Modern Systems: Security

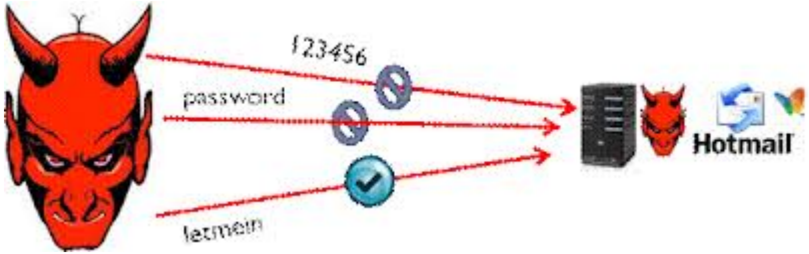
Molly Q Feldman
22 September 2014
Based on slides from CS6410 Fall 2013 and Emin Gun Sirer's SPSOP'11 talk

Outline

- Introduction
- Background
- Nexus
- Fabric
- Trustworthy computing today
- Conclusion

What We've Been Talking About...

- Attacks from Adversaries
  - How to *identify* them
  - How to *contain* them
- Theoretical models
- Some system implementation (Honeyfarms, Vigilante etc.)



What We've Been Talking About...

- Attacks from Adversaries
    - How to *identify* them
    - How to *contain* them
- Theoretical models
- Some system implementation (Honeyfarm, Vigilante etc.)

What if an adversary isn't external?

What are Security Risks?

- External Threats
- Internal Issues
- Bad Code
- Operator Error

Today's Topic

a PL approach and an OS approach to *trustworthy computing*

- how to guarantee the future behavior of applications
- authorization
- trust levels

Two Systems

Nexus

Fabric

Two Systems

Nexus

Fabric

Outline

- ~~Introduction~~
- Background
- Nexus
- Fabric
- Trustworthy computing today
- Conclusion

Trust Establishment

Three techniques for establishing trust:

Axiomatic

- trust by fiat

Analytic

- an analyzer checked and ascertained a property

Synthetic

- an execution environment assures a desired property

Authorization

- We want to authorize actions to maintain security
- Comes down to a simple **if** statement:

> “should this principal be allowed to perform this operation on a resource?”

principal: a user, group, system component, computer etc. that a security system trusts implicitly

Outline

- ~~Introduction~~
- ~~Background~~
- Nexus
- Fabric
- Trustworthy computing today
- Conclusion

Nexus [SOSP'11]

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

OS approach to security, introduces logical attestation

Overview

- Theoretical extension to Trusted Platform Modules
- Logical Attestation
- Nexus OS
- Lots of applications

Trusted Platform Modules

Secure coprocessors provide a unique key and on-board cryptographic functions to capture software state

What can it do?

- Sealed storage
- remote attestation
- platform authentication

Why TPM?

- Cheap!
- Rapidly becoming the standard security model

The Problem with TPM

- Only supports *axiomatic* trust
  - hash-based attestation violates privacy!
  - does not capture dynamic run time state or configuration
  - whitelisting

Credentials-Based Authorization

- attributable property descriptions represented as logical formulas
- every request is accompanied by credentials
  - need *general* mechanisms for capturing them
- access to resources are protected by a guard

Logical Attestation

Credentials: take the form of Nexus Authorization Logic (NAL) proofs

Guard: simple proof checker

Labels ⇔ Credentials

Label is a statement attributed to a principal,
"*P* says *S*"

Logical Attestation cont.

