

# Understanding Audit Logs: Techniques, Experiences, and Requirements

Liang Zhenkai 梁振凯 and Zeng Jun 曾俊



# Security Incidents Are on The Rise

TECH | TWITTER | CYBERSECURITY

## Months la boggles m

Some of the biggest  
By Jay Peters | @jaypeters | D

US & WORLD | TECH | HEA

## 1.5 millic Singapo

Local media say  
By James Vincent | Jul 20



### SINGHEALTH

**DATA WAS TAKEN?**  
ERIC NUMBER,  
GENDER, RACE  
DATE OF BIRTH OF  
ON PATIENTS  
OF OUTPATIENT  
DISPENSED TO  
1,000 PATIENTS

**DATA WAS NOT TAKEN?**  
AND TEST RESULTS  
PHYSICIANS' NOTES  
WERE NOT AMENDED  
DELETED

CHANNEL NEWSASIA

**What** happened? **Who** is affected? **How** to prevent?

# Endpoint Monitoring Solutions

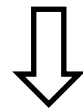
Endpoint monitoring solutions record **audit logs** for attack investigation



## Audit logs:

- A history of events representing OS-level activities
- Provide visibility into security incidents with data provenance

```
type=SYSCALL msg=audit(30/09/19 20:34:53.383:98866813) : arch=x86_64  
syscall=read exit=25 a0=0x3 ppid=15757 pid=30204 auid=junzeng sess=6309
```

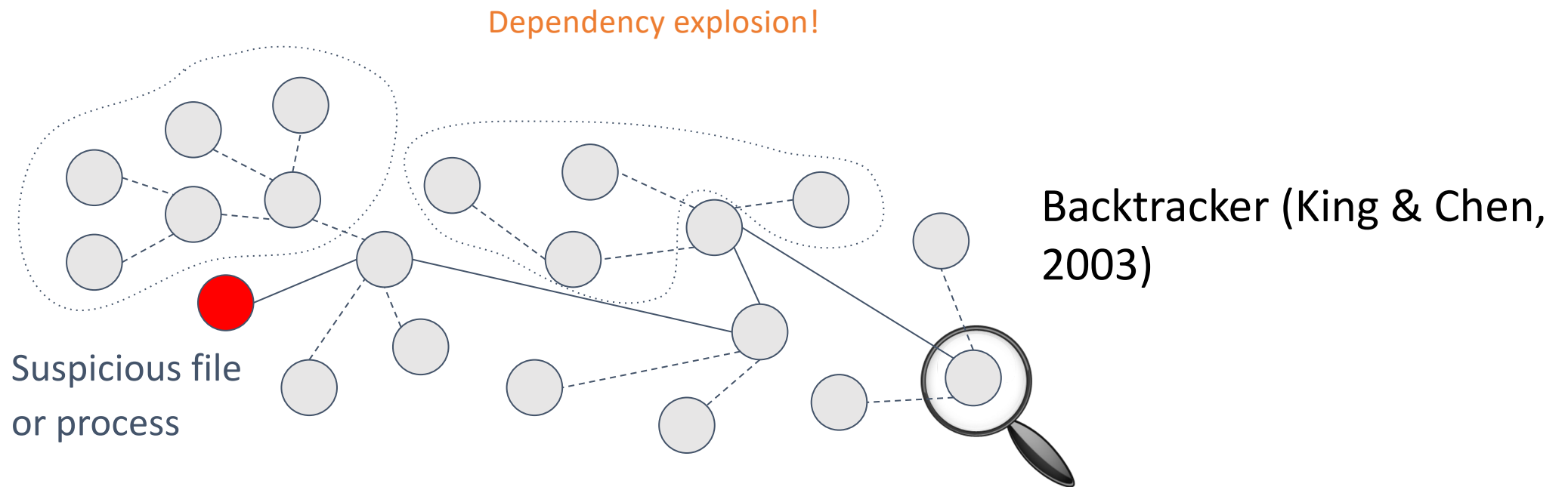


Provenance Analysis



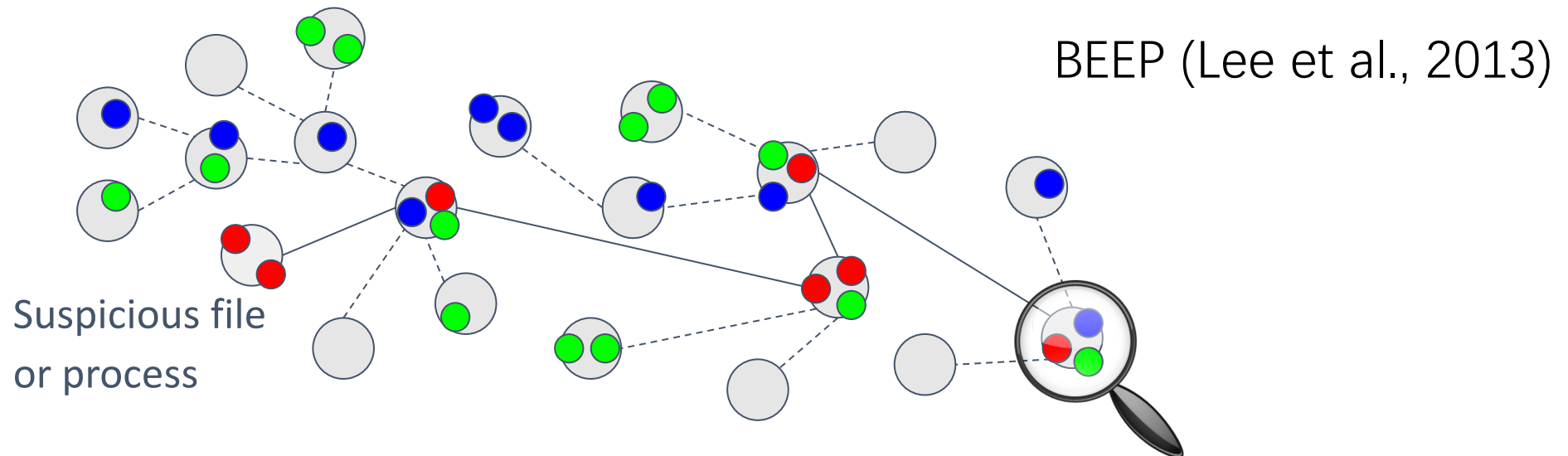
# Audit Log Analysis

- Starting from a detection point, *Backtracker* does:
  - Events & objects identification related detection point
  - Generate dependency graph
  - Use rules to prune unrelated nodes in the dependency graph



