Modern Systems: Security

October 8, 2012

Background: Trusted Platform Modules

What is a TPM?

- 16 Platform Configuration Registers (PCRs)
- Random Number Generator (RNG)
- Endorsement Key (EK) burned in hardware

What can it do?

- Sealed storage
- Remote attestation
- Platform authentication

Who uses them?

- Microsoft, Google, Oracle, VMWare, etc.

Background: Information Flow

Information release vs information propagation

Security levels and noninterference

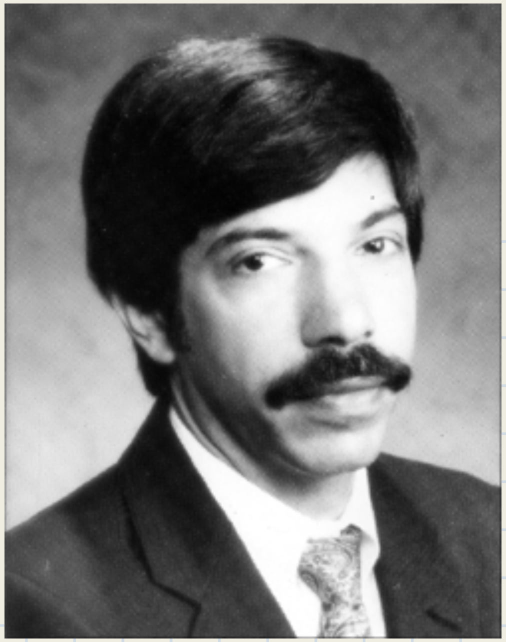
Explicit vs implicit flows

Covert channels

Security type systems and static analysis

Nexus

Emin Gün Sirer
Willem de Bruijn
Patrick Reynolds*
Alan Shieh
Kevin Walsh
Dan Williams**
Fred B. Schneider



*Now at GitHub **Now at IBM Watson

What’s wrong with TPMs?

Axiomatic Trust

Requires whitelisting software

Violates user privacy

Maintenance is a pain

Nexus Authentication Logic

Goal: an analytic basis for authorization

Mechanism: “A logic of belief”

Features:

- Principles and subprinciples
- Delegation
- Guards

Logical Attestation

Credentials-Based Authorization

- All access control based on credentials
- Credentials take form of Nexus Authorization Logic (NAL) proofs
- Guard on resources a simple proof checker

A Label is a statement attributed to a principal

“P says S”

Labels are credentials

Logical Attestation (con't)

Goal formulas guard system resources

"Owner says TimeNow < Mar19"

Goal formulas satisfied by gathering credentials

"Filesystem says NTP speaksfor Filesystem on TimeNow && NTP says TimeNow < Mar19"

Time-sensitive and non-monotonic statements must be backed by an authority. Authority is set by goal formula.

Nexus OS Features

Microkernel -- Small TCB (~21K LOC)

Some standard POSIX features
- python
- lighttpd
- sqlite

Non-standard features
- Labels, Labelstores, Guards, Authorities
- Introspection
- Interposition
- Secure Persistent Storage
- Secure Boot Sequence

