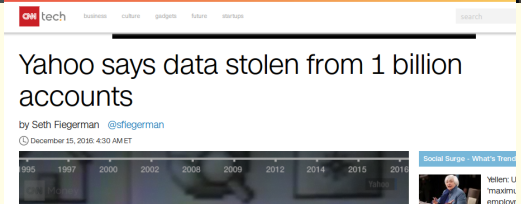
Dependency-Preserving Data Compaction for Scalable Forensic Analysis[1]

Md Nahid Hossain, Junao Wang, R. Sekar, and Scott D. Stoller

Stony Brook University

[1]This work was supported by DARPA (contract FA8650-15-C-7561) and in part by NSF (grants CNS-1319137, CNS-1421893 and CCF-1414078) and ONR (grants N00014-15-1-2208, N00014-15-1-2378 and N00014-17-1-2891).

Long-running Attack Campaigns on the Increase ...



Motivation

- Attacks may be detected a long time after
  - System logs need to be stored for several months or more
  - Log data can be GBs per day, *per host*
  - Enterprises with 1000s of hosts can produce *petabytes* of data per year
- Not just a storage cost: forensic analysis of large data sets can be painfully slow
- Motivates the need for data compaction techniques.

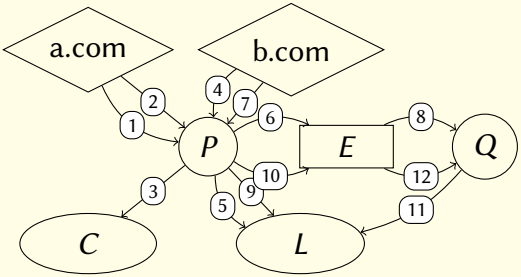
Goals

- Work with readily available system-call audit data, e.g., Windows ETW and Linux audit logs
  - Fine-grained dependency tracking can improve forensic analysis, but such tracking is not ready for large-scale deployment.
- Reduce data volume *without degrading* forensic analysis.
  - Events can be dropped if they don't change (most) forensic analysis results.

Contributions

- Two novel *graph-based* reduction techniques: Full dependence (FD) and Source dependence (SD) preservation.
  - 4.6 to 19x reduction on total events.
- Efficient algorithms and optimizations for implementing reductions
  - Can process/analyze about 1M events per second per core
- Compact data representation techniques for
  - *offline storage:* 35x smaller on average (before compression)
  - *online analysis:* Less than 5 bytes of storage per event in the original data.

Background: Dependence Graphs



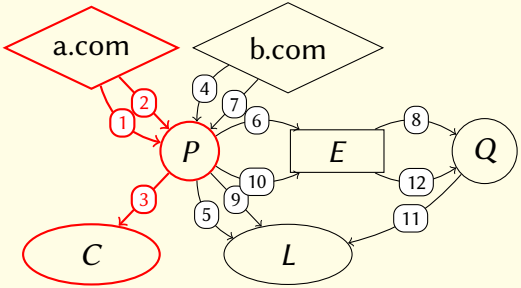
Node: a *subject* (process) or *object* (file/network connection).

Edge: timestamped event, oriented in the direction of information flow.

- e.g., flow from P to C occurred at $t = 3$

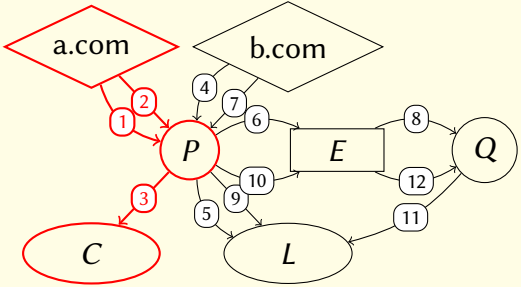
Forensic Analysis

Backward analysis: Given a suspect node (C), trace back to the entry point of attack (a.com)

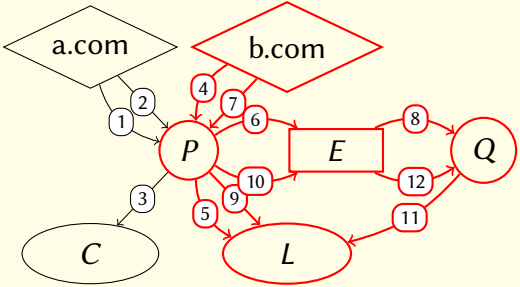


Forensic Analysis

Backward analysis: Given a suspect node (C), trace back to the entry point of attack (a.com)



Forward/Impact analysis: Identify all nodes potentially compromised by a suspect node (b.com).



