

Wenchao Zhou

Georgetown University

In collaboration with Boon Thau Loo, Andreas Haeberlen, Zachary Ives (Penn), and Micah Sherr (Georgetown)

Introduction

Success of Cloud

- Economics of outsourcing data, computing and management
- □ Virtualization of resources (storage, computing, networking)
- Continued migration of applications to the cloud
 - Amazon EC2, Salesforce, Office 365, iCloud, etc
 - Middleware and firewalls in enterprise networks [SIGCOMM 12]
 - Interdomain routing [HotNets 12]

Increasing interaction between applications/clients

Motivation

Call for Attribution

Needed in tasks with collective efforts

□ Who is responsible for unexpected symptoms?

Attacks, bugs, client-side misbehavior

Evidences for accountability

A Simple Example



A simple task that requires collective effort: routing

- □ System administrator observes strange behavior
- Example: the route to foo.com has suddenly changed

An Ideal Solution



What does attribution look like?

A Data-centric Perspective



We assume a general distributed system

□ A network consisting of nodes (e.g., VMs)

The state of a node is a set of tuples (routes, config, ...)

□ **Idea**: Attribution as reasoning of state dependencies

Base tuples: boundary of the reasoning, considered as *facts*

Provenance for Attribution



Provenance for encoding state dependencies

- Explains the derivation of tuples
- Captures the dependencies between tuples as a graph
- □ Attribution of a tuple is a tree rooted at the tuple
 - Route r1 disappeared as B removes the link between B and C

Challenges



Historical information about distributed state



Historical information about distributed state

Correct and complete provenance in transient state



- Historical information about distributed state
- Correct and complete provenance in transient state
- Distributed maintenance performance tradeoffs

Challenges



- Historical information about distributed state
- Correct and complete provenance in transient state
- Distributed maintenance performance tradeoffs
- Security guarantee in an untrusted environment

Related Work

Provenance for distributed settings

- Cloud systems: PA-S3fs [MMS 10], RAMP [IPW 11]
- Collaborative data sharing systems: Orchestra [GIK+ 07]

Provenance for historical system state

- □ PASS [MHB+ 06]
- workflow systems (Kepler [ABJ 06], VisTrails [CFS+ 06], etc)

Provenance security

□ Sprov [HSW 09], Pedigree [RBT+ 08]

Challenges



- Provenance model (distribution + time)
- Storage and maintenance at large scale
- Distributed provenance querying
- Security guarantees in adversarial environment

Outline

Introduction

□ Motivation: Explain general system anomalies

□ Approach: Secure Time-aware Provenance

Provenance Model [SIGMOD 10, VLDB 13]

- Provenance Maintenance and Querying
- Securing Network Provenance
- Conclusion

State Transition Systems – State



- Node's state captured as tuples
- Message captured as a triplet (src, dest, +/-tuple)
- System state S = (H,M), where H is a set of per-node state, and M is the channel state

Transition Logic as Derivation Rules

State transition in general distributed systems

□ E.g. state machine or event-driven model

□ **Idea**: New state as derivation result of old states

Derivation rules: abstract dependency logic

 $\Box \text{ Example} \quad \tau @ n \Rightarrow \tau_1 @ n_1 \land \tau_2 @ n_2 \land \dots \land \tau_m @ n_m \Rightarrow$ *Rule head Rule body*

Rule head is derived, if all the predicates in rule body hold

Written as Network Datalog (NDlog) rules [LCG+ 06]

Extracting Dependency Logic

Option 1: Inferred provenance Declarative Chord DHT

- Declarative specifications explicitly capture provenance
- □ E.g. Declarative networking, SQL queries, etc.

Option 2: Disclosed provenance Hadoop MapReduce

Modified source code reports provenance

Option 3: Observed provenance Quagga Software Router Defined on observed I/Os of a black-box system

Example: Pairwise Minimal Cost

sp1: pathCost(@S,D,C) :- link(@S,D,C).

sp2: pathCost(@Z,D,C1+C2) :- link(@S,Z,C1),

minCost(@S,D,C2).

sp3: minCost(@S,D,MIN<C>) :- pathCost(@S,D,C).

link(@Src,Dst,C) – "a direct link from node *Src* to *Dst* with cost *C*"

Example: Pairwise Minimal Cost

sp1: pathCost(@S,D,C) :- link(@S,D,C).

sp2: pathCost(@Z,D,C1+C2) :- link(@S,Z,C1), minCost(@S,D,C2).

sp3: minCost(@S,D,MIN<C>) :- pathCost(@S,D,C).

link(@Src,Dst,C) - "a direct link from node Src to Dst with cost C"

pathCost(@Src,Dst,C) - "a path from node Src to Dst with cost C"

Example: Pairwise Minimal Cost

sp1: pathCost(@S,D,C) :- link(@S,D,C).

- \leftarrow One-hop paths
- sp2: pathCost(@Z,D,C1+C2) :- link(@S,Z,C1), ← minCost(@S,D,C2).
- sp3: minCost(@S,D,MIN<C>) :- pathCost(@S,D,C).
- \leftarrow Multi-hop paths
- \leftarrow Aggregation for min cost

link(@Src,Dst,C) – "a direct link from node *Src* to *Dst* with cost *C*"

pathCost(@Src,Dst,C) – "a path from node Src to Dst with cost C"

minCost(@Src,Dst,C) – "best path from node *Src* to *Dst* with minimal cost *C*"

Execution Model

Pipeline Semi-naïve evaluation [LCG+ 06]

sp2a: ΔpathCost(@Z,D,C1+C2) :- Δlink(@S,Z,C1), minCost(@S,D,C2). sp2b: ΔpathCost(@Z,D,C1+C2) :- link(@S,Z,C1), ΔminCost(@S,D,C2).