Goal Formulas

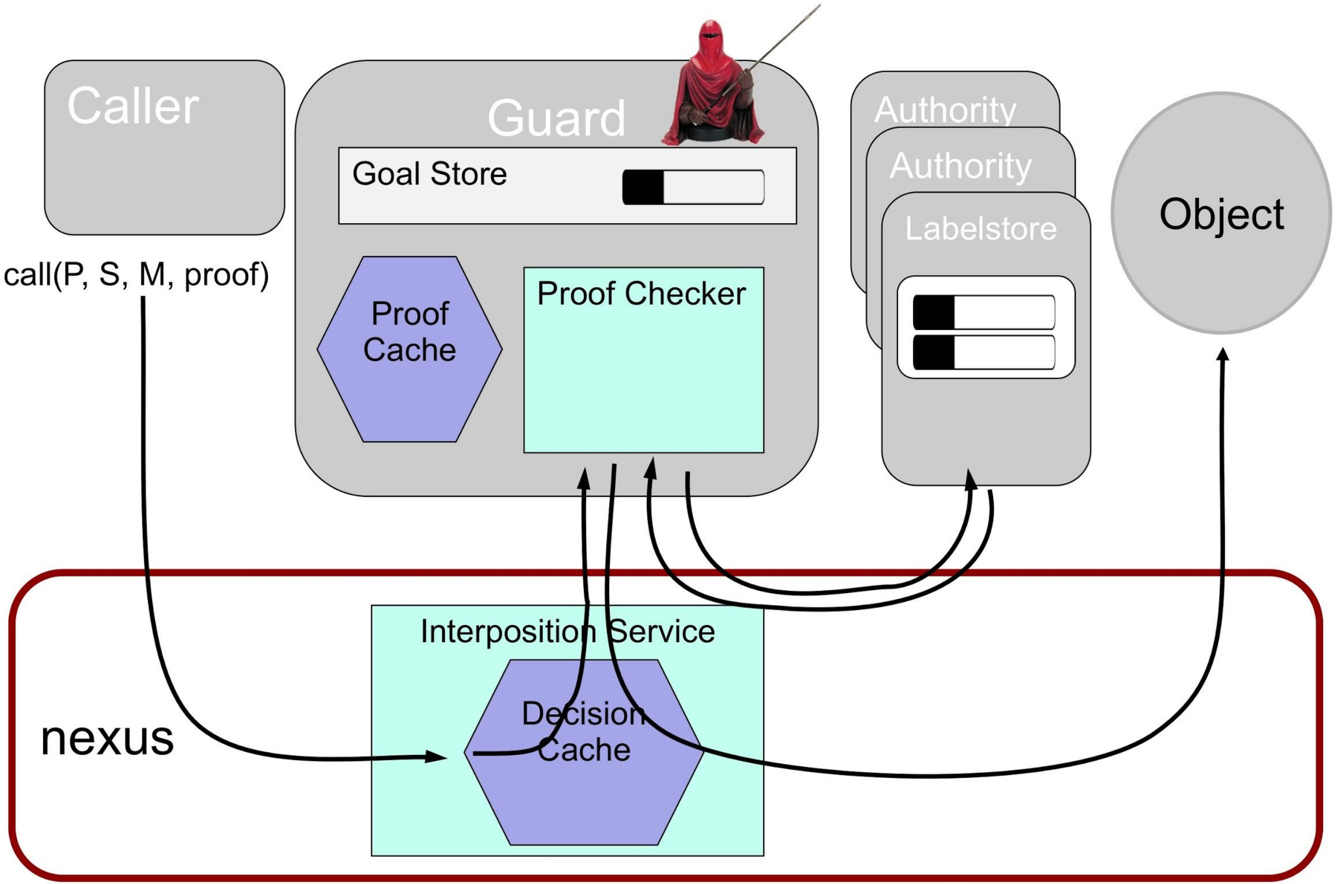
- guard system resources
- satisfied by gathering credentials
- authority is set by goal formulas

Examples

"Owner *says* TimeNow < Sept22"

"Filesystems *says* NTP *speaksfor* Filesystem on TimeNow && NTP *says* TimeNow < Sept22"

Nexus OS


Caller
Guard
Goal Store
Proof Cache
Proof Checker
Authority
Authority
Labelstore
Object
call(P, S, M, proof)
nexus
Interposition Service
Decision Cache

Implementation: Nexus OS

Microkernel architecture

Standard Features (POSIX)

- python
- lighttpd
- sqlite

Additional Features

- *Labels, labelstores, guards, authorities*
- *Introspection*
- *Interposition*
- Secure Persistent Storage
- Secure Bootstrap Sequence

Implementing Logical Attestation Labels

Need to provide *speedups*

- Cryptography is expensive, so Nexus only encrypts labels when exporting
- Invoking guards is expensive, so Nexus caches decisions whenever possible

Introspection & Interpositioning

Introspection

- live access to kernel multidata
- provides *synthetic* trust
- labeling functions verify runtime properties