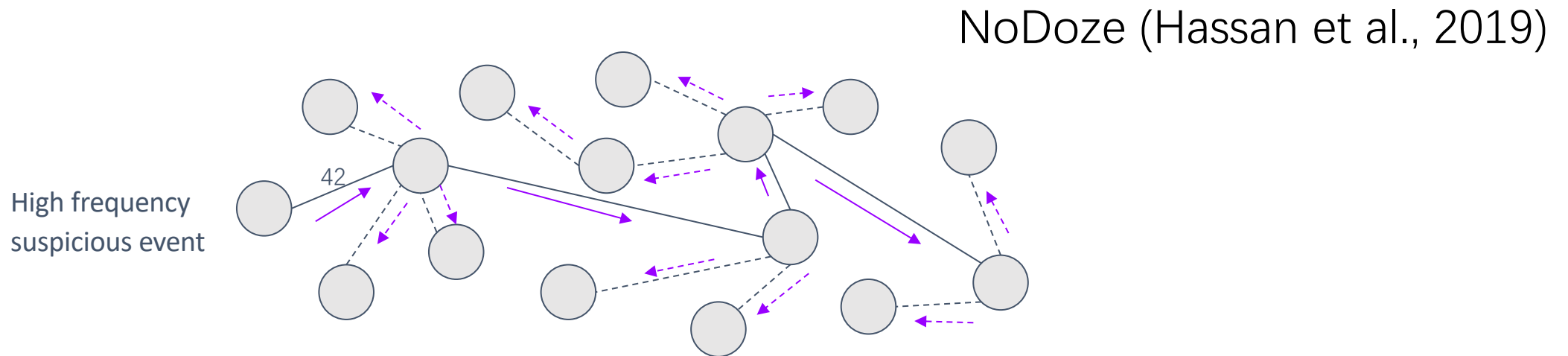
# Audit Log Analysis

- Resolve *dependency explosion* problem in a long running application
  - Fine-grained provenance tracing technique
  - Identifying unit boundaries & dependences
  - Partition into individual *unit*
  - Code instrumentation



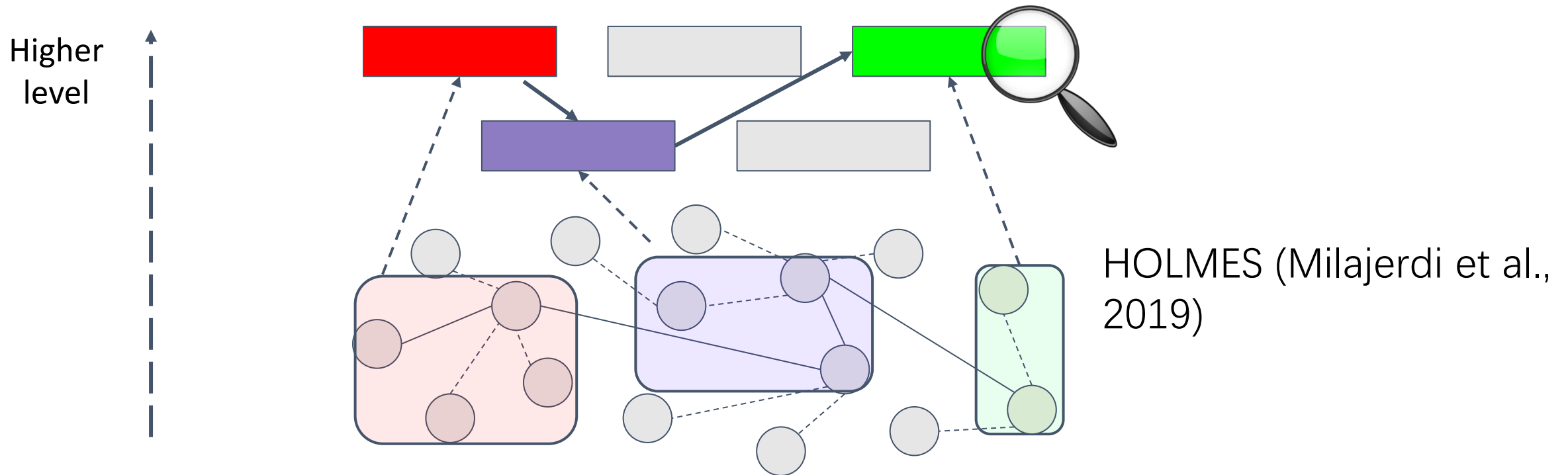
# Audit Log Analysis

- Address *threat alert fatigue* during threat investigation
  - Assign anomaly scores to every edge in dependency graph
  - Based on frequency of events that have occurred (historical & contextual information)
  - Propagated score through edges in the graph
  - Generate aggregated anomaly score for triaging



# Audit Log Analysis

- Generate high-level graph during threat investigation
  - Develop robust & reliable detection signal
  - Correlate between suspicious information flow



# Related Work

- Scale up provenance analysis:
  - Data reduction [NDSS'16, 18 ...] & Query system [Security'18, ATC'18 ...]

Can we automatically **abstract** high-level behaviors from low-level audit logs and **cluster** semantically similar behaviors before human inspection?

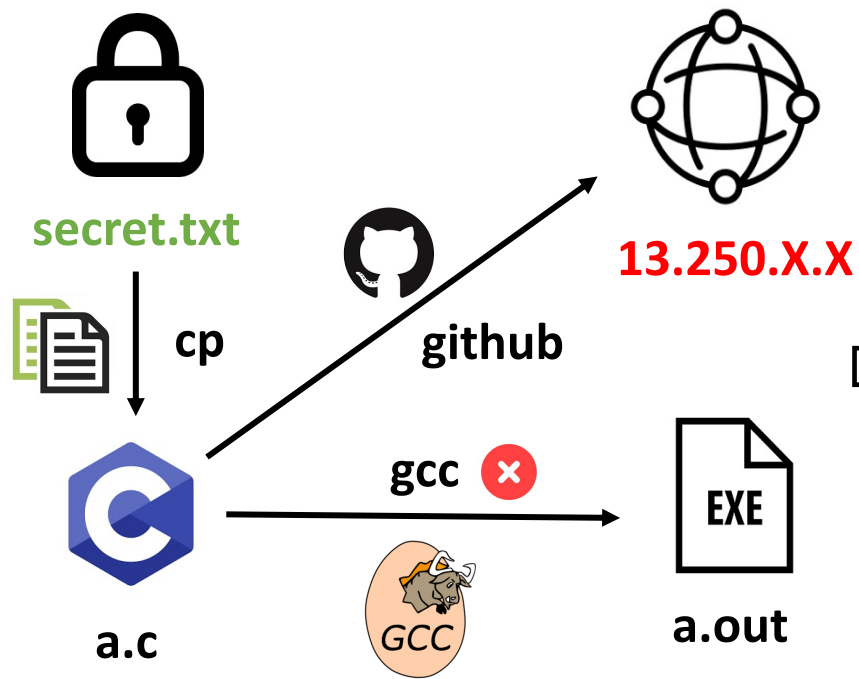
- Query graph [VLDB'15, CCS'19], Tactics Techniques Procedures (TTPs) specification [SP'19,20], and Tag policy [Security'17,18]

Behavior-specific rules heavily rely on domain knowledge (**time-consuming**)