Rewrite into event-condition-action rules

□ Consume updates, and generate new updates

Execution Traces



Execution trace as an ordered sequences of events

Encode the execution of a state transition system

Provenance Model



Correctness

Provenance should be "consistent" with the trace
 Both are artifact from a system execution

□ Idea: Extract a subtrace from provenance graph

- Extracting subtrace using topological sort
- Edges in provenance graph represents dependencies

Question: how do we define "consistency"

Provenance Properties

Provenance is valid

□ The extracted subtrace should be a viable trace

Provenance is sound

The extracted subtrace has same event orders as actual trace
 Problem: order of concurrent events (no synchronized clocks)
 Idea: per-node perspective (indistinguishable executions)

Provenance is complete

Provenance includes complete explanation of state (changes)

□ **Idea:** state (changes) are reproducible based on provenance

Provenance is minimal

Provenance is exactly the explanation and nothing more

Outline

Introduction

- □ Motivation: Explain general system anomalies
- □ Approach: Secure Time-aware Provenance
- Provenance Model
- Maintenance and Querying [VLDB 13]
- Securing Provenance
- Conclusion

Provenance Maintenance [SIGMOD 10]

Provenance as views of network state [GIK+ 07]

- Maintain in relational tables (prov, ruleExec, send, recv)
- Incremental view maintenance
- □ Pipelined Semi-Naïve (PSN) [LCG+ 06] evaluation

Automatic rewrite of derivation rules

Additionally maintain provenance data

Does NOT affect the scalability of the base protocol



Recursive Provenance Querying

Traversal of the provenance graph

- Step 1: Retrieve rule execution instances
- Step 2: Expand dependent derivations



Recursive Provenance Querying

Traversal of the provenance graph

- □ Step 1: Retrieve rule execution instances
- □ Step 2: Expand dependent derivations

Generic framework for provenance querying

- □ Formulated in declarative networking engine
- Allows customization (annotation defined in provenance semiring [GKT 07]) and optimization (caching, etc)

Performance Tradeoffs

Proactive maintenance

- Provenance deltas deltas between adjacent versions
- Incrementally applied in querying

Reactive maintenance

- □ **Idea**: sufficient data for reconstructing provenance
- Input logs communications and update of base tuples
- Reconstruct provenance by deterministic replay
- □ Long-running systems? Periodic snapshots
- Analogous to log-structured versioning systems

Outline

Introduction

- □ Motivation: Explain general system anomalies
- □ Approach: Secure Network Provenance
- Provenance Model
- Provenance Maintenance and Querying
- Securing Network Provenance [SOSP 11]
- Conclusions

Challenge: Adversaries Can Lie



Problem: adversary can ...

- ... fabricate plausible (yet incorrect) response
 - ... point accusation towards innocent nodes

Threat Model

Existing work

Trusted kernel, monitor, or hardware

• E.g. Backtracker [OSDI 06], ReVirt [OSDI 02], A2M [SOSP 07]

These components may have bugs or be compromised

□ Alternatives that do have require such trust?

No trusted components

- Adversary has full control over an arbitrary subset of the network (Byzantine faults).
 - E.g. Compromised nodes can tamper, drop, or replay information
- Pessimistic threat model gives strong guarantees

Ideal Guarantees

Ideally: explanation is always complete and accurate

Fundamental limitations

□ E.g. Faulty nodes secretly exchange messages

□ E.g. Faulty nodes communicate outside the system

What guarantees can we provide?

Realistic Guarantees [SOSP 11]



- No faults: Explanation is complete and accurate
- Byzantine fault: Explanation identifies at least one faulty node

Securing Cross-Node Edges



Idea 1: Each node keeps vertices about local actions

□ TAP model cleanly partition the provenance graph

Idea 2: Make the graph tamper-evident

□ Secure cross-node edges (evidence of omissions)

Secure Provenance Maintenance

Tamper-evident logs [HKD 07]

- Linear append-only list of events
- Recursively-defined hash chain
- Include top-level hash in messages
- □ Any tampering breaks the chain!



37



Secure Provenance Querying

Recursively construct the provenance graph

- Retrieve secure logs from remote nodes
- Check for tampering, omission, and equivocation
- Replay the log to regenerate the provenance graph



Secure Provenance Querying

Recursively construct the provenance graph

- Retrieve secure logs from remote nodes
- Check for tampering, omission, and equivocation
- Replay the log to regenerate the provenance graph



Secure Provenance Querying

Recursively construct the provenance graph

- Retrieve secure logs from remote nodes
- Check for tampering, omission, and equivocation
- Replay the log to regenerate the provenance graph



NetTrails [SIGMOD-demo 11]

Based on the RapidNet declarative networking engine http://netdb.cis.upenn.edu/rapidnet/

System available for download.



Outline

Introduction

- □ Motivation: Explain general system anomalies
- □ Approach: Secure Network Provenance
- Provenance Model
- Provenance Maintenance and Querying
- Securing Network Provenance

Conclusions

Ongoing and Future Directions

Privacy concerns of provenance

Tension between attribution and privacy

□ Results in Interdomain routing [HotNets 11, SIGCOMM 12]

Better use of provenance data

Provenance-based recovery and damage assessment

 Feedback for invariant refinement. Deduce invariants (desired properties) by mining reported provenance.

Answer why-not questions

Project website: http://snp.cis.upenn.edu/