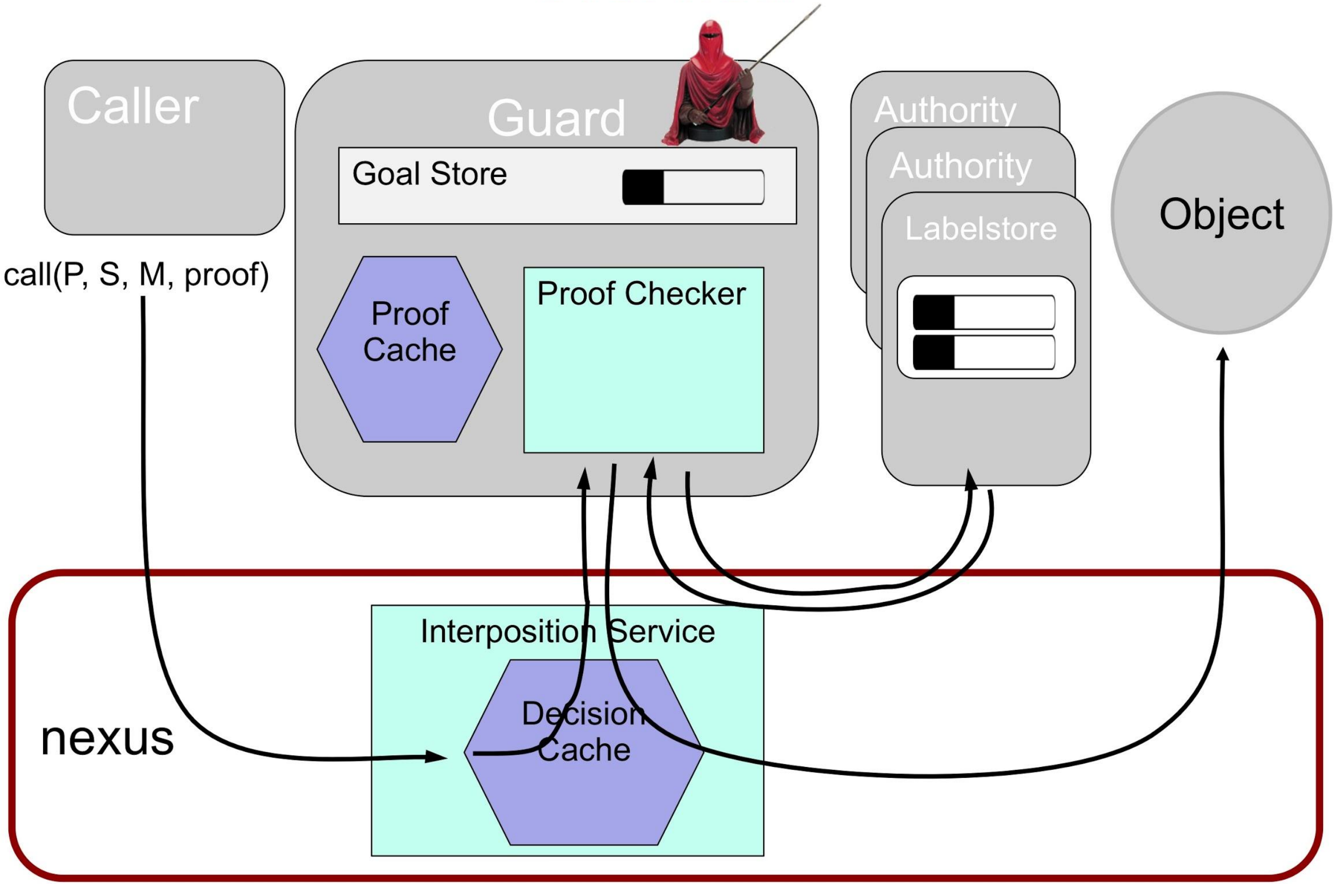
Provides synthetic trust

Interpositioning

- running untrusted code
- allows us to capture and transform I/O instructions
- can block IPC and isolate a process

Makes untrustworthy code trustworthy

Nexus OS



Caller
Guard
Goal Store
Proof Cache
Proof Checker
Authority
Authority
Labelstore
Object
call(P, S, M, proof)
nexus
Interposition Service
Decision Cache