Reduction Techniques

Reduction Techniques

Log-based:

- Examine a sequence of events
- Determine if some of them can be eliminated without altering dependencies.

Graph-based:

- Construct a dependence graph
- Use graph reachability to prune redundant events
- Can leverage *global context* for more powerful reductions

Log-based reduction: Run-Merging

Merge successive events if they are identical

(1) A reads F1

(3) A reads F1
(4) B writes F2

(6) B writes F2

Log-based reduction: Run-Merging

Merge successive events if they are identical

(1) A reads F1

(4) B writes F2

Log-based reduction: Run-Merging

Merge successive events if they are identical

(1) A reads F1
(2) B writes F2
(3) A reads F1
(4) B writes F2
(5) A reads F1
(6) B writes F2

Limitation: thrown off by intervening events. (No reduction possible.)

Log-based reduction: Xu et al [CCS 2016]

Continue merging identical events if intervening events are on different objects and subjects.

(1) A reads F1
(2) B writes F2
(3) A reads F1
(4) B writes F2
(5) A reads F1
(6) B writes F2

Log-based reduction: Xu et al [CCS 2016]

Continue merging identical events if intervening events are on different objects and subjects.

(1) A reads F1

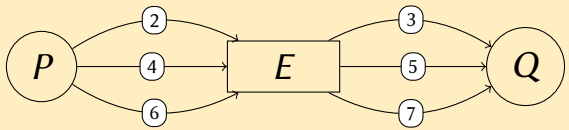
(2) B writes F2

Reductions are now restored.

Log-based reduction: Limitations

Lack of global context makes it difficult to safely merge events in several common scenarios, e.g., pipes, log-files, etc.

Communication via pipe

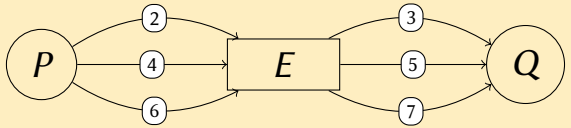


Xu et al cannot merge (4) with (2) due to outgoing edge from E at (3).

Our Approach: Full Dependence (FD) Preservation

Continue to merge identical events if intervening events don't alter *global dependence* of the event's source, e.g., node P below.

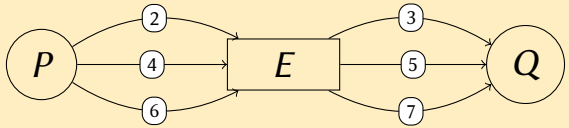
Before reduction



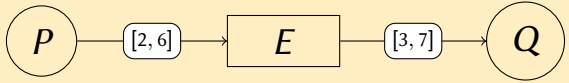
Our Approach: Full Dependence (FD) Preservation

Continue to merge identical events if intervening events don't alter *global dependence* of the event's source, e.g., node P below.

Before reduction



After reduction

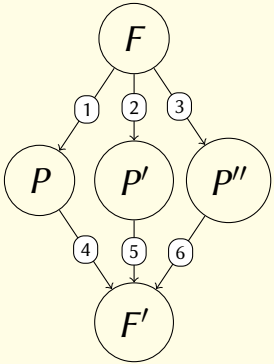


Source Dependence (SD) Preservation

- Takes advantage of how forensic analysis is typically done.
  - Backtrack from suspect node to a source node ("entry point"), then perform forward analysis
- Need only preserve dependencies from source nodes
  - i.e., nodes with no incoming edges
- More powerful than FD: can drop all edges that *don't* introduce new dependencies on source nodes

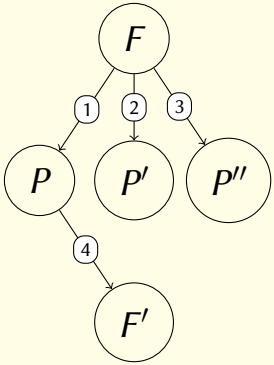
Source Dependence (SD) Preservation

- Takes advantage of how forensic analysis is typically done.
  - Backtrack from suspect node to a source node ("entry point"), then perform forward analysis
- Need only preserve dependencies from source nodes
  - i.e., nodes with no incoming edges
- More powerful than FD: can drop all edges that *don't* introduce new dependencies on source nodes



