WATSON: Abstracting Behaviors from Audit Logs via Aggregation of Contextual Semantics

Jun Zeng, Zheng Leong Chua, Yinfang Chen, Kaihang Ji, Zhenkai Liang, Jian Mao

NDSS 2021
Security Incidents Are on The Rise Globally

Months later, the great Twitter hack still boggles my mind

Some of the biggest accounts in the world were made to tweet a bitcoin scam

By Jay Peters | @jaypeters | Dec 15, 2020, 11:45am EST

1.5 million affected by hack targeting Singapore’s health data

Local media say the hack is believed to be state-sponsored

By James Vincent | Jul 20, 2018, 6:48am EDT

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

- **FD: 0x3**
- **PID: 30204**
Investigation Using Audit Logs

Researchers use a **provenance graph** to navigate through audit logs:

- **Nodes**: system entities (e.g., process, file, and socket) & **Edges**: system calls
- **Backward/forward tracking** to find root cause of an attack and its ramifications

Real-world audit logs are always **large-Scale**, and provenance graphs are **Sophisticated**!
Related Work

- Scale up provenance analysis:
  - Data reduction [NDSS’16, 18 …] & Query system [Security’18, ATC’18 …]
  - Recognizing behaviors of interest requires intensive manual efforts

  **A semantic gap** between low-level events and high-level behaviors

- Apply expert-defined specifications to bridge the gap
  - Match audit events against domain rules that describe behaviors
  - 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**)
Related Work

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

Recognizing behaviors of interest requires intensive manual efforts.

A semantic gap between low-level events and high-level behaviors.

- Apply expert-defined specifications to bridge the gap.
- Match audit events against domain rules that describe behaviors.
- Query graph [VLDB’15, CCS’19], Tactics Techniques Procedures (TTPs) specification [SP’19,20], and Tag policy [Security’17,18]

How can we

- **abstract** high-level behaviors from low-level audit logs?
- **cluster** similar behaviors to assist human investigation?

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:
1. `cp` secret.txt 13.250.X.X
2. `gcc` a.c
3. `cp` a.c github
4. `gcc` a.c
5. `cp` a.c EXE
6. `cp` a.c a.out

Motivating Example Logs:

```
ls
vim
gcc
collect2
as
ld
Pro1.c
a.out
sudo
vim
tar
bash
git add
git commit
git push
13.250.X.X
apt
sudo
apt
sh
gpgv
apt-key
dpkg
update-motd
ssh
http
apt-config
dpkg
rm
find
apt-key
bash
ssh
13.250.X.X
```
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

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

- Input: audit logs (e.g., Linux Audit\textsuperscript{[1]})
- Output: representative behaviors

\textsuperscript{[1]} Linux Kernel Audit Subsystem. https://github.com/linux-audit/audit-kernel.
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):
  \[
  \mathcal{KG} = \{(h, r, t) | h, t \in \{\text{Process, File, Socket}\}, r \in \{\text{Syscall}\}\}
  \]
- 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 ≈ Tail** → **Context decides semantics**
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

Different!
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
Evaluation

• **Experimental Setup:**
  - Simulated dataset: \(275,863,292\) events in 4,280 SSH sessions
  - DARPA Trace Dataset\(^2\): \(726,072,596\) events in 211 graphs

• **Behavior Abstraction Accuracy:**
  - Can WATSON cluster similar behaviors?

• **Event Semantics Explicability:**
  - Does inferred event semantics match our domain knowledge?

• **Efficacy in Attack Investigation:**
  - How many manual efforts can WATSON save?

Behavior Abstraction Accuracy

Use vector-representation semantics of behaviors to predict SSH sessions with similar behaviors:

- 17 daily routines and 8 real-life attacks
- Extensive noisy behaviors

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Recall</th>
<th>Precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Installation</td>
<td>95.3%</td>
<td>97.9%</td>
<td>96.6%</td>
</tr>
<tr>
<td>Data Theft</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Average</td>
<td>94.2%</td>
<td>92.8%</td>
<td>92.8%</td>
</tr>
</tbody>
</table>

High F1 score on both benign and malicious behavior abstraction
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.

Semantically similar system entities are clustered in the embedding 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
Summary

- We propose WATSON to
  - Abstract behaviors from audit events
  - Cluster semantically similar behaviors

- Insights
  - Infer audit event semantics by usage contexts
  - Identify behaviors with information flows rooted at data objects

- Evaluation
  - Substantially reduce analysis workload
  - High F1 score on behavior abstraction
WATSON: Abstracting Behaviors from Audit Logs via Aggregation of Contextual Semantics

Thanks!

junzeng@comp.nus.edu.sg