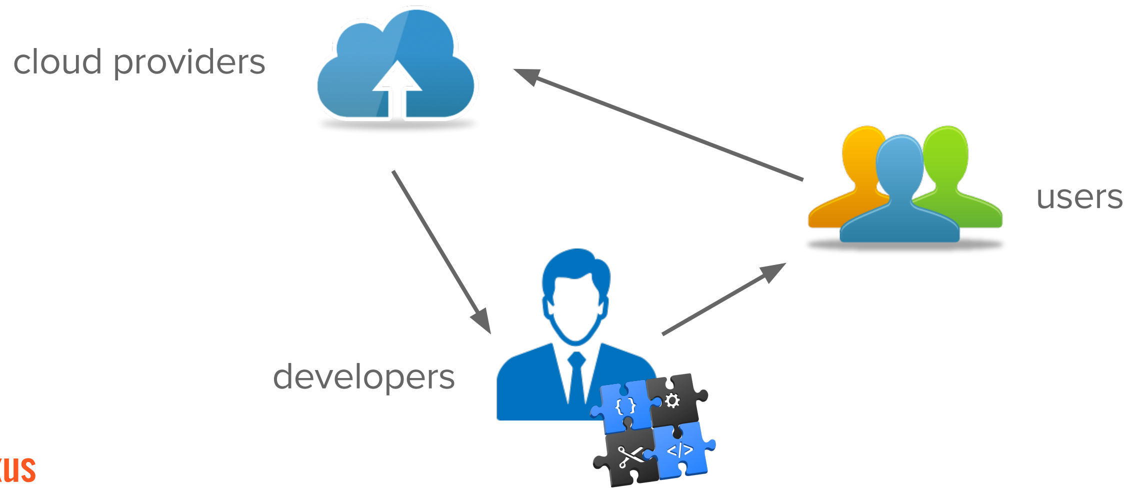
Applications

A LOT of application areas

- Fauxbook
- Movie Player
- Java Object Store
- Not-a-Bot
- TruDocs
- CertiPics
- Protocol Verifiers

Case Example: Fauxbook

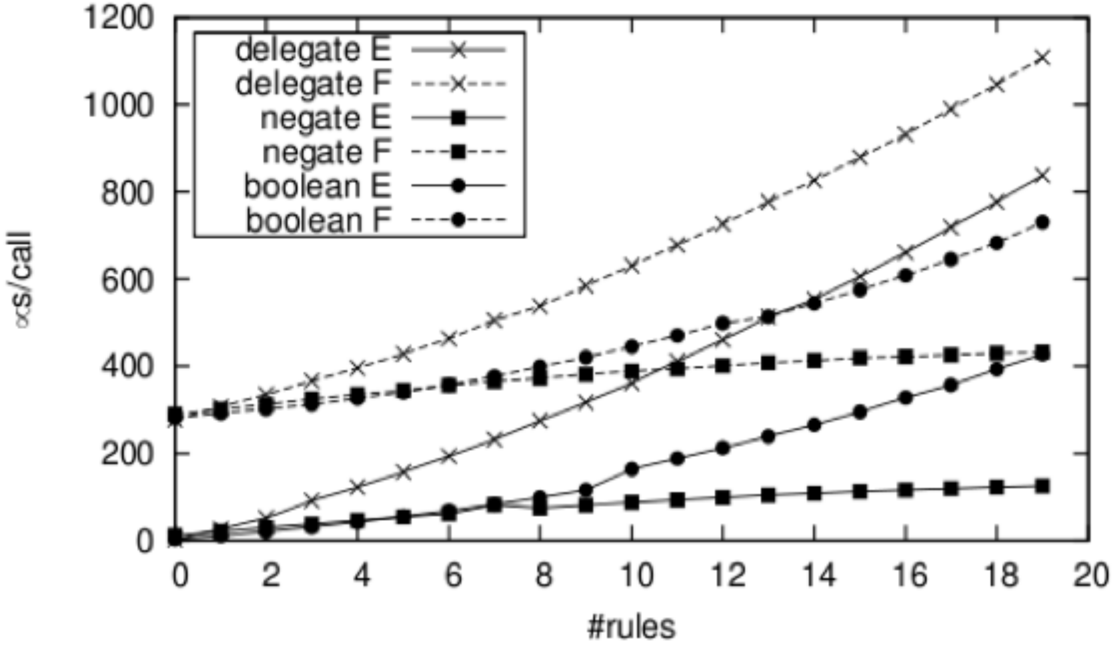
- A privacy-protected social network!



