Secure Forensics without Trusted Components

Wenchao Zhou, Qiong Fei, Arjun Narayan, Andreas Haeberlen, Boon Thou Loo, Micah Sherr
*University of Pennsylvania, *Georgetown University

Introduction

Goal: Develop capability to answer diagnostic or forensics questions about network state
- Systems are found to be in an unexpected state
- Determine the causes: why did the route to foo.com change?
- Determine the effects: what other routes have been affected?

Getting correct forensics answers is difficult
- Nodes may be compromised by the attacker
- Fabricate plausible (yet incorrect) response
- Misdirect accusation to innocent nodes
- Existing work relies on trusted components, e.g., OS kernel, virtual machine, monitor, hardware, etc

Tamper-evident provenance (TEP), a forensics system that can operate in a completely untrusted environment
- A novel data structure for forensics in adversarial environments
- Tamper-evident forensics query engine
- Prototype implementation and case studies on various systems

Tamper-evident Query Engine

Architectural overview
- Logging at execution time
- On-demand replay for querying

Provenance store
- Use tamper-evident logging
- Record minimal system state for deterministic replay

Vertex processor
- Fetch the logs and perform deterministic replay
- Generate immediate successors and predecessors

Query processor
- Recursively expand the provenance graph
- Use vertex processor to assemble answers to higher-level queries

Implementation and Case Studies

Three techniques to extract provenance
- M1 - Inferred Provenance: Dependencies are explicitly captured in the implementation (e.g., via the use of declarative language)
- M2 - Reported Provenance: Modified code reports provenance
- M3 - External Specification: Dependencies are defined between observed input and output of black-box applications

Use cases
- Chord DHT (M1): explain finger entries / lookup results
- Hadoop MapReduce (M2): explain suspicious WordCount results
- Quagga BGP (M3): explain routing entry changes / oscillations

Evaluation — secure forensics with reasonable overheads
- Runtime overhead: fixed-size overhead for each message
- Storage overhead: easily fit into commodity hard disks
- Query overhead: up to 70 seconds for provenance querying

Dem o: Interactive Visualization Tool

Provenance graph on a hyperbolic plane
- Focus on the part that users are most interested in
- Smooth transition when the focus changes

Future extensions
- Progressively expand provenance vertices
- Incorporate tamper-evident query engine

Acknowledgments

This work was supported by NSF grants IIS-0812270, CNS-0845552, CNS-1054229, AFOSR MURI grant FA9550 08 1 0352, DARPA award N66001-11-C-4020, and NPS award N00244-11-1-0008.

Figure 1. In the system (left), network A’s policy blocks traffic to Alice, and Alice reaches foo.com through B & C. If Eve compromises A (right), she can change the policy and eavesdrop Alice’s traffic.

Figure 2. Example provenance graph for the bestCost(C,D) tuple in the classic provenance notation (left) and the extended TEP provenance notation (right).

Figure 3. Normalized increase in traffic (left), and per-node log growth (right) excluding checkpoints.