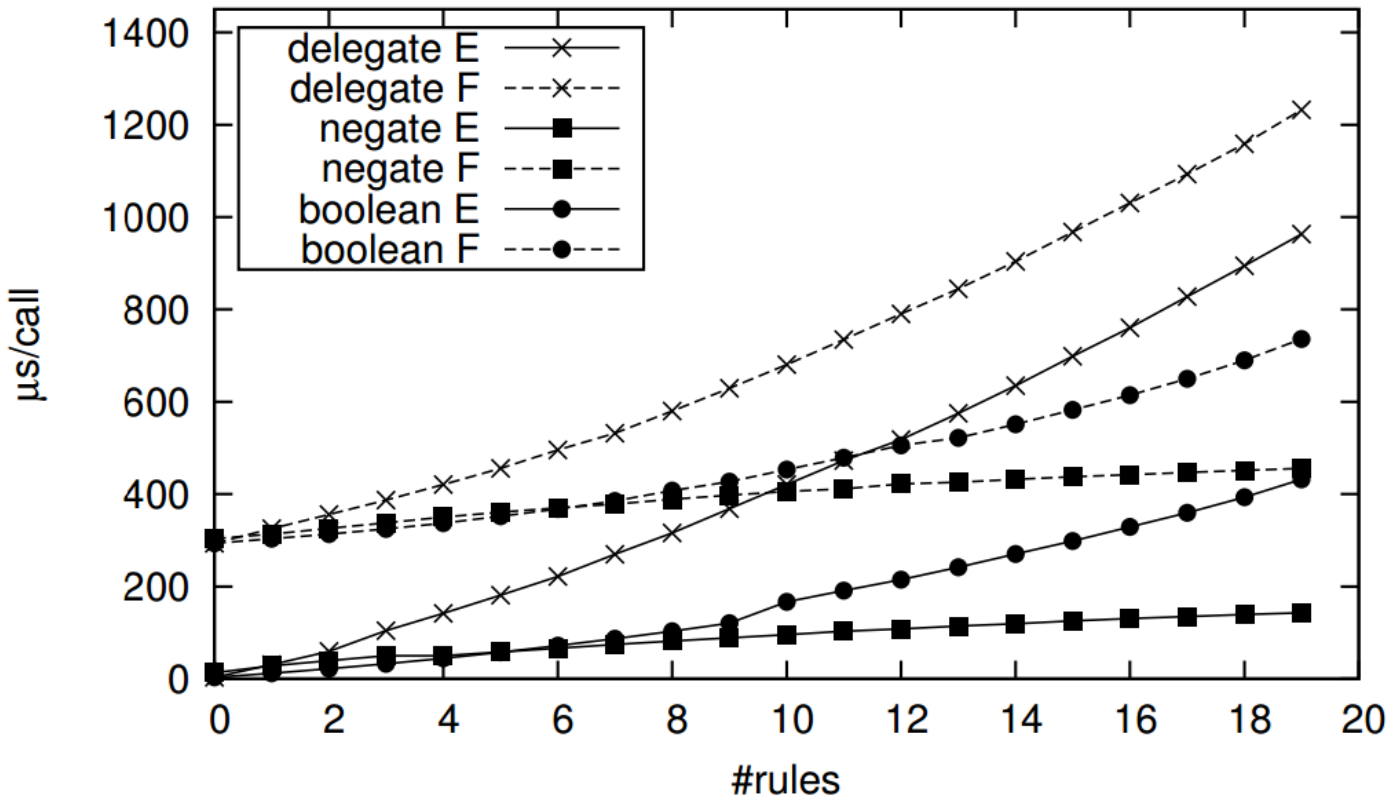
Decision cache - caches authorization decisions by the guard

Guard cache – caches partial proof-checking results.