Nexus OS Support for Logical Attestation

Cryptography is expensive, so Nexus only encrypts labels when exporting

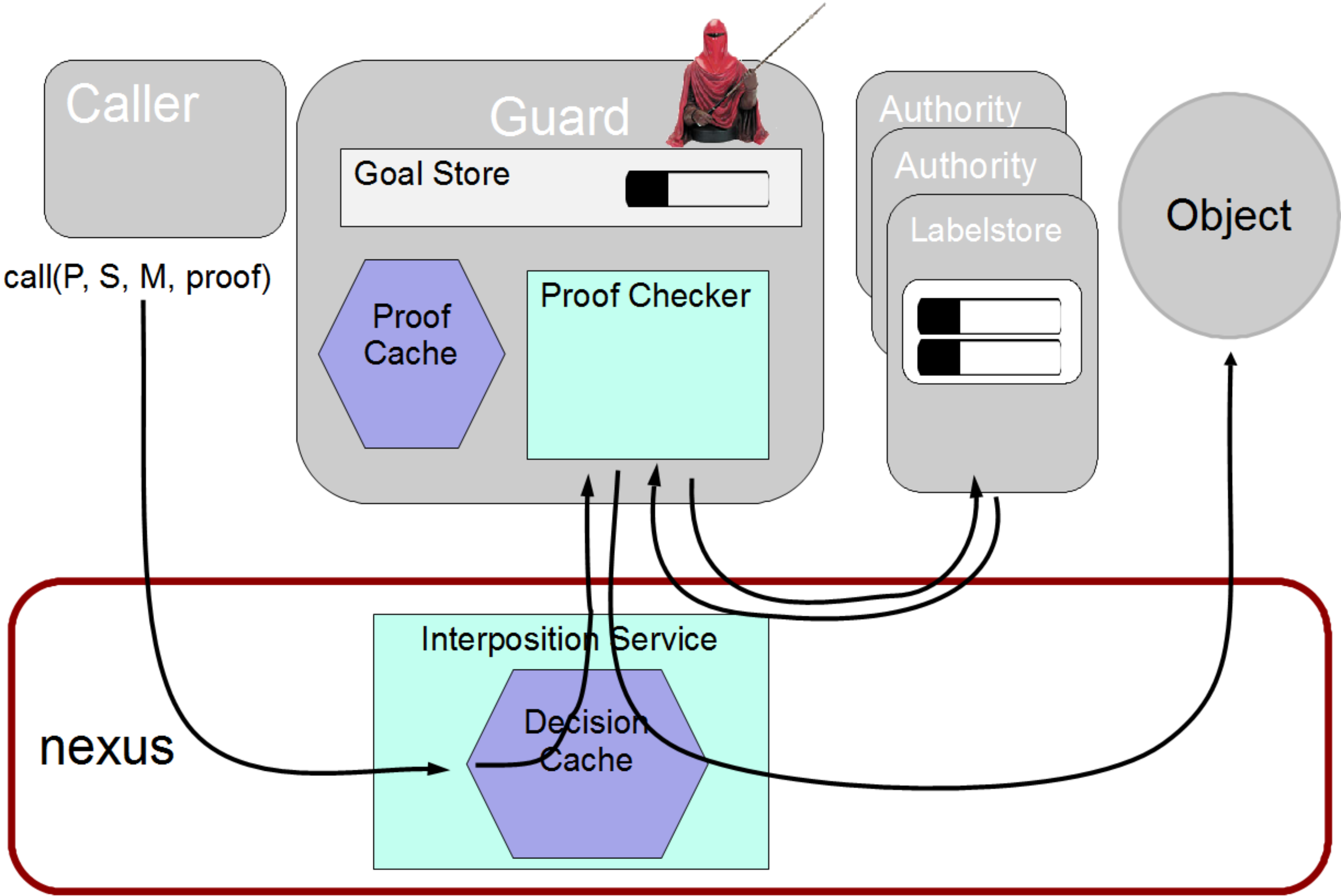
- Labels created with ‘say’ system call
- Labels kept in kernel data structure (labelstores)
- Labels can be passed through secure IPC

Invoking guards is expensive, so Nexus caches decisions whenever possible

- Decisions cache invalidated if relevant system state changes
- Proofs that rely on authorities can sometimes have lemmas extracted and cached

Overview

NB: Slide by Gun Sirer



Introspection and Interpositioning

Introspection: live access to kernel metadata

- Used to provide synthetic basis for trust
- Labeling functions can verify system’s runtime properties

Interpositioning

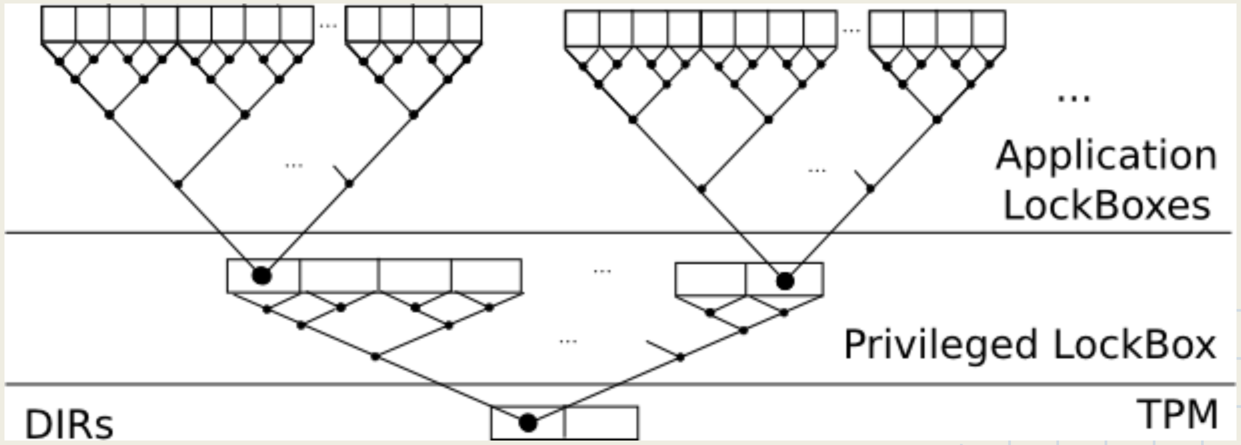
- Sometimes we need to run untrusted code
- Interpositioning allows us to capture and transform (if needed) I/O instructions to enforce some policy
- Can also block IPC, isolating a process from its environment
- Makes an untrusted process trustworthy

