A Semantic Framework for Data Analysis in Networked Systems

Arun Viswanathan, Alefiya Hussain, Jelena Mirkovic, Stephen Schwab, John Wroclawski

USC/Information Sciences Institute
Data Analysis in Networked Systems

- Is my hypothesis validated?
- Did my experiment run as expected?
- Why did failure X happen?
- Is there any evidence of a known attack?

- Alerts
- Audit Logs
- Packet Dumps
- Application Logs
Our Semantic Approach

MODELS
Capture high-level understanding of system

EXPERT

Packet dumps
Webserver logs
Auth logs

Data collected from an execution of a system

Semantic Analysis Framework

Pose Questions?

hypothesis? expectations met? evidence of known attacks?

failure X why?

Answers to Questions

Models drive analysis over data!
Approximate Lay of the Land

Level of Analysis Abstraction

Low

High

Performance

Low

High

Patterns
Ex. wireshark, tcpdump, snort

Queries
Ex. SQL, Splunk

Language-based
Ex. Bro, SEC

Logic-based
Ex. temporal-logic specification for IDS, CTPL-logic for malware

Custom Hackery
Ex. scripts, tools
Approximate Lay of the Land

Low-level data details
(low expressiveness, high performance, low reusability)

- Patterns
  Ex. wireshark, tcpdump, snort
- Queries
  Ex. SQL, Splunk
- Language-based
  Ex. Bro, SEC

Custom Hackery
Ex. scripts, tools

Logic-based
Ex. temporal-logic specification for IDS, CTPL-logic for malware

Trade performance for expressiveness

Models
(high expressiveness, usable performance, reusable)

Key differences with other logic-based approaches

- **Composable abstractions** to capture semantics
- **Expressive relationships** for networked systems
Basics of our Modeling Approach

Behavior
(fundamental abstraction)
Sequence or group of one or more related facts

Complex Behaviors
Related behaviors

Model
Top-level behavior

Models encode higher-level system semantics!

DATA
Multitype, multivariate, timestamped

FACTS
(ex: FILE_OPEN, FILE_CLOSE, TCP_PACKET, ....)

Relationships are key
Relationships in the Modeling Language

Temporal Relationships
- Causality/Ordering
- Eventuality
- Invariance
- Synchrony/Timing

Concurrent Relationships
- Parallelism
- Overlaps

Logical Relationships
- Combinations
- Exclusions

Dependency relationships b/w data attributes

Temporal Operators
- FILE_OPEN ~> FILE_CLOSE

Interval Temporal Operators
- HTTP_FLOW \textit{olap} FTP_FLOW

Logical Operators
- \texttt{EXPT\_SUCCESS} $\text{xor}$ \texttt{EXPT\_FAIL}

- FILE\_CLOSE.name = FILE\_OPEN.name

A file open \textit{eventually leads to} a file close

HTTP and FTP flows are \textit{concurrent}.

Experiment \textit{either succeeds or fails}

File open and file close are behaviors \textit{related by their filename}. 
**Cache Poisoning Behavior**

**Objective:** Attacker poisons the victim's DNS cache.

Steps 1-4 keep running in a loop.

**KEY ISSUES**

Attacker fails to poison cache due to

1. Race conditions with real nameserver.
2. Incorrectly GUESSED responses.

**Cache Poisoning Behavior (DNS Kaminsky)**

1. **Send Query**
   - **Attacker (A)**

2. **Forward Query**
   - **Real Nameserver (R)**

3. **Flood of GUESSED responses**
   - **Victim Nameserver (V)**

4. **Correct response**

Steps 1-4 keep running in a loop.
Analysis using typical approach

Tricky to analyze

- Requires Expertise.
- Too many random values in the data to extract using simple patterns.
- Race conditions (timing issues) are hard to debug over 10's of thousands of packets.
- Many ways to fail.
SUCCESS = A guesses right and wins race with R
Model of Behavior

SUCCESS = A guesses right and wins race with R.

TIMING_FAIL = A guesses right but loses race to R.

Nodes: Simple behavior

Arrows: Causal relationships

Path (from root to leaf): Complex Behaviors
Model of Behavior

Behavior Model = 1 SUCCESS + 3 FAILURES

SUCCESS = A guesses right and wins race with R.