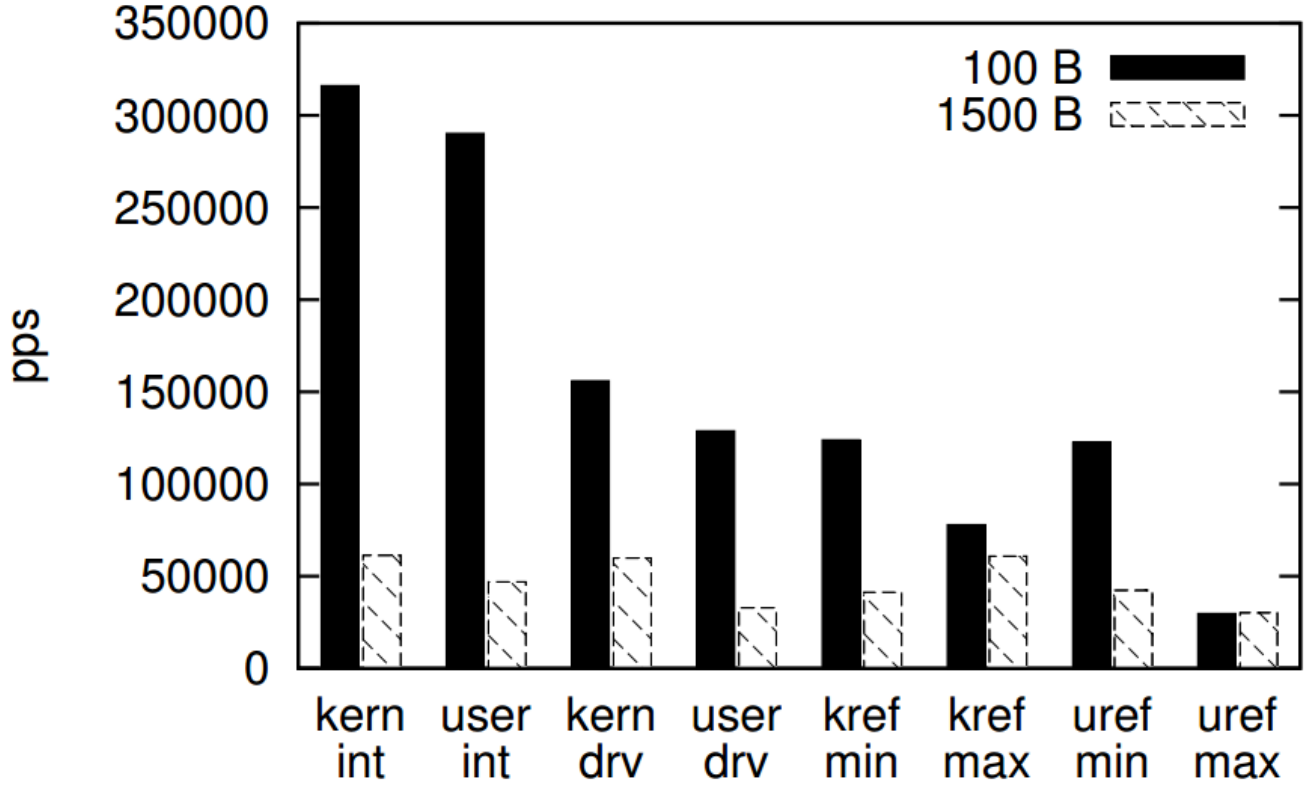
State and Authorities – stateful predicates cannot be cached!

- Authorities – listens on bound IPC, answers queries about stateful predicates.

Proof Evaluation Cost



Interpositioning Overhead



Comparison of Approaches

Information flow

Comprehensive/transitive

- Noninterfere-*ish* (declassification)
- Confidentiality/integrity, not availability

Policies

- Written by app developer
- Trust relationships

TCB: compiler(33k), crypto

Credentials based authentication

Constrains at resource interface

Policies

- Written by OS writer
- General propositional logic

TCB: OS(25k), theorem-prover, crypto

Questions

Overall, which approach do you prefer?

- Do you see any value in using both?

What is a reasonable (performance) cost for security in practice?

- Is it 10%? 10X? 100X?

What do you think should be the root-of-trust?

- The hardware? OS? Compiler? Theorem-provers?

Perspective

Noninterference is powerful, but too restrictive

- Practical systems rely on declassification/endorsement.
  - No single answer. (Survey paper: Sabelfield 2009)

Need to address covert channels to be truly comprehensive.

Is formal verification the answer?

- seL4