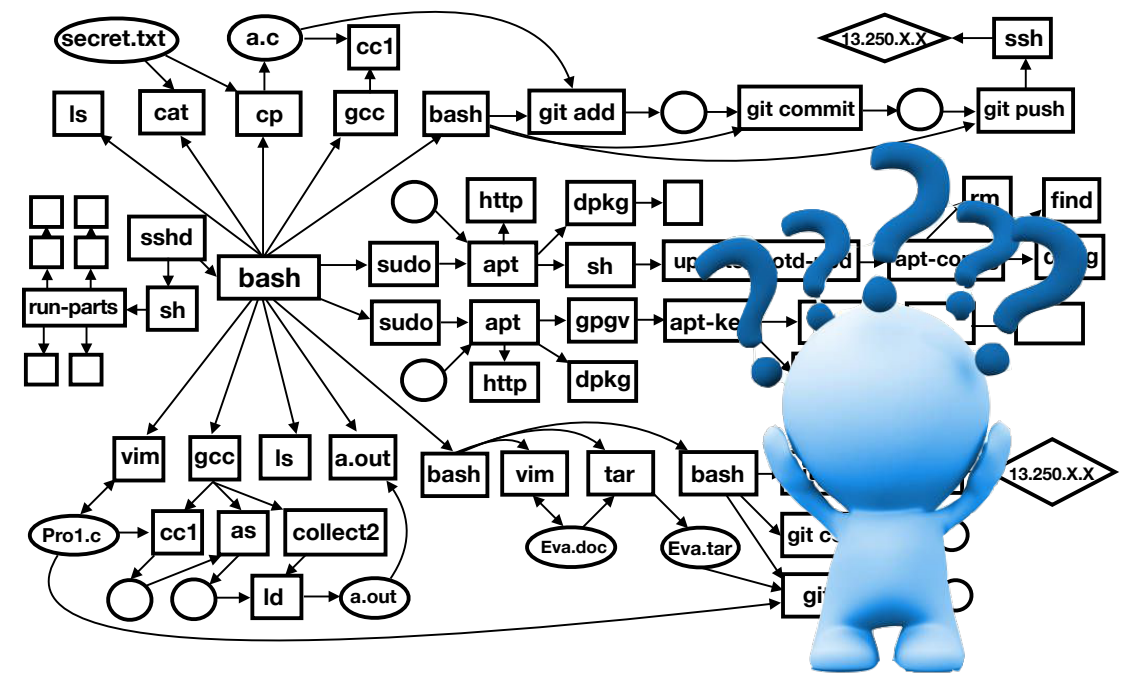
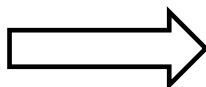


# Motivating Example

Attack Scenario: A software tester **exfiltrates sensitive data** that he has access to



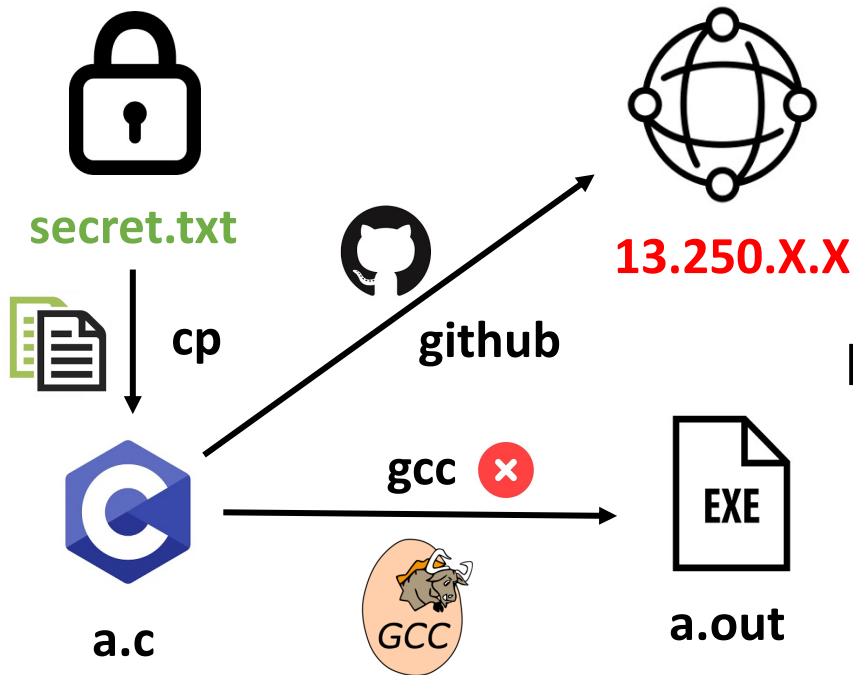
Data Exfiltration Steps



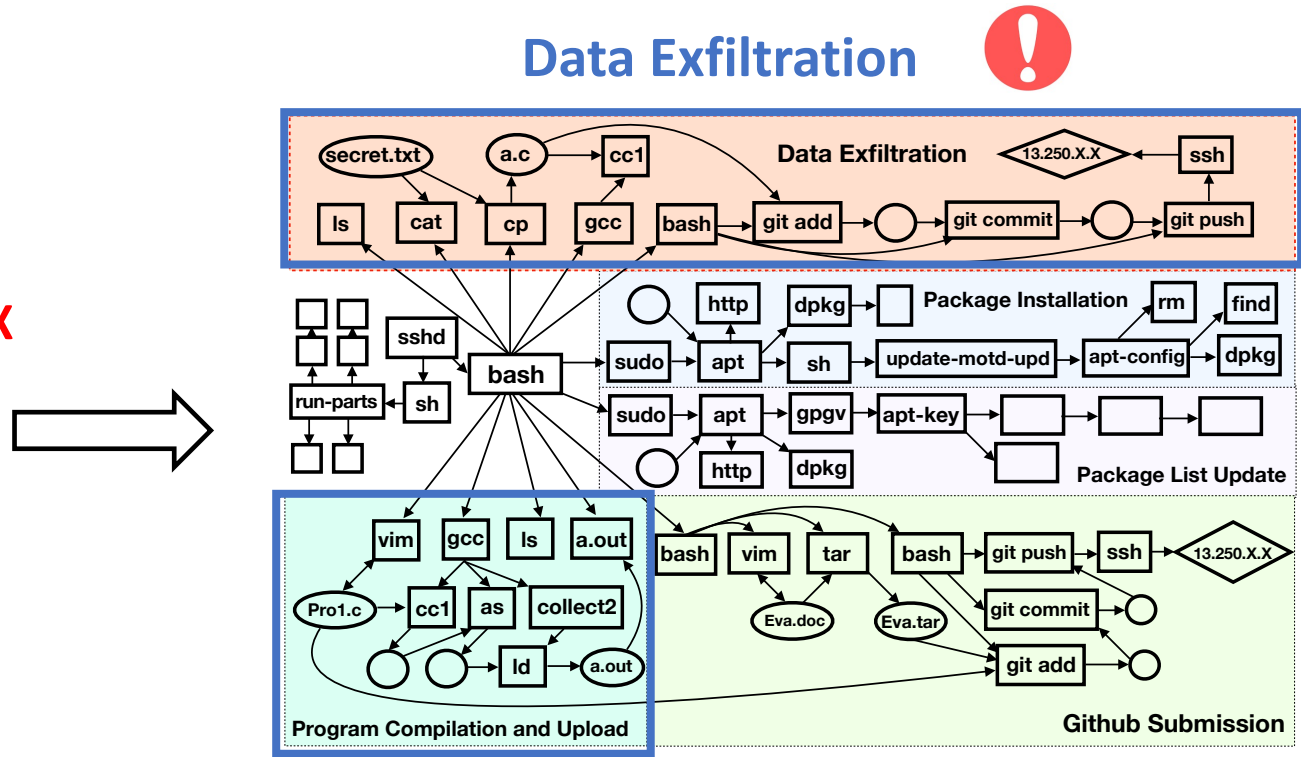
Motivating Example Logs

# Motivating Example

Attack Scenario: A software tester **exfiltrates sensitive data** that he has access to



Data Exfiltration Steps



Program Compiling and Upload (cluster)

