SLEUTH: Real-time Attack Scenario Reconstruction from COTS Audit Data

Md Nahid Hossain, Sadegh M. Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R. Sekar, Scott D. Stoller, and V.N. Venkatakrishnan
Stony Brook University and University of Illinois at Chicago

1This work was primarily supported by DARPA (contract FA8650-15-C-7561) and in part by NSF (CNS-1319137, CNS-1421893, CNS-1514472 and DGE-1069311) and ONR (N00014-15-1-2208 and N00014-15-1-2378).
Challenges of Advanced Persistent Threat (APT) Campaigns

- APTs combine social engineering (e.g., spearphishing) with advanced exploits
  - Get past first-line defenses, e.g., ASLR, DEP, and sandboxes

- Enterprises forced to rely on second-line defenses
  - Intrusion detection systems (IDS), Security incident and event management (SIEM), ...

- Key challenges
  - “Needle in a haystack” — spot the minuscule fraction of real attacks within vast quantities of data emitted by these systems.
  - “Connecting the dots” — stitch isolated steps together into a larger campaign.

Result: Many APT campaigns remain undetected for months.
Previous Research

Attack detection: Numerous intrusion detection techniques have been developed.
- Real-world use hampered by high false positive rates

Linking attack campaign steps: Backtracker [King and Chen] and subsequent works use dependencies recorded in system logs to stitch together attacker activities
- Forensic tool — does not help analyst to understand ongoing attacks in real-time.
- Result can include many irrelevant events due to explosion of (false) dependencies.
- Fine-grained dependency tracking techniques developed to address this problem, but have performance and compatibility costs.
Goals and Challenges

- Real-time reconstruction of APT campaign from audit logs
  - Provide compact visual summary of the campaign

- Key challenges
  - *Data volume*: hundreds of millions to billions per day
  - *“Needle in a haystack”* — only a small fraction of these are attacks, perhaps one in a million
    - Avoid being swamped in false positives
  - *“Connecting the dots”* — link successive steps of an APT campaign

- Part of the DARPA Transparent Computing program
  - Our adversarial evaluation relies on Red Team engagements organized by DARPA.
SLEUTH Architecture and Contributions

- Space-efficient in-memory dependence graph representation
- Effective attack detection based on trustworthiness and confidentiality tags
- Customizable policy framework for tag assignment and propagation
- Highly effective and efficient tag-based backward and impact analysis
- Experimental evaluation: fast, accurate and compact visual representation of APT campaigns
Illustrative Example

**Attacker goal:** Insert backdoor into a vendor’s software

**Steps:**

1. Use a browser vulnerability to drop a malicious version of *crt1.o* in `/home/bob`
**Illustrative Example**

- **Attacker goal:** Insert backdoor into a vendor’s software

- **Steps:**
  1. Use a browser vulnerability to drop a malicious version of `crt1.o` in `/home/bob`
  2. Modify Bob’s `.bashrc` to redefine `sudo`
**Illustrative Example**

- **Attacker goal:** Insert backdoor into a vendor’s software

- **Steps:**
  1. Use a browser vulnerability to drop a malicious version of `crt1.o` in `/home/bob`
  2. Modify Bob’s `.bashrc` to redefine `sudo`
  3. Next time Bob uses `sudo`, it copies `/home/bob/crt1.o` to `/lib/crt1.o`
**Attacker goal:** Insert backdoor into a vendor’s software

**Steps:**

1. Use a browser vulnerability to drop a malicious version of `crt1.o` in `/home/bob`
2. Modify Bob’s `.bashrc` to redefine `sudo`
3. Next time Bob uses `sudo`, it copies `/home/bob/crt1.o` to `/lib/crt1.o`
4. When Alice builds her software, malicious `crt1.o` code is included in her executable.
**Attacker goal:** Insert backdoor into a vendor’s software

**Steps:**

1. Use a browser vulnerability to drop a malicious version of `crt1.o` in `/home/bob`
2. Modify Bob’s `.bashrc` to redefine sudo
3. Next time Bob uses sudo, it copies `/home/bob/crt1.o` to `/lib/crt1.o`
4. When Alice builds her software, malicious `crt1.o` code is included in her executable.
5. When this software is run, it exfiltrates sensitive data (password file)
## Adversarial Engagement Overview

<table>
<thead>
<tr>
<th>Campaign</th>
<th>Length (hours)</th>
<th># of events</th>
<th>Drop &amp; load</th>
<th>Gather intel.</th>
<th>Insert backdoor</th>
<th>Escalate privilege</th>
<th>Data exfil.</th>
<th>Clean-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win-1</td>
<td>06:22</td>
<td>100K</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Win-2</td>
<td>19:43</td>
<td>401K</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lin-1</td>
<td>07:59</td>
<td>2.68M</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lin-2</td>
<td>79:06</td>
<td>38.5M</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Lin-3</td>
<td>79:05</td>
<td>19.3M</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bsd-1</td>
<td>08:17</td>
<td>701K</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Bsd-2</td>
<td>78:56</td>
<td>5.86M</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Bsd-3</td>
<td>79:04</td>
<td>5.68M</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Attack Detection Using Provenance Tags
Provenance Tags

Trustworthiness (t-tag)

**Benign authentic**: Data from strongly *authenticated* sources believed to be *benign*.

**Benign**: Believed to be benign, but sources not well-authenticated.

**Unknown**: No good basis to trust this source.
Provenance Tags

Trustworthiness (t-tag)

**Benign authentic:** Data from strongly *authenticated* sources believed to be *benign*.

**Benign:** Believed to be benign, but sources not well-authenticated.

**Unknown:** No good basis to trust this source.

Code Vs Data Trustworthiness

- Processes have two t-tags: *code t-tag* and *data t-tag*
- Separation (a) aids detection and (b) speeds analysis by focusing on fewer root causes
Provenance Tags

Trustworthiness (t-tag)

Benign authentic: Data from strongly authenticated sources believed to be benign.

Benign: Believed to be benign, but sources not well-authenticated.

Unknown: No good basis to trust this source.

Confidentiality (c-tag)

Secret: Highly sensitive, e.g., /etc/shadow

Sensitive: Disclosure has security impact, but less than disclosed secrets.

Private: Loss may not pose a direct security threat.

Public: Widely available, e.g., on public web sites

Code Vs Data Trustworthiness

- Processes have two t-tags: code t-tag and data t-tag
- Separation (a) aids detection and (b) speeds analysis by focusing on fewer root causes
Approach: Focus on *motive* and *means*

**Motive:** Does an act advance attacker’s high-level objectives?
- Deploy and run attacker code
- Replace/modify important files, e.g., `/etc/passwd`, ssh keys, ...
- Steal and exfiltrate sensitive data

**Means:** Can the attacker control the action?
- Is the process performing the action trustworthy?
Attack Detection Using Provenance Tags

Approach: Focus on *motive* and *means*

**Motive:** Does an act advance attacker’s high-level objectives?
- Deploy and run attacker code
- Replace/modify important files, e.g., `/etc/passwd`, ssh keys, ...
- Steal and exfiltrate sensitive data

**Means:** Can the attacker control the action?
- Is the process performing the action trustworthy?

**Benefits**
- Application-independent
- No need for training
- Resists attacker manipulation (assuming provenance isn’t compromised)
**Untrusted exec (UE):** Subject w/ high code trustworthiness execs lower t-tag object.

**Suspicious modification (SM):** Subject with lower code tag modