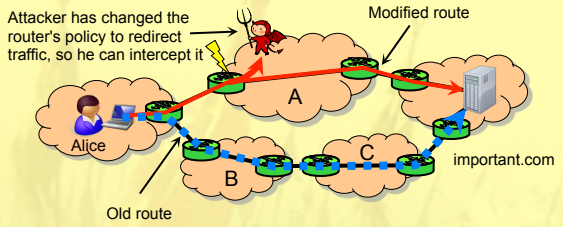
Secure Network Provenance

Andreas Haeberlen* Boon Thau Loo* Micah Sherr# Zachary G. Ives* Wenchao Zhou#
Ang Chen* Alexander Gurney* W. Brad Moore# Arjun Narayan* Hanjun Xiao* Yang Wu* Mingchen Zhao*
*University of Pennsylvania #Georgetown University

CNS-1065130

1 Problem: Secure forensics

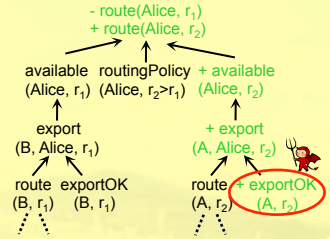


Scenario: Attacker has secretly **compromised** some unknown part of a distributed system

- Affected nodes may now run different software
- Data may be corrupted or destroyed
- Nodes can "tell lies" to confuse the administrators

Goal: Enable the administrators to **detect** and **correctly diagnose** the problem

2 Approach: Data provenance

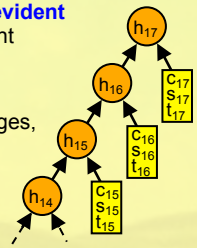


Idea: System should be able to "explain" its own state to the administrator

- Explanation contains the **provenance** of the state (based on concept from databases)
- Provenance should be **tamper-evident**: If the adversary tells lies, we can reliably detect this
- Effect: Misbehaving nodes must give the correct explanation (→discovery) or tell a lie (→discovery)

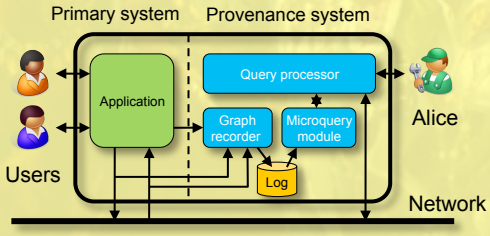
3 Key ideas

- Each node maintains a **tamper-evident log** of all the messages it has sent and received.
- If a compromised node modifies, forges, omits, or reorders messages, this can be detected
- Forensic investigator can **audit** a node's log and **replay** it to reconstruct its execution



- To **extract provenance**, the system can be instrumented; in some cases (declarative languages, 'maybe' rules), extraction can be automated
- Detection can be guaranteed for **observable messages** - that is, messages that directly or indirectly affect at least one correct node
- The investigator must trust his local machine, but otherwise **no trusted components are needed**

4 The SNooPy system

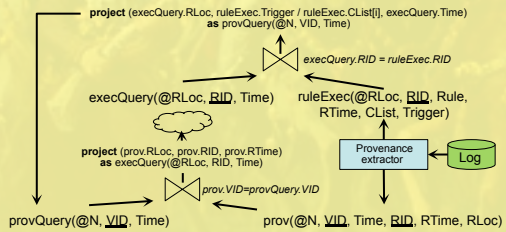


Practical implementation of SNP

- **Widely applicable** - evaluated with BGP inter-domain routing, a DHT, and Hadoop MapReduce
- Detection guarantees **formally proven**
- Reasonable overhead
- Code available from http://snp.cis.upenn.edu/

[TaPP'11, SIGMOD'11 demo, SOSP'11]

5 Storing the provenance



Problem: Store & query provenance efficiently

- Builds a model of the system's workload and automatically chooses **most efficient data structure** to store the provenance
- Can **partially reconstruct** the provenance graph (only the parts that are needed to answer the query)

[TaPP'12, VLDB'13]

6 Generalizing provenance

What if there are **privacy concerns**?
- A special, highly efficient ZKP can help!
- PVR algorithm [HotNets'11, SIGCOMM'12]

What about **negative events** (omissions)?
- There is a negative 'twin' of provenance!
- Negative provenance [HotNets'13, SIGCOMM'14]

Can **covert channels** be detected as well?
- Yes – if we can reliably detect timing anomalies!
- Time-deterministic replay [OSDI'14]

Is this applicable to the **data plane**?
- Reduce crypto cost, use hardware offloading!
- Secure Packet Provenance [submitted]

Can this help with **root-cause analysis**?
- Yes – if we have a non-faulty reference event
- Differential provenance [HotNets'15, SIGCOMM'16]

Can vulnerabilities be **repaired automatically**?
- Need to consider code as well – not just data!
- Meta-provenance [HotNets'15, NSDI'17]