Motivating Example Logs

# Challenges for Behavior Abstraction

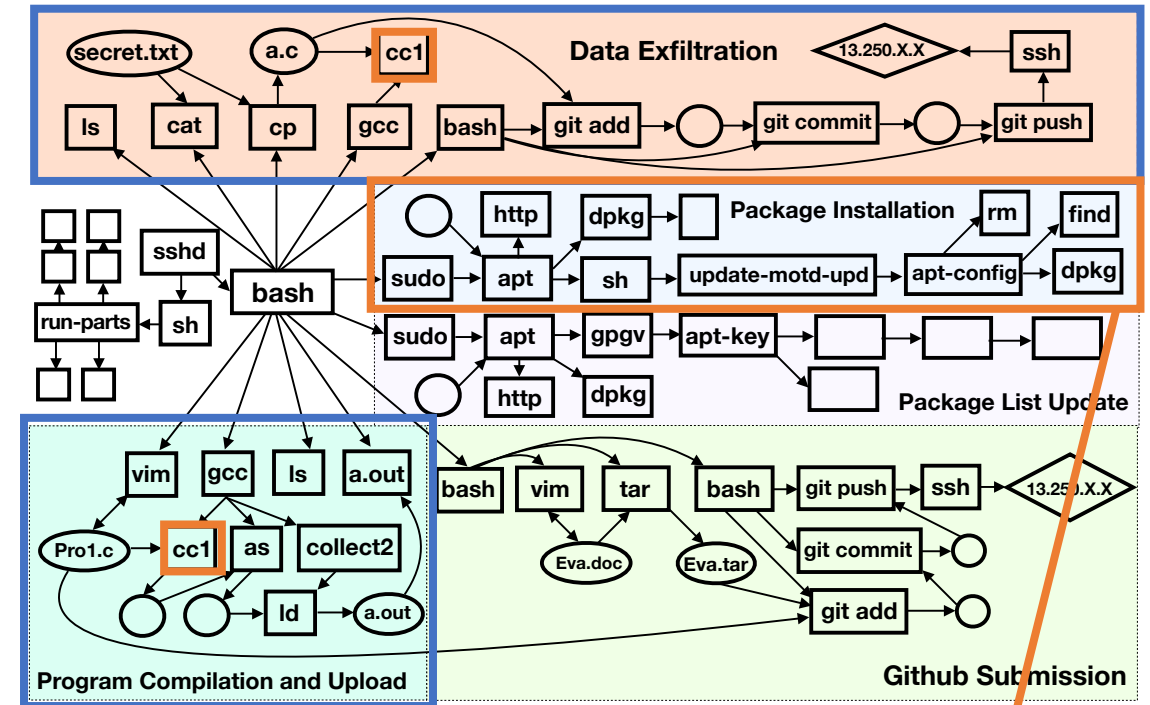
## Data Exfiltration

### Event Semantics Inference:

- Logs record **general-purpose** system activities but lack knowledge of **high-level semantics**

### Individual Behavior Identification:

- The volume of audit logs is **overwhelming**
- Audit events are **highly interleaving**

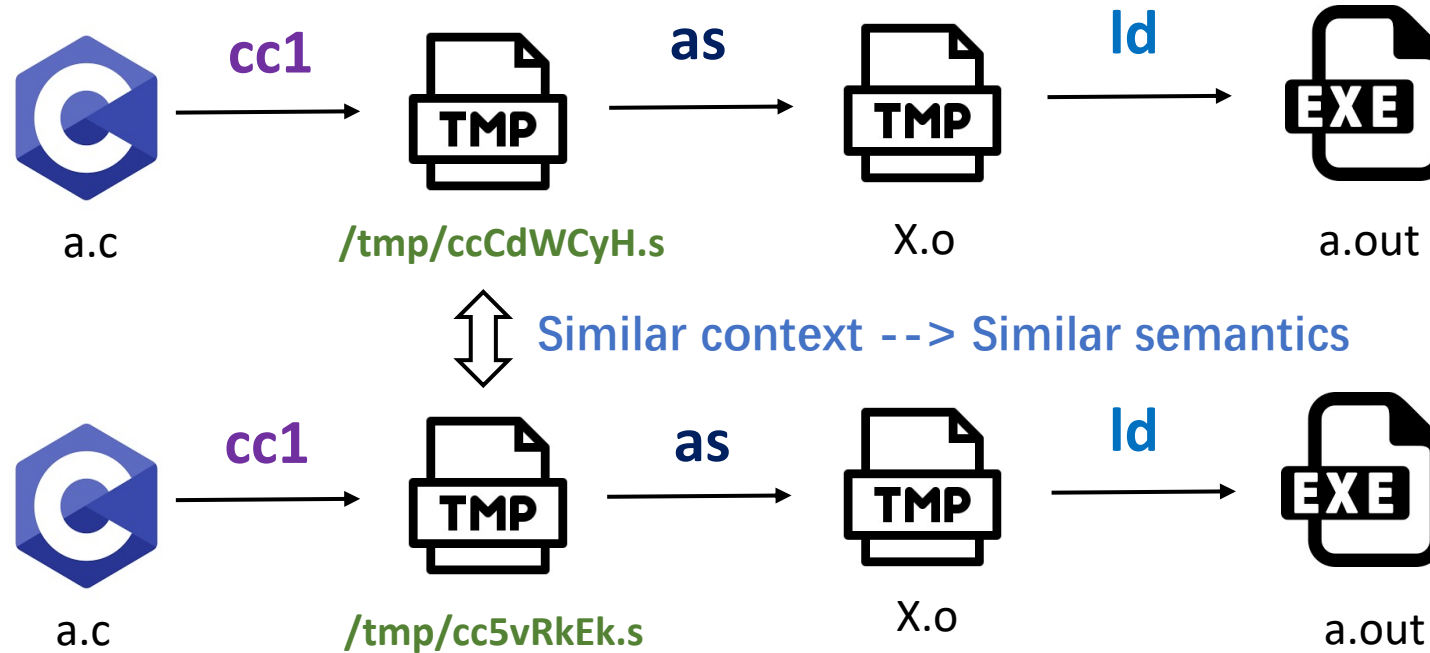


## Program Compiling and Upload

Package Installation Events > 50,000

# Our Insights

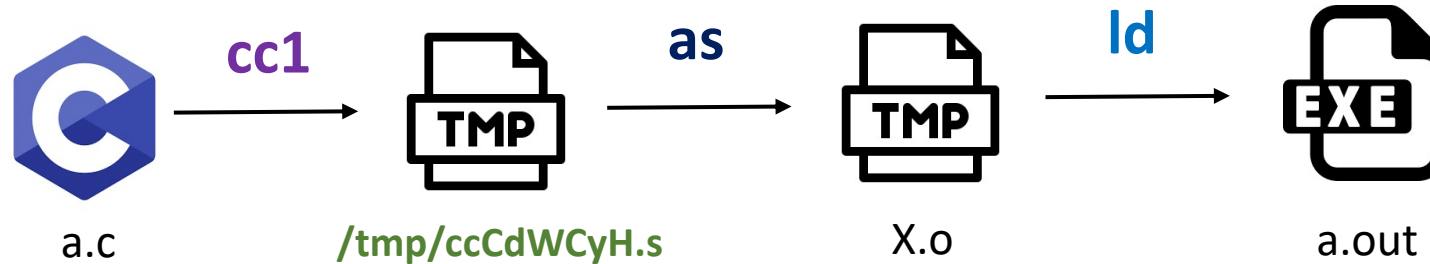
How do analysts manually interpret the semantics of audit events?