Source Dependence (SD) Preservation

- Takes advantage of how forensic analysis is typically done.
  - Backtrack from suspect node to a source node ("entry point"), then perform forward analysis
- Need only preserve dependencies from source nodes
  - i.e., nodes with no incoming edges
- More powerful than FD: can drop all edges that *don't* introduce new dependencies on source nodes



Optimizations for Efficient Implementation

Efficient Implementation

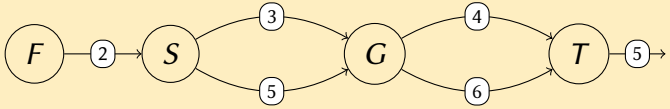
- Global context is more powerful, but is expensive to compute on *timestamped graphs*
- Reachability is a function of time, unlike standard graphs
- Dependencies cannot be cached and reused

Solution: Convert to a standard graph by *versioning* all nodes.

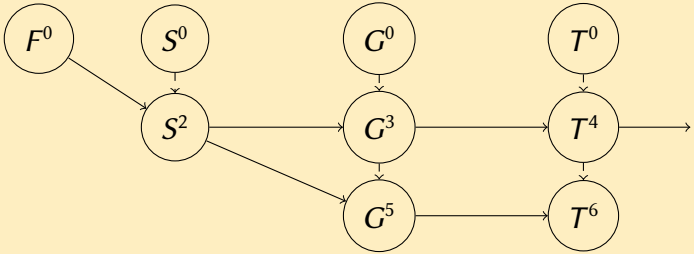
- Develop optimizations that reduce number of versions by 20x
  - On average, just 1.26 versions per object or subject

Versioned Graph

Timestamped Graph



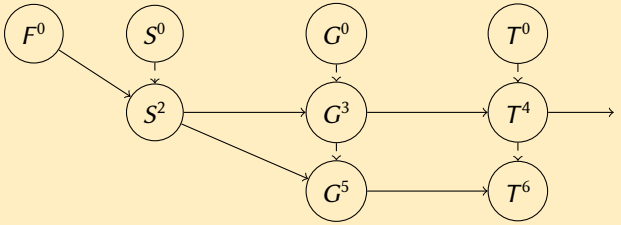
Naive Versioned Graph



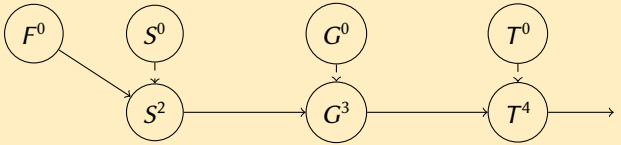
Redundant Edge Optimization (REO)

Discard edge from u to v if there is already an edge from the latest version of u to some version of v.

Naive Versioned Graph



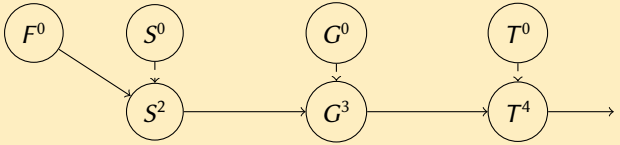
After Applying REO Optimization



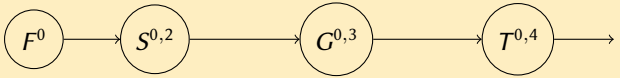
Redundant node optimization (RNO)

Skip new version generation for sink nodes (i.e., nodes with out-degree zero)

Versioned graph with REO optimization



After applying RNO optimization



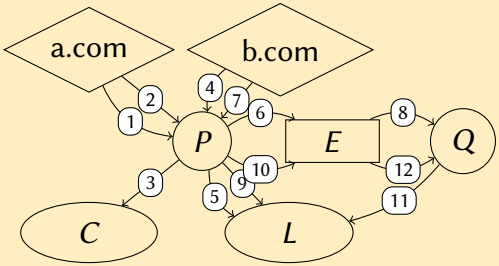
Cycle-Collapsing Optimization (CCO)

Before adding an edge from a version u^r to v^s, if there is a cycle involving u and v, then:

- Combine the two nodes into a "supernode" representing an equivalence class
- Discard future edges between combined nodes