Secure Persistent Storage

TPM has very limited onboard secure storage. Nexus multiplexes it with Secure Storage Regions (SSRs).

Confidentiality ensured with CTR AES

Integrity ensured with a Merkle hash tree, with the root stored in the TPM



Storage (con't)

Virtual Data Integrity Register (VDIR)

- kernel abstraction used to hold SSR hashes
- VDIRs stored in hash tree with root in TPM

Virtual Key (VKEY) - used for secure key storage

VDIR and VKEY operations can be protected with logical attestation, so complicated security policies (HIPAA, etc) can be enforced.

NB: Updates to TPM storage are not atomic, so an updated protocol is needed to protect against power failures.

Secure Boot Sequence

Build a hash chain from the BIOS to the Nexus kernel

- Power on: PCRs initialized to known value
- BIOS extends PCRs with firmware hash
- Firmware extends PCRs with bootloader hash
- Bootloader extends PCRs with Nexus hash
- Nexus unseals Storage Root Key (SRK) and restores internal state from disk

Each stage of boot sequence relies on hash of previous stage, so the kernel cannot be subverted by an attacker changing the software stack or stealing the disk and putting it in a different computer.

Does Nexus really solve the TPM’s privacy problem (i.e. exposing the user’s software stack)?

Nexus Applications

Fauxbook

Movie Player

Java Object Store

Not-A-Bot

TruDocs

CertiPics

Protocol Verifiers

Fauxbook

A privacy-protecting social network!

Application developer assured fair share of resources

Users assured data will not leak out of social circle

Developers cannot inspect or data mine user information

Cloud provider must run Nexus OS

Data protections ensured with logical attestation

Streaming Movie Player, Java Object Store, and Not-A-Bot

Movie Player

- Old solution: Content provider requires hash of OS and media player before streaming content
- Nexus solution: Nexus provides certificate showing the media player cannot write to disk or the network

Java Object Store

- Old solution: Dynamically type check objects during deserialization
- Nexus solution: Nexus provides certificate that objects were serialized with a type safe JVM

Not-A-Bot

- Emails sent with certificate from keyboard driver showing that a human wrote the email