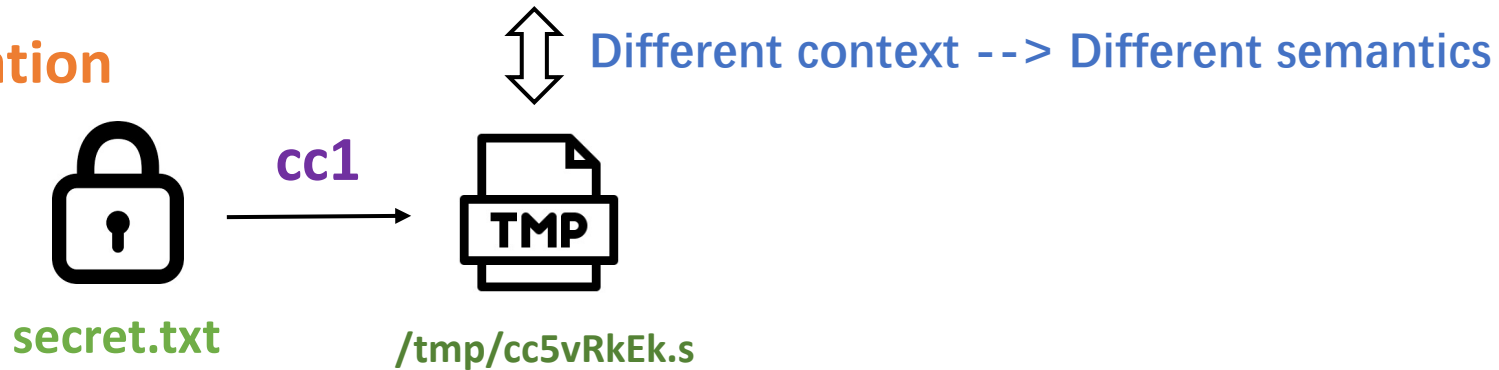
Compiling program using GCC

# Our Insights

How do analysts manually interpret the semantics of audit events?



## Data Exfiltration

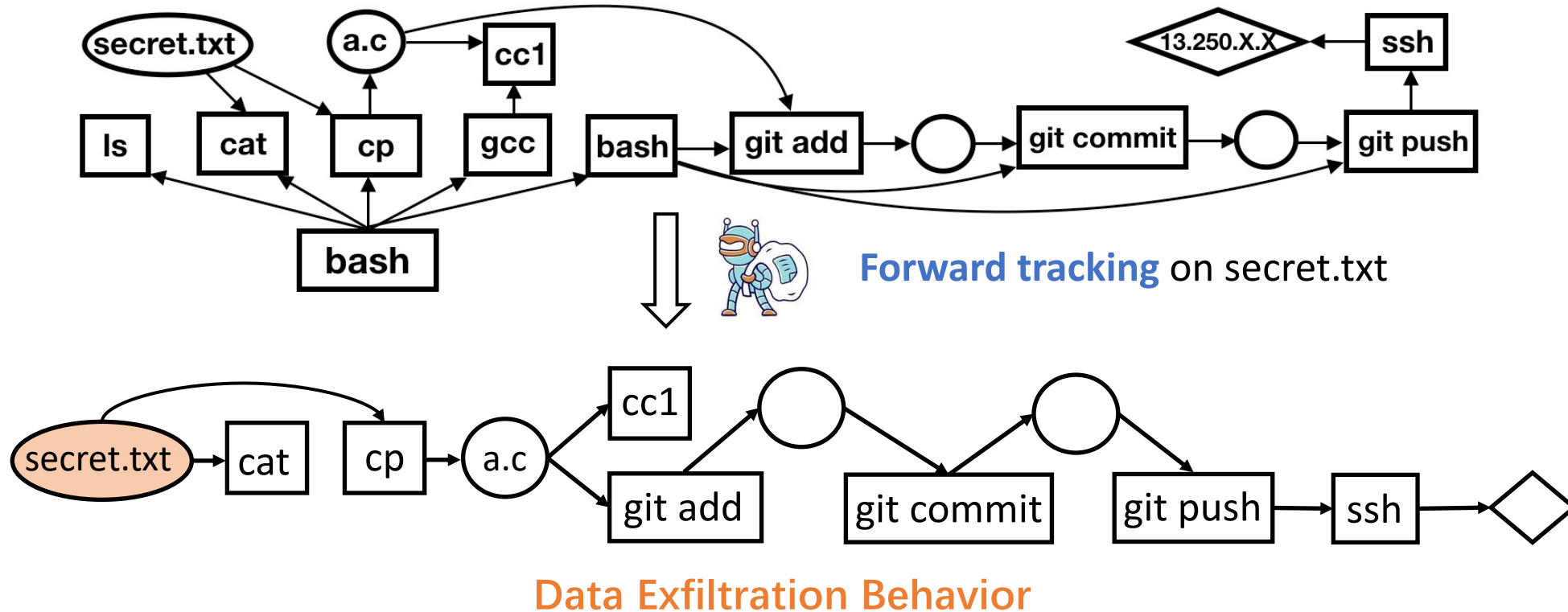


Compiling program using GCC

Reveal the semantics of audit events from their usage **contexts** in logs

# Our Insights

How do analysts manually identify behaviors from audit events?

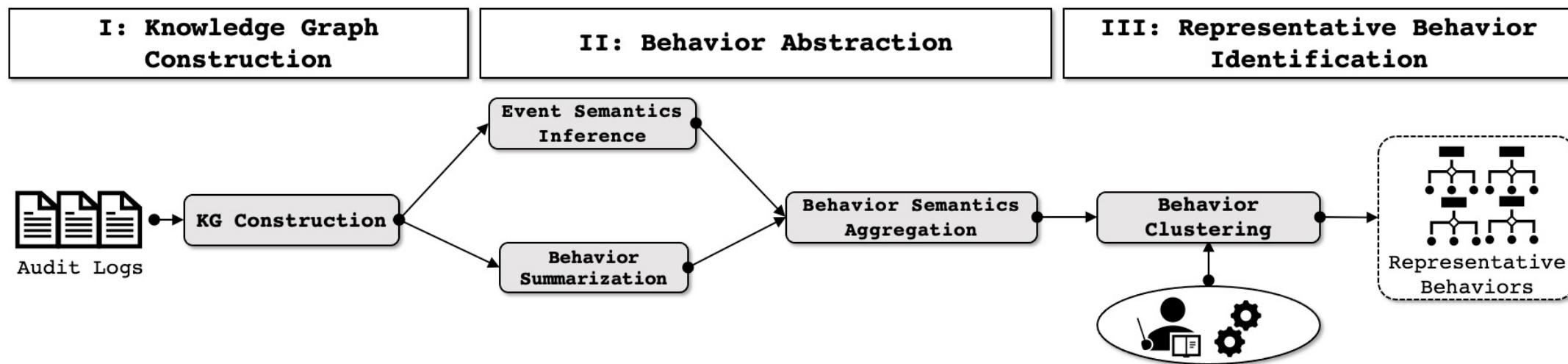


Summarize behaviors by tracking **information flows** rooted at **data objects**

# WATSON

An automated behavior abstraction approach that **aggregates the semantics of audit logs to model behavioral patterns**

