Storytelling Security: Scalable Causal Analysis for Host-Wide Anomaly Detection

Danfeng (Daphne) Yao
Assistant Professor

Department of Computer Science
Virginia Tech

ARO STIR Report August 2012
Personnel and Collaborators in Yao group

Current Ph.D. students
- Kui Xu
- Huijun Xiong
- Johnny Shu
- Tony Zhang
- Hussain Almohri
- Karim Elish

Incoming Ph.D. student
- Fang Liu

Graduated Ph.D. student
- Saman Zarandioon

at Amazon.com

Collaborators in CS Dept
- Naren Ramakrishnan
- Barbara Ryder
- Layne Watson
Anomaly Detection For System Assurance

Problem: how to ensure system assurance?

- **Signature based scanning, firewalls, IDS/IPS**
- **To detect malware behaviors at run time**
  - E.g., system call execution, memory/stack access
- **But what about zero-day malware/exploit?**
  - **To avoid infection**
    E.g., to prevent remote code execution, MTD
  - **To detect changes in code base**
    E.g., TPM attestation
  - **Anomaly detection**
    E.g., [Denning ’87], [Forrest et al. ’96], [Sekar ’01], [Giffin ’04]

- But how to define the normalcy of a host?
Anomaly detection requires:

- Definitions for the norm or normalcy, or
- Mechanisms to learn normal patterns, and
- Mechanisms to observe and collect authentic data

Why simple statistical methods are inadequate in computer anomaly detection?

State-of-the-art anomaly detection solutions are limited to system calls

Our goal: host-wide monitoring and anomaly detection

Our storytelling security approach: to perform scalable structured causal analysis of events on a computer
Our Existing Work on User-Intention Based Traffic Dependence Analysis

H. Zhang, D. Yao, N. Ramakrishnan, and M. Banick.
Cause and Effect in Traffic Anomaly Detection

How to distinguish the malicious outbound packets from the legitimate ones on a host?

Our approach: To identify dependence among outbound traffic
A Technical Challenge

Browser automatically sends many outbound requests.
Work Flow of CR-Miner

Traffic events (outbound)

User events

CR-Miner

Dependence rules

User event

Traffic dependence graph (TDG)

Threat model: application-level malware
Our Storytelling Security Vision: Scalable Structured Causal Analysis for Host-Wide Monitoring

- Outbound packets
- Device modules
- Networked Applications
- Transport
- Network
- Data Link
- Physical
- User inputs
- System call traces
- File sys events
# Events and their attributes

## User events

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Event Name</th>
<th>Value</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 0:0:01.077</td>
<td>KeyDown</td>
<td>Return</td>
<td><a href="http://www.engadget.com/">http://www.engadget.com/</a></td>
</tr>
<tr>
<td>B 0:0:02.910</td>
<td>MouseClick - Left</td>
<td>X=1069 Y=474</td>
<td><a href="http://www.cnet.com/">http://www.cnet.com/</a></td>
</tr>
<tr>
<td>C 0:0:03.000</td>
<td>Wheel</td>
<td>-120</td>
<td>N/A</td>
</tr>
</tbody>
</table>

## Traffic events

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Object Requested</th>
<th>Remote Domain Name</th>
<th>Referrer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0:0:02.863</td>
<td>/</td>
<td><a href="http://www.engadget.com">www.engadget.com</a></td>
<td><a href="http://www.engadget.com/">http://www.engadget.com/</a></td>
</tr>
<tr>
<td>2 0:0:02.873</td>
<td>/media/main.css</td>
<td><a href="http://www.engadget.com">www.engadget.com</a></td>
<td><a href="http://www.engadget.com/">http://www.engadget.com/</a>...</td>
</tr>
<tr>
<td>3 0:0:03.113</td>
<td>/</td>
<td><a href="http://www.cnet.com">www.cnet.com</a></td>
<td>null</td>
</tr>
</tbody>
</table>
Definitions in Our Traffic Dependency Graph (TDG)

Definition of security: a legitimate traffic event should belong to a tree in a TDG that is rooted at a legitimate user event. Otherwise, it is a vagabond traffic event.

User events (Root)

Traffic events (Subroot)

Timeline
Our BFS-Based Algorithm to Construct Traffic Dependence Graph

Input:
- an existing TDG (trees of events, which root at user events)
- a new outbound traffic event $q$

Output:
- whether or not $q$ is legitimate
Our prototype in Windows is called CR-Miner.

Signer and verifier for the integrity of HTTP requests with MAC
Highlights on Experiments

1. How accurate is the dependency inference algorithm?
   - ≥ 98% hit rates for all users
   - Average 99.6% with white listing (0.4% contains true positives)
   - 99.72% for top 20 Alexa.com websites (i.e., 0.28% false positives)

2. Does the inference accuracy suffer in noisy traffic?
   - 99.2% accuracy in two-user merged data set

3. Can we detect real-world stealthy malware traffic?
   - Infostealer spyware
   - Proof-of-concept password sniffer (malicious Firefox extension similar to Firespyfox)

User study with 20 participants; 30-minute surfing for each user

Hit rate: percentage of traffic events whose parents are identified by CR-Miner

Related Work in Yao Group

- What/who causes outbound traffic
  - [Hao et al. IEEE WSCS ’12]
- What/who downloads files on the computer
  - [Xu et al. NSS ‘11]
- Where the keystroke is from; where the packet is from (cryptographic provenance verification)
  - [Xu et al. IEEE TDSC ‘12]
- Whether or not the apps behave
  - [Elish et al. IEEE MoST ‘12]

For preserving system integrity
Future Work on Storytelling Security

- **To automatically mine causal relations with machine learning techniques**
  
  E.g., how to define features considering the data diversity and semantics?
  
  Our preliminary work on naïve Bayesian classifier (for pair-wise dependencies) gave promising results.

- **To model storytelling security, theoretical analysis**
  
  E.g., general requirements, components, workflow, limitations
  
  E.g., FSA representation
  
  E.g., connection with Schneider’s EM

- **Experimental demonstration**
  
  E.g., including DNS traffic in traffic dependency analysis
  
  E.g., analysis of server-side applications