TIMING_FAIL = A guesses right but loses race to R.

BADGUESS_1 = A guesses wrong response

SUCCESS = A guesses right and wins race with R.
Encoding the Model

1. Capture simple behaviors (to capture facts for each distinct attack step)

   \[
   \text{VtoR\_query} = \text{DNSREQRES.dns\_req}(\text{sip} = \$AtoV\_query.dip, \\
   \text{dnsquesname} = \$AtoV\_query.dnsquesname)
   \]

2. Relate simple behaviors to form complex behaviors (to capture the causal relationships between steps)

   \[
   \text{TIMING\_FAIL} = (AtoV\_query \rightsquigarrow VtoR\_query \rightsquigarrow RtoV\_resp \rightsquigarrow AtoV\_resp)
   \]

3. Define Behavior Model (assertion to capture users understanding of system operation)

   \[
   \text{DNSKAMINSKY} = \text{SUCCESS} \text{xor TIMING\_FAIL xor BADGUESS\_1 xor BADGUESS\_2}
   \]
Analysis Using the Model

Behavior captured in 20 lines of model

DNS Kaminsky Behavior model

[states]
sB = {sip=$sA.dip,dip=$sA.sip}

[behavior]
b = sA ~> sB

[states]
sB = {sip=$sA.dip,dip=$sA.sip}

[behavior]
b = sA ~> sB

[states]
sB = {sip=$sA.dip,dip=$sA.sip}

[behavior]
b = sA ~> sB

DNS Data

Behavior captured in 20 lines of model

DNS Kaminsky Behavior model

Answers in the form of facts satisfying the model.

Summary : DNSCACHEPOISON_TIMING_FAIL

Total Matching Instances: 622

dtype | timestamp | sip | dip | sport | dport | dnsid | dnsauth
-----------------------------------------------
PKT_DNS | 1275515486 | 10.1.11.2 | 10.1.4.2 | 6916 | 53 | 47217 |
PKT_DNS | 1275515486 | 10.1.4.2 | 10.1.6.3 | 32778 | 53 | 15578 |
PKT_DNS | 1275515486 | 10.1.6.3 | 10.1.4.2 | 53 | 32778 | 15578 | realns.eby.com
PKT_DNS | 1275515486 | 10.1.4.2 | 10.1.6.3 | 53 | 32778 | 47217 | fakens.fakeeby.com
PKT_DNS | 1275515486 | 10.1.4.2 | 10.1.6.3 | 53 | 32778 | 47217 | fakens.fakeeby.com
PKT_DNS | 1275515486 | 10.1.4.2 | 10.1.6.3 | 53 | 32778 | 47217 | fakens.fakeeby.com
PKT_DNS | 1275515486 | 10.1.4.2 | 10.1.6.3 | 53 | 32778 | 47217 | fakens.fakeeby.com
PKT_DNS | 1275515486 | 10.1.4.2 | 10.1.6.3 | 53 | 32778 | 47217 | fakens.fakeeby.com

TIMING_FAIL
(A loses the race against R)
Current Implementation and Performance

- Prototype algorithm for applying models over data.

- Algorithm performance
  - $O(N^2)$ worst-case performance
  - Straight-forward

- Analysis Framework
  - Written in Python
  - SQLite-based storage backend

- Scalability and performance issues are under active investigation.
Applicability

- Broad range of event-based modeling in networked systems
- More examples in paper
  - Modeling hypotheses
    - Ex. Validating DoS detection heuristics over traces
  - Modeling a security threat
    - Ex. Model of a simple worm spread over IDS logs
  - Modeling dynamic change
    - Ex. Model of changes in traffic rate due to attack.
Future Work

- Extend Modeling Capabilities
  - Modeling probabilistic behavior
  - Modeling packet distributions
- Analysis Framework
  - Scalability and performance
  - Reducing the computational complexity of correlations using dependent attributes.
Composing, Sharing and Reusing

Semantic Analysis Framework enables data analysis at higher-levels of abstraction.

Composing models to create higher-level meaning

Sharing and reusing expertise

Abstract Behavior Models

WIPE

Exploratory data analysis

Enable sharing and reuse of experiments
Thank You!

Our framework will soon be publicly available at
http://thirdeye.isi.deterlab.net

Please register on our mailing-list to stay in tune with release and updates