TruDocs, CertiPics, and Protocol Verifiers

TruDocs and CertiPics

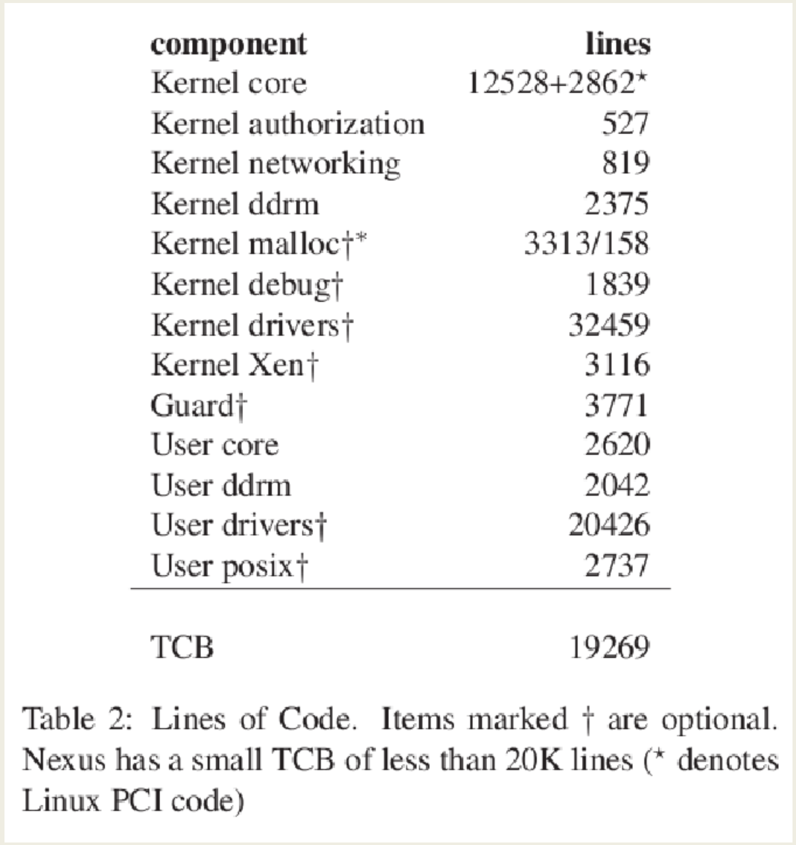
- Document management systems meant to prevent forgery and plagiarism
- CertiPics keeps original image, final image, and a log of changes, allowing a verifier to ensure no policy breech
- TruDocs exports a certificate stating that the new document speaks for another if its quotations follow certain policies

Protocol Verifiers

- Guards ensure that outgoing messages follow certain safety rules

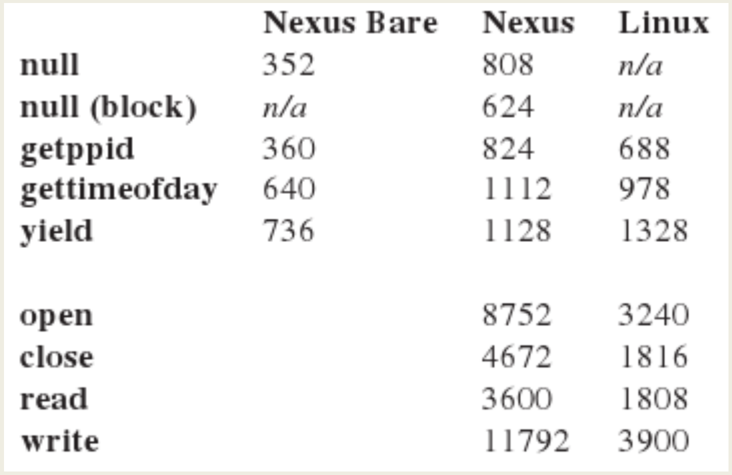
Nexus Evaluation: Microbenchmarks

Size of TCB



component	lines
Kernel core	12528+2862*
Kernel authorization	527
Kernel networking	819
Kernel ddrm	2375
Kernel malloc†*	3313/158
Kernel debug†	1839
Kernel drivers†	32459
Kernel Xen†	3116
Guard†	3771
User core	2620
User ddrm	2042
User drivers†	20426
User posix†	2737
TCB	19269