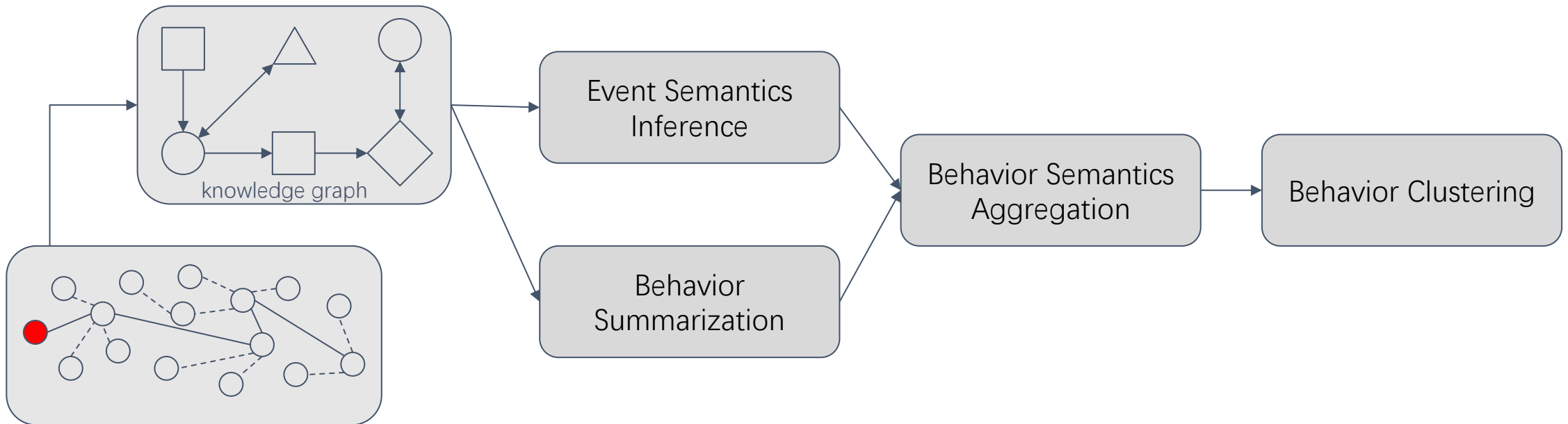
- Input: audit logs (e.g., Linux Audit<sup>[1]</sup>)
- Output: representative behaviors



[1] Linux Kernel Audit Subsystem. <https://github.com/linux-audit/audit-kernel>.

# Watson

- Bridging the semantic gap between low-level audit logs & high-level system behaviors
  - Using contextual information in log-based KG (event semantic inference)
  - Clustering semantically similar behaviors (behavior summarization)





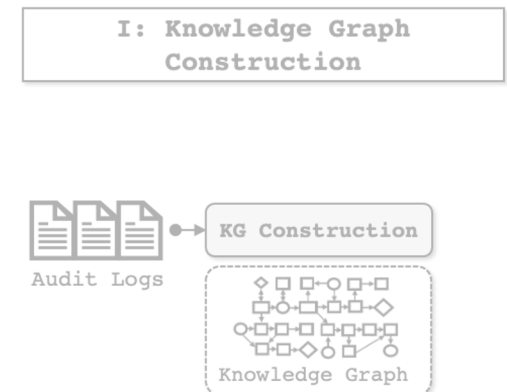
# Knowledge Graph Construction

We propose to use a **knowledge graph** (KG) to represent audit logs:

- KG is a directed acyclic graph built upon triples
- Each triple, corresponding to an audit event, consists of three elements (head, relation, and tail):

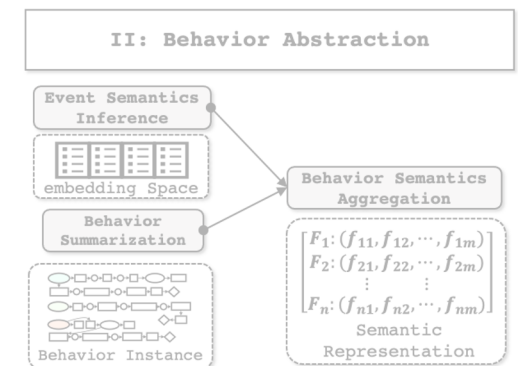
$$\mathcal{KG} = \{(h, r, t) | h, t \in \{Process, File, Socket\}, r \in \{Syscall\}\}$$

- KG unifies **heterogeneous** events in a **homogeneous** manner



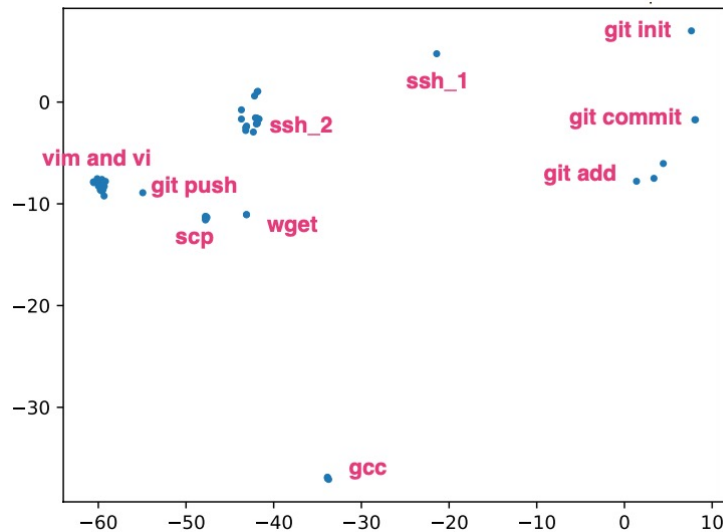
# Event Semantics Inference

- Suitable **granularity** to capture contextual semantics
  - Prior work [CCS'17] studies log semantics using events as basic units.
  - Lose contextual information within events
  - Working on **Elements** (head, relation, and tail) preserves more contexts
- Employ an embedding model to extract contexts
  - Map elements into a vector space
  - Spatial distance represents semantic similarities
  - **TransE**: a translation-based embedding model
  - **Head + Relation  $\approx$  Tail  $\rightarrow$  Context decides semantics**

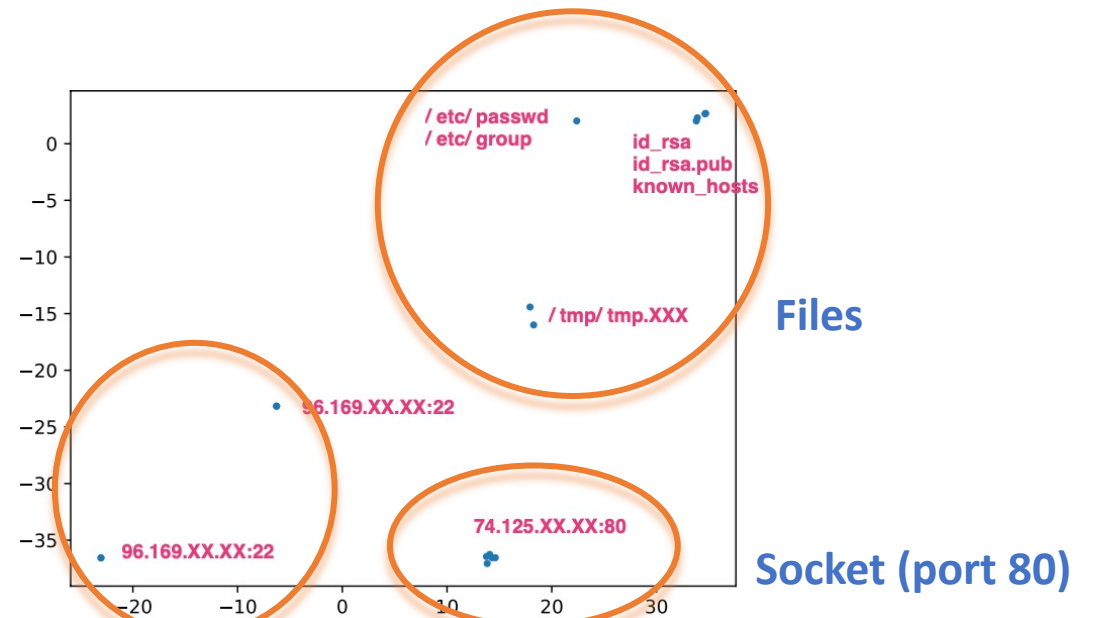


# Event Semantics Explicability

Use t-SNE to project the embedding space (64 dimensional in our case) into a 2D-plane, giving us an intuition of embedding distribution