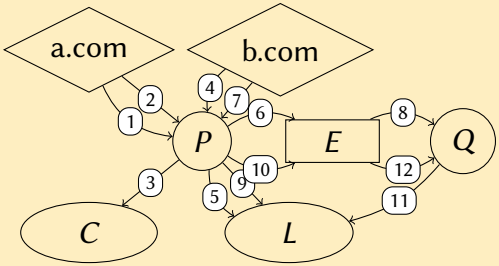
Optimized Versioned Graph

Timestamped Graph

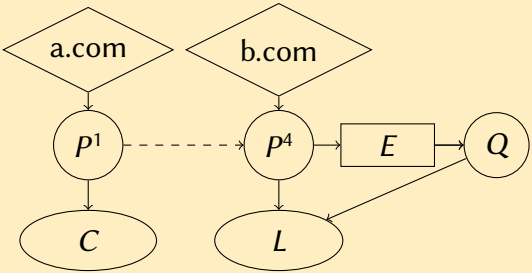


Optimized Versioned Graph

Timestamped Graph



Optimized Versioned Graph



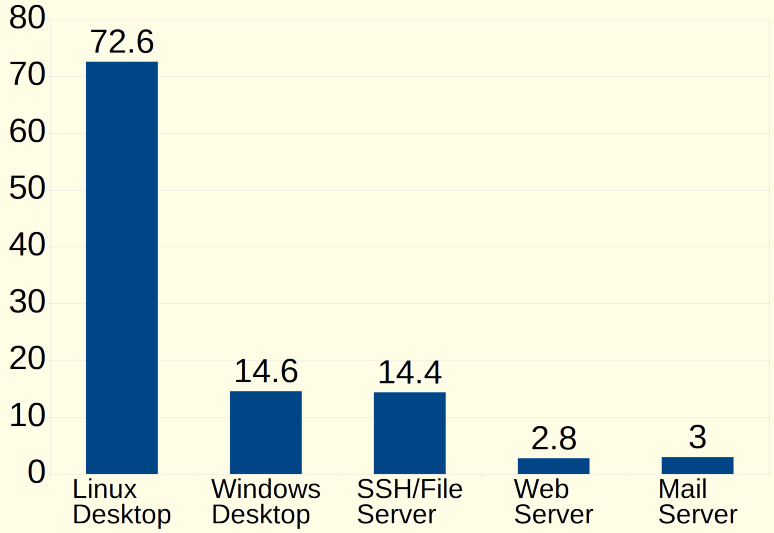
Experimental Evaluation

Evaluation Dataset

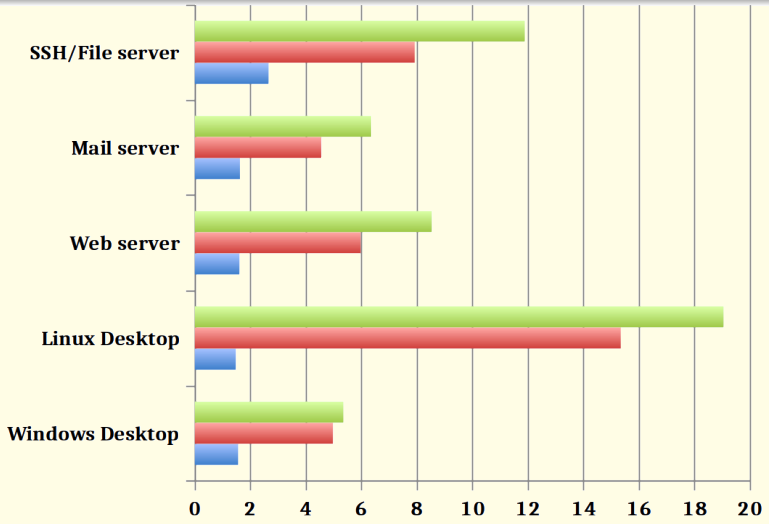
- Dataset from DARPA Transparent Computing Program's 2^{nd} adversarial engagement
  - Data from Windows and Linux desktop systems over 7 days.
  - Benign background activity, plus attacks involving full APT life-cycle.
- Data from our lab servers (Linux).
  - Benign use over 7 days.