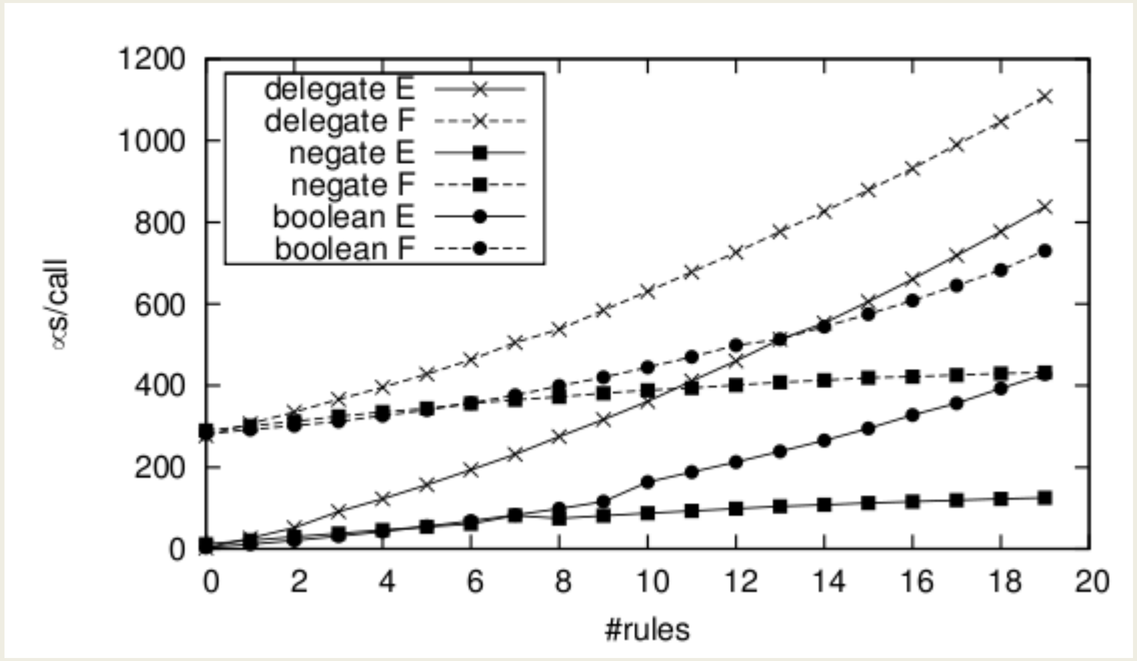
Table 2: Lines of Code. Items marked † are optional. Nexus has a small TCB of less than 20K lines (* denotes Linux PCI code)



	Nexus Bare	Nexus	Linux
null	352	808	*n/a*
null (block)	*n/a*	624	*n/a*
getppid	360	824	688
gettimeofday	640	1112	978
yield	736	1128	1328
open		8752	3240
close		4672	1816
read		3600	1808
write		11792	3900

System Call Overhead

Nexus Evaluation: Proof Evaluation Costs



NB: Most proofs in Nexus have fewer than 15 rules.