(a) 53 Program embeddings



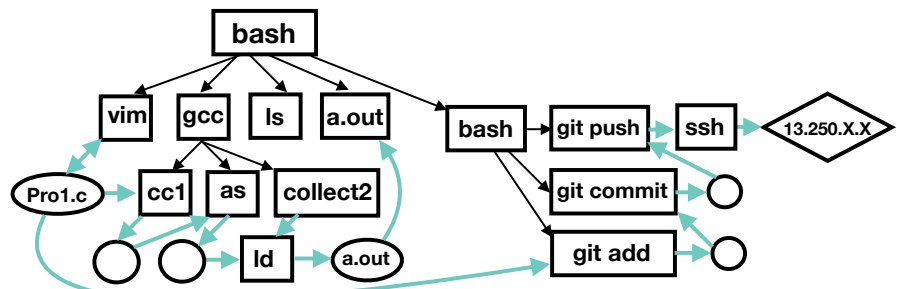
(b) 25 data object embeddings

**Semantically similar** system entities are **clustered** in the embedding space

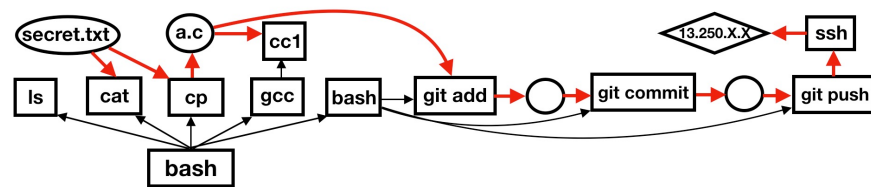
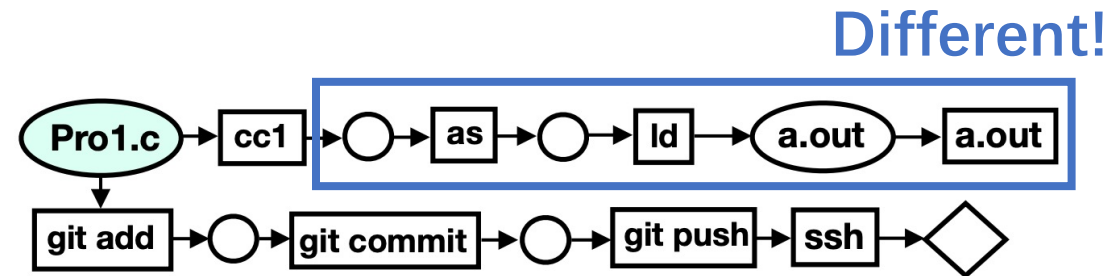
# Behavior Summarization

Individual behavior identification: Apply an adapted depth-first search (DFS) to track information flows rooted at a data object:

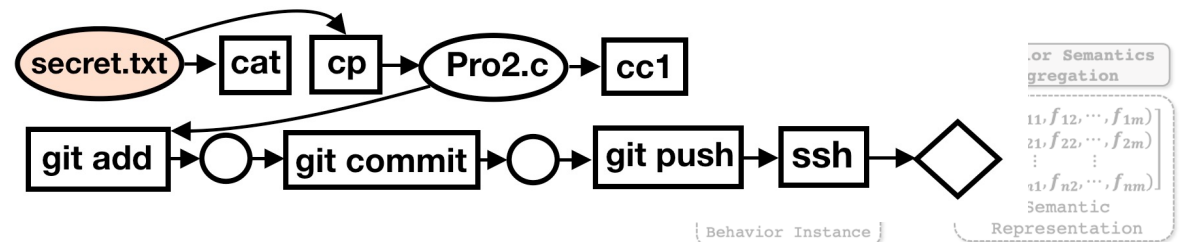
- Perform the DFS on every data object except libraries
- Two behaviors are merged if one is the subset of another




Program Compiling and Upload

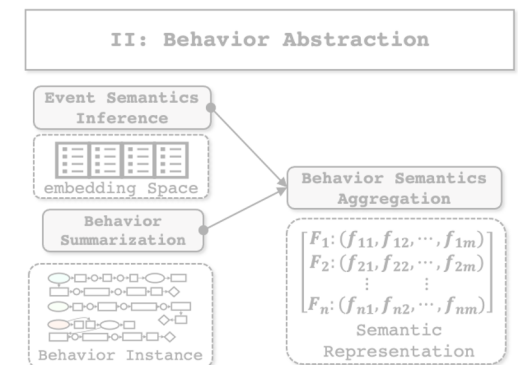


Data Exfiltration



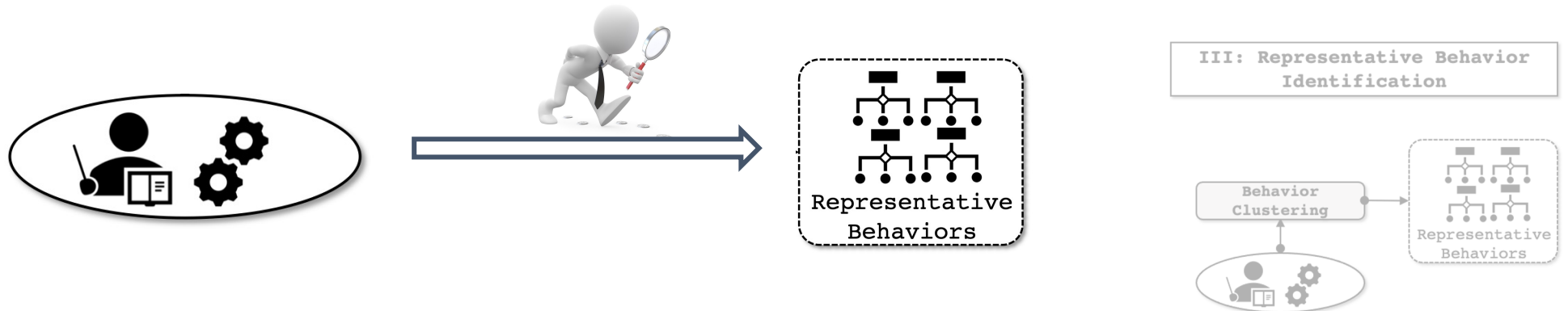
# Behavior Semantics Aggregation

- How to aggregate event semantics to represent behavior semantics?
  - Naïve approach: Add up the semantics of a behavior's constituent events
  - Assumption: audit events equally contribute to behavior semantics 
- **Relative event importance**
  - Observation: behavior-related events are common across behaviors, while behavior-unrelated events the opposite
  - Apply frequency as a metric to define event importance
  - Quantify the frequency: **Inverse Document Frequency (IDF)**
- The presence of **noisy events**
  - Redundant events [CCS'16] & Mundane events