Dataset	Size
Linux Desktop	12.9 GB
Windows Desktop	2.1 GB
SSH/File Server	6.7 GB
Web server	1.3 GB
Mail server	1.2 GB

Data Size (Millions of events)



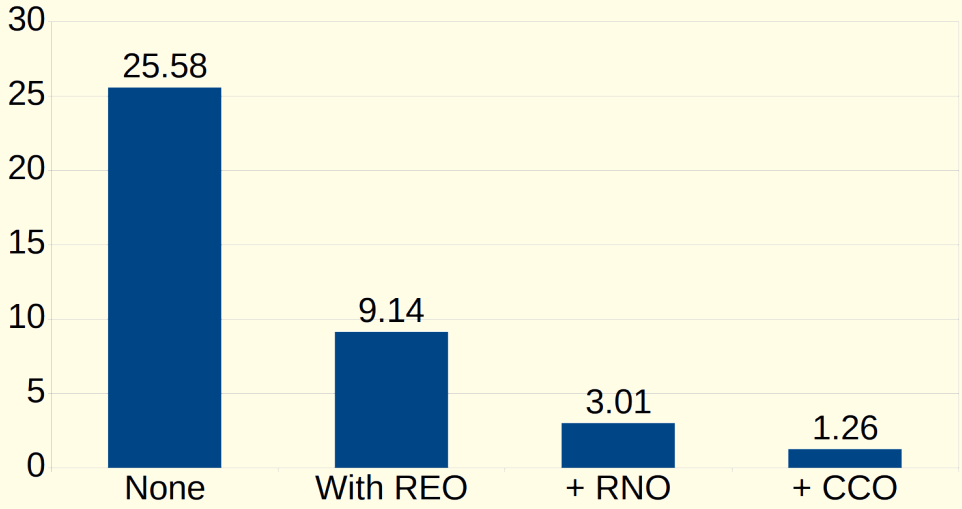
Reduction Factor



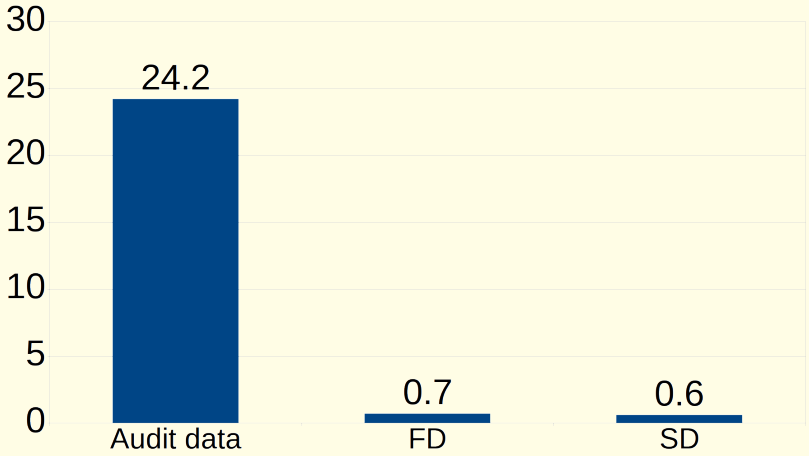
Average:

- Xu et al: 1.8x
- FD: 7.8x
- SD: 10.2x

Average No. of Versions per node



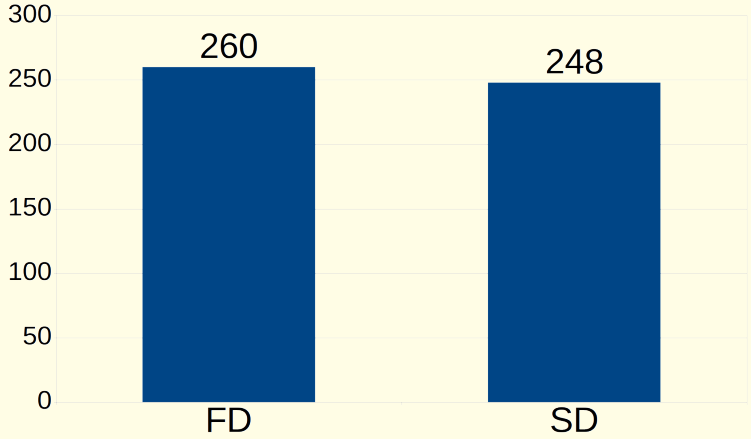
Total Size on Disk (GB)



Factor of decrease:

- FD: 35.3x
- SD: 41.4x

Dependence Graph Size (MB)



Total memory use across all 5 data sets (about 100M events originally)

(Preliminary) Results on DARPA 3rd Engagement

- Dataset about **10x** larger
- *Linux Desktop:* **800M** events, reduced by **70x** using FD.
- *Windows Desktop:* **70M** events, reduced by **10x** using FD.
  - Data size and several other differences between the two auditing systems contributed to the large difference in reduction rates.
- *Summary:* Larger data sets offer more opportunities for reduction
  - Long running processes experience extended periods without new dependencies.
  - During these periods, FD eliminates most operations.

Impact of Reduction on Forensic Analysis

Do reductions interfere with forensic analysis?

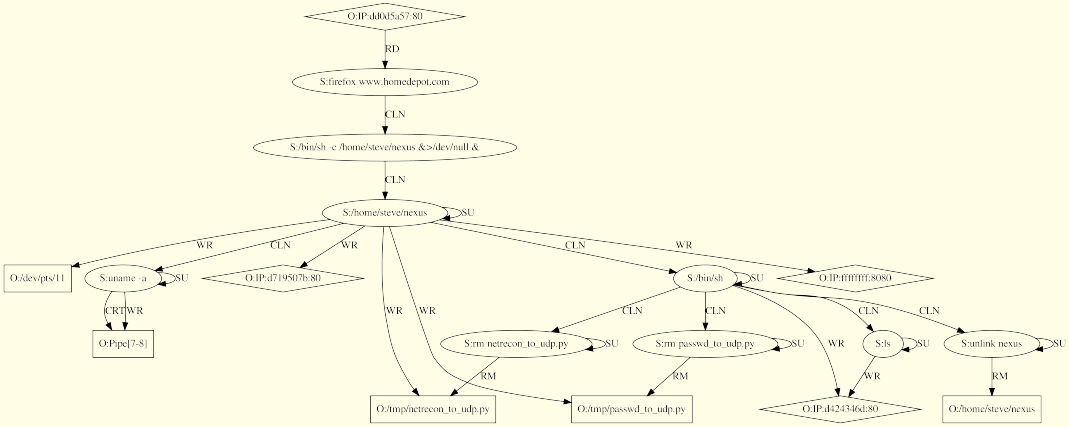
In theory: We prove that

- FD preserves forward/backward analysis results for all nodes.
- SD preserves forward/backward analysis results for source nodes.

In practice: We evaluated during the DARPA adversarial exercise

- We had no prior knowledge about attacks
- Performed forensic analysis in real-time on FD-reduced dependence graph stored in main memory
- Analysis triggered when our SLEUTH [Sec 2017] system flagged a suspicious event.

Example Graph Constructed after Analysis



Forensic analysis with and without Reductions

Dataset	Attack Scenario	Analysis Type	Number of Entities		
			Base	FD	SD
Linux Desktop	A	Backward	7	7	7
		Forward	15	15	15
	B	Backward	3	3	3
		Forward	10	10	10
Windows Desktop	A	Backward	4	4	4
		Forward	17	17	17
	B	Backward	2	2	2
		Forward	9	9	9
	C	Backward	4	4	4
		Forward	7	7	7

Related Work

- *LogGC* [CCS 2013], *ProTracer* [NDSS 2016], *MPI* [Sec 2017]:
  - Specialized for their fine-grained flow tracking system
  - We target coarse-grained auditing, which can be deployed in large enterprises
- *Xu et al* [CCS 2016]: Log-based reduction uses only local context, is unable to achieve the kind of reductions we get using global context.
- Others
  - Provenance collection techniques (e.g., PASS [ATC 2006]) incorporate run-merging, plus cycle detection/avoidance techniques.
  - *ProvWalls* [TOIT 2017] describes reductions for MAC systems.
  - Many other related works discussed in the paper.

Summary

- Developed highly effective reduction techniques for audit logs.
  - FD and SD achieve 5x to 19x reduction in the number of events
- Developed efficient algorithms and optimizations for FD and SD
  - Our techniques can process and analyze about 1M events per second per host.
- Presented compact data representation techniques to achieve over an order of magnitude reduction in storage and main memory sizes.

Summary

- Developed highly effective reduction techniques for audit logs.
  - FD and SD achieve 5x to 19x reduction in the number of events
- Developed efficient algorithms and optimizations for FD and SD
  - Our techniques can process and analyze about 1M events per second per host.
- Presented compact data representation techniques to achieve over an order of magnitude reduction in storage and main memory sizes.

Thank you! Questions?