Evaluation Results

	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

Evaluation Results



Most proofs in Nexus have less than 15 rules

Evaluation Results

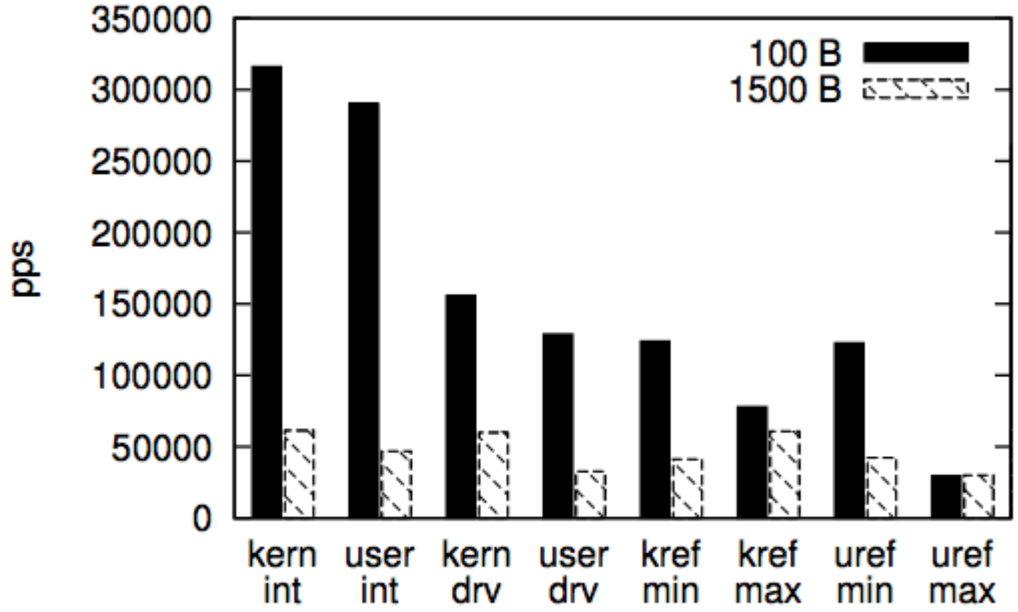


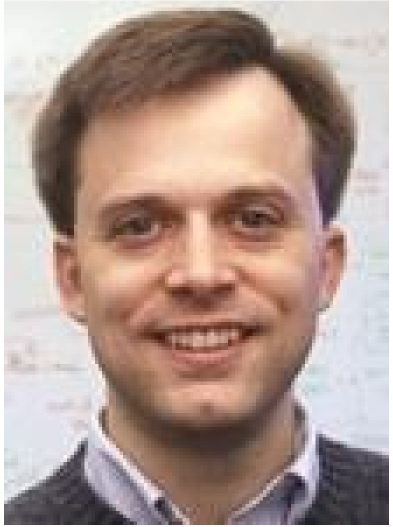
Figure 7: Overhead of interpositioning. Caching decisions decrease packet processing rate by less than 6%.

Outline

- ~~Introduction~~
- ~~Nexus~~
- Fabric
- Trustworthy computing today
- Conclusion

Fabric [SOSP '09]

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



PL approach to secure distributed systems

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

The Big Ideas

Fabric combines many ideas from previous work

- compile-time and run-time information flow
- access control
- peer-to-peer replication
- optimistic transactions

What is Information Flow?