# Representative Behavior Identification

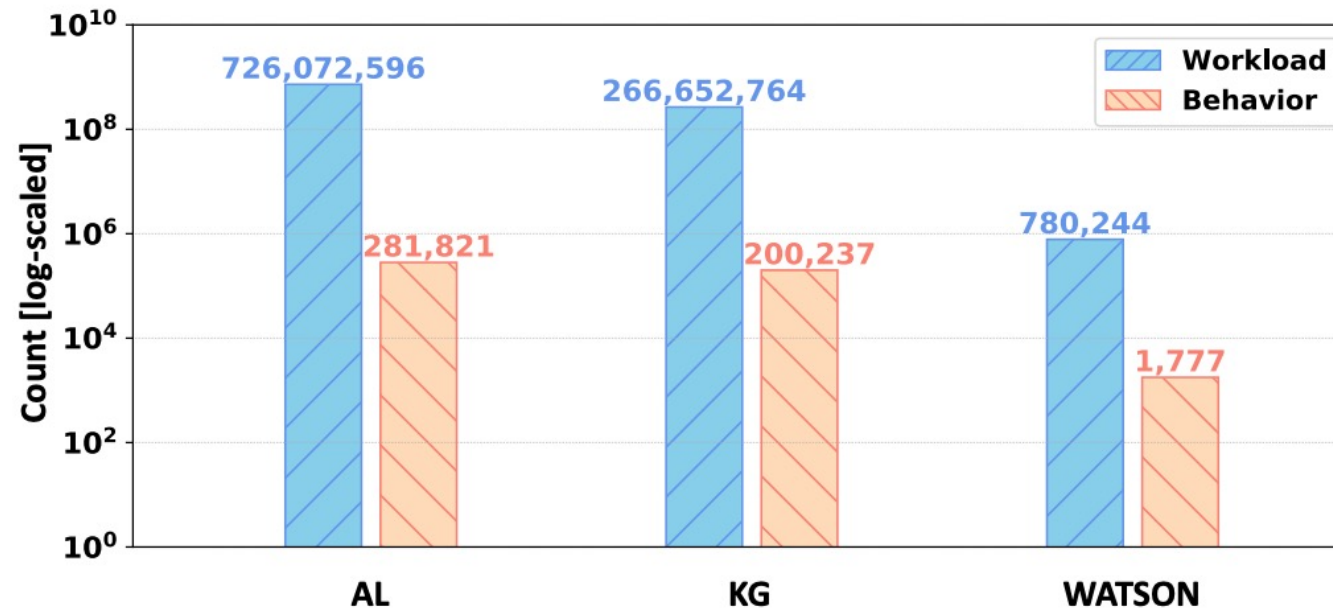
- Cluster semantically similar behaviors: **Agglomerative Hierarchical Clustering analysis (HCA)**
- Extract the most representative behaviors
  - Representativeness: Behavior's average similarity with other behaviors in a cluster
  - **Analysis workload reduction**: Do not go through the whole behavior space



# Efficacy in Attack Investigation

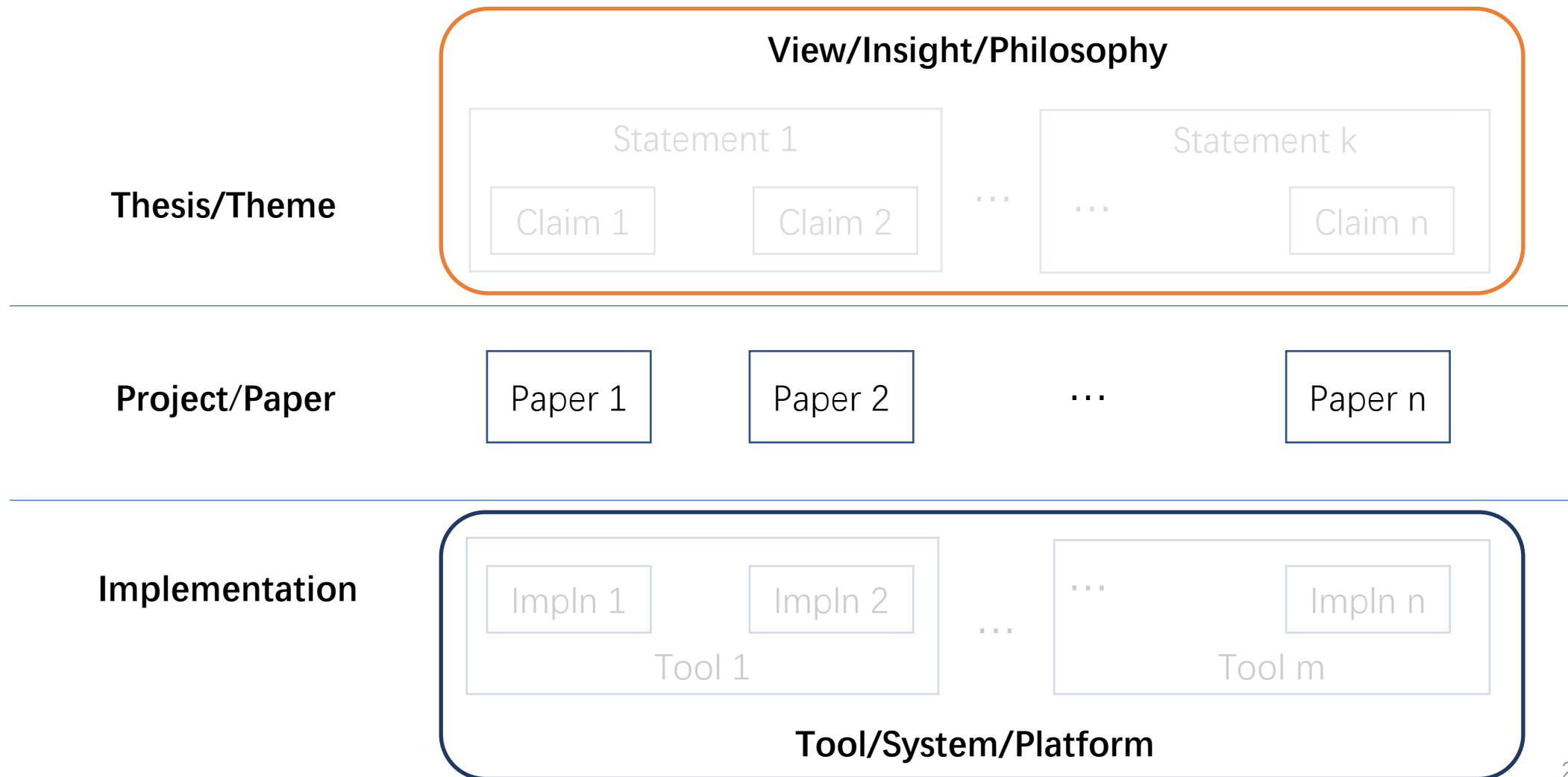
Measure the **analysis workload reduction** of APT attack investigation in the DARPA TRACE dataset:

- Analysis workload: the number of events to recognize all behaviors



**Two orders of magnitude reduction** in analysis workload and behaviors

# Dimensions of Computer System Research





# Dimensions of System Research

<b>Human</b>	Human factor, social engineering	Law, policy, politics
<b>System</b>	Isolation, memory exploit, provenance analysis, ...	Psychology, cognition, responsibility
	<b>System</b>	<b>Human</b>



Understanding systems 理解系统  
Abstracting knowledge 提炼知识  
Connecting facts 参悟规律

- Development of audit log analysis
- Our Insights
  - Infer audit event semantics by usage contexts
  - Identify behaviors with information flows rooted at data objects
- Research and experimentation

liangzk@comp.nus.edu.sg

**Thank you!**



Understand the movement of the sun and moon from traces of shadows under the roof.

审堂下之阴，而知日月之行，阴阳之变也