Nexus Evaluation: Webserver

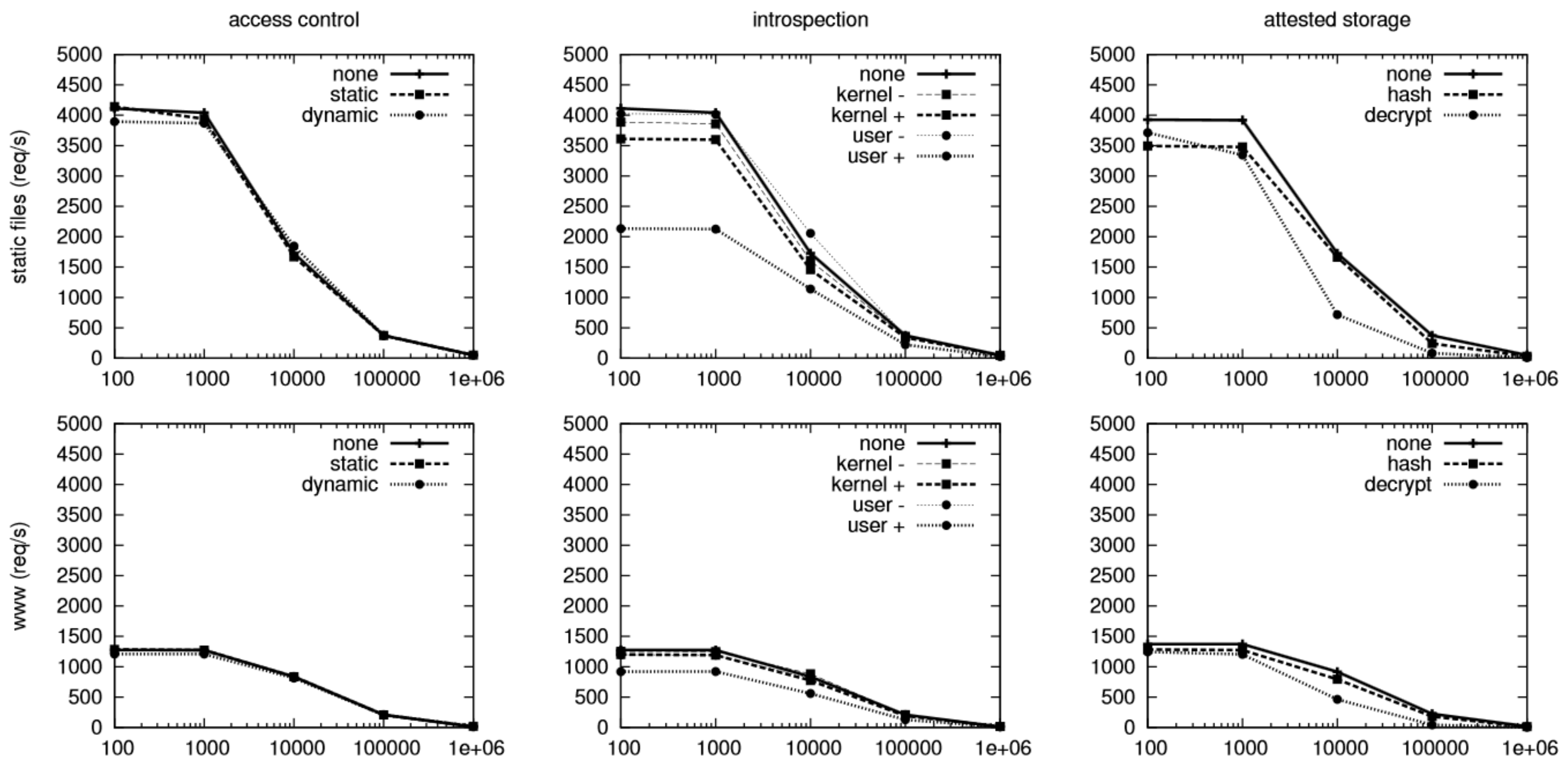


Figure 8: Application evaluation: impact of access control (col. 1), reference monitors (col. 2) and attested storage (col. 3) on a webserver serving static files (row 1) and dynamic Python content (row 2). Filesize varies from 100B to 1MB, the x-axis is plotted in logarithmic scale.

Fabric

Andrew Myers
Owen Arden
Mike George
Jed Liu
K. Vikram
Danfeng Zhang

Fabric Overview

What is Fabric?

- A distributed system for federated storage and computation
- A high-level programming language designed to provide an interface to the above system

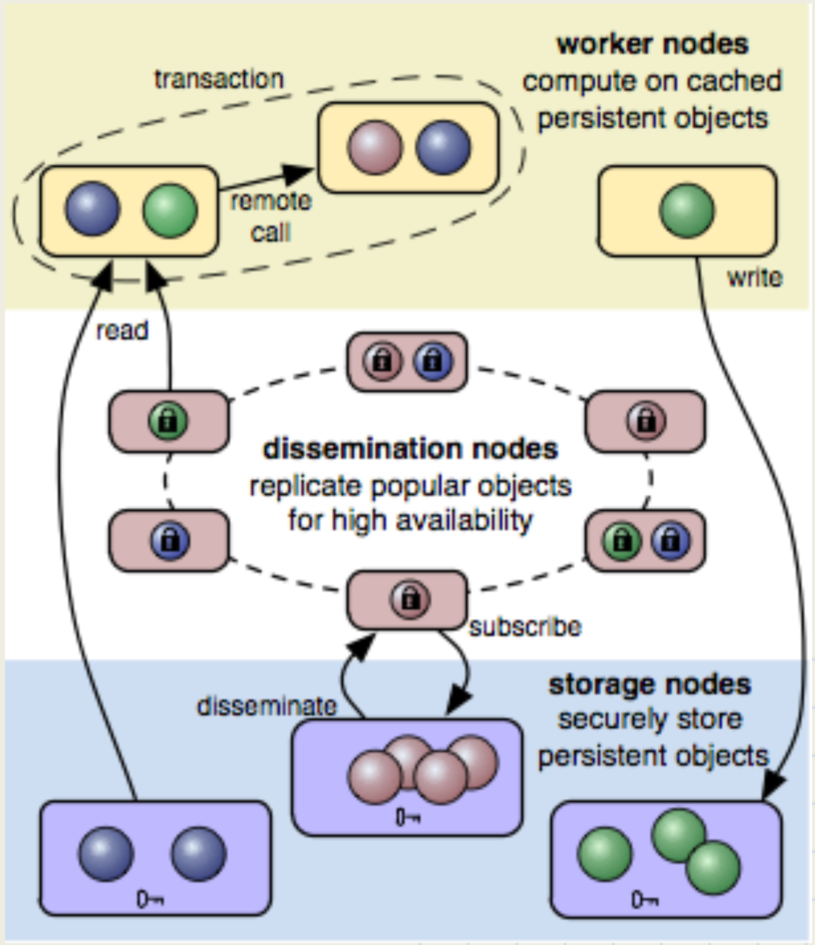
Design goal: secure shared storage and computation between mutually distrusting entities

Fabric Architecture

An unbounded number of networked nodes, both trusted and untrusted.