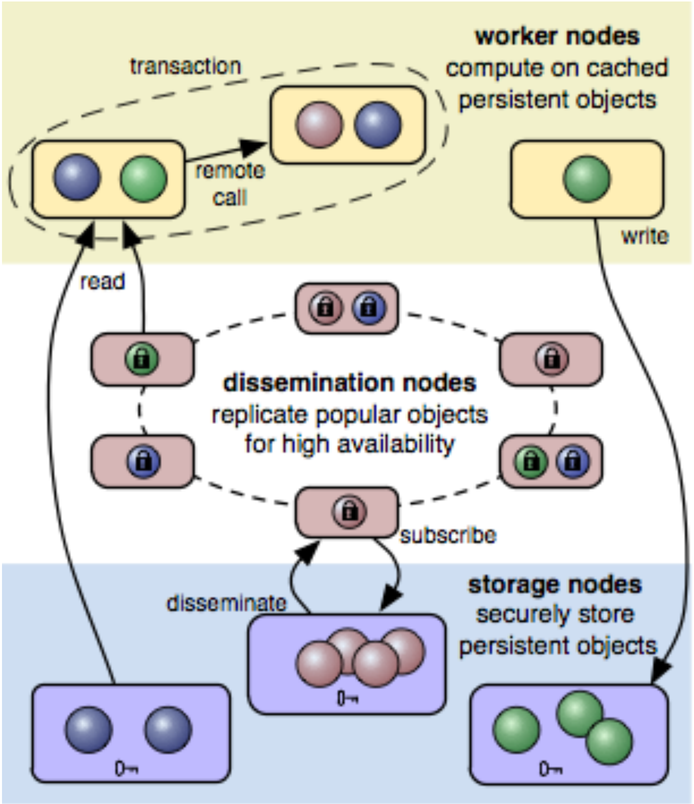
- Information release vs. information propagation
- Security levels and noninterference
- Explicit vs. implicit flows
- Security type systems and static analysis

Fabric Architecture

The Model: an unbounded number of networked nodes, both trusted and untrusted.

Three types of nodes

1. storage nodes
2. dissemination nodes
3. worker nodes



Security Model: Principals

What are they? users, roles, groups, Fabric nodes etc.

What do they do? authority, privilege, trust

How do they interact? they can delegate to other using the *acts-for* relation

Security Model: Principals

What are they? users, roles, groups, Fabric nodes etc.

What do they do? authority, privilege, trust

How do they interact? they can delegate to other using the *acts-for* relation

Like Nexus *speaksfor*

Security Model: Labels

How do we use them? carried with objects and state which principals can perform which operations on that object

How do they help? code is statically checked at compile time to prevent information flow from being violated

Security Model: Labels cont.

What do they preserve? Information flow and trust ordering

What do they look like? (next slide)

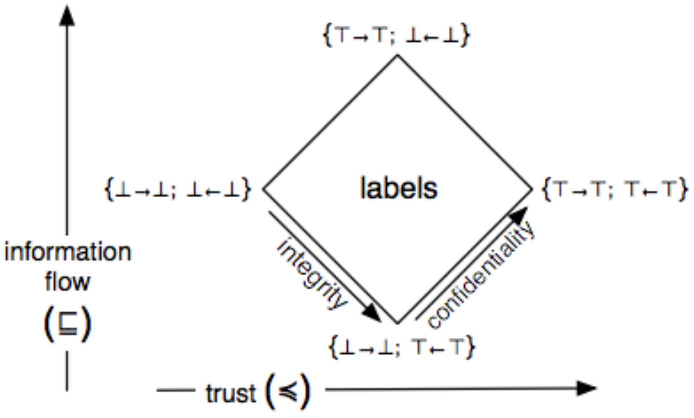


Figure 2: Orderings on the space of labels

Security Model: Labels cont.

```
void m1{alice←} () {
  Worker w = findWorker("bob.cs.cornell.edu");
  if (w actsfor bob) {
    int{alice→bob} data = 1;
    int{alice→} y = m2@w(data);
  }
}

int{alice→bob} m2{alice←} (int{alice→bob} x) {
  return x+1;
}
```

Applications

Not as many as Nexus

- CMS
- SIF (Servlet Information Flow) calendar

Evaluation Results

	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.

Outline

- ~~Introduction~~
- ~~Background~~
- ~~Nexus~~
- ~~Fabric~~
- Trustworthy computing today
- Conclusion

What Happened....

- Fabric “won”
  - overwhelmingly the PL approach became accepted
- Work on Nexus continued (none since 2011)

Fabric Papers

CSF’15, PLAS’14, POST’14, NSDI’14, PLDI’12, Oakland’12, CCS’11, CCS’10, Jed’s Thesis, **SOSP’09**

Outline

- ~~Introduction~~
- ~~Background~~
- ~~Nexus~~
- ~~Fabric~~
- ~~Trustworthy computing today~~
- Conclusion

Two Approaches to Trustworthy Computing

Both Approaches

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

What should we trust?

"Arguably, a large part of designing a secure system is concerned with aligning what must be trusted with what can be trusted."
-Fred Schneider

Trust your OS!
-Nexus

Trust your compiler!
-Fabric