Three types of nodes

- Storage nodes
- Worker nodes
- Dissemination nodes


worker nodes
compute on cached persistent objects
transaction
remote call
write
read
dissemination nodes
replicate popular objects for high availability
subscribe
disseminate
storage nodes
securely store persistent objects

Fabric Security Model

Principals: authority, privilege, trust

- Examples: users, roles, groups, organization, privileges, Fabric nodes
- Principles can delegate to other principals with the ‘acts-for’ relation (same as Nexus ‘speaksfor’)

Labels

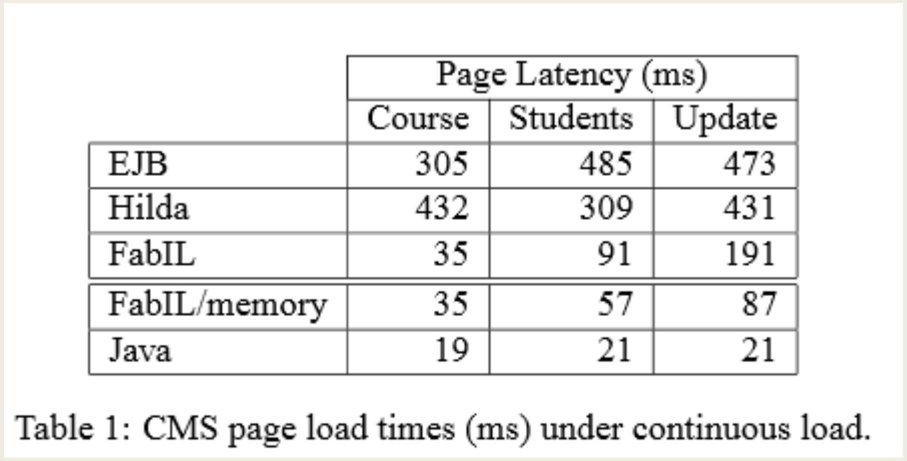
- Carried with objects, state which principles can perform which operations to that object
- Code statically checked at compile time to prevent implicit and explicit flows that violate the policy

Fabric Evaluation

Cornell CMS ported to FabIL.

SIF Calendar, OO7 ported to full fabric language.

End result: porting code is easy, but it's up to an order of magnitude slower. Is this a useful result?



	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.

Comparisons

Confidentiality and integrity:

- Nexus: Trusted IPC through kernel. Persistent state checked against Merkle hash tree.
- Fabric: All network communication over SSL. Persistent state checked against hash. Updates transactional.

Availability

- Nexus: OS kernel makes guarantees about fairly multiplexing resources.
- Fabric: Relies on network's availability guarantees.

Security Models

Nexus: Access control based on credentials and first order logic.

Fabric: Access control based on language features and information flow.

Higher order bits

“Arguably, a large part of designing a secure system is concerned with aligning what must be trusted with what can be trusted.”

- Fred Schneider

Nexus says, “Trust your OS!”

Fabric says, “Trust your compiler!”

Does either approach have an inherent strength or weakness versus the other?

Conclusion

Two approaches to authorization: Nexus Authorization Logic and Information Flow

Both systems

- use synthetic and analytic bases of trust
- are roughly an order of magnitude slower than unsecured systems in the worst case
- require extra sophistication from the programmer

Are these good tradeoffs?

Additional Sources

Nexus OS website: http://www.cs.cornell.edu/people/egs/nexus/index.php

Gun Sirer’s slides on Nexus from SOSP ‘11: http://www.cs.cornell.edu/People/egs/papers/nexus-sosp-slides.pptx

Fabric website: http://www.cs.cornell.edu/projects/fabric/

Principles of Secure Information Flow Analysis, Geoffrey Smith, Chapter 13 (pp. 291-307) of *Malware Detection*, Springer-Verlag, 2007.Smith

Nexus authorization logic (NAL): Design rationale and applications" http://dl.acm.org/citation.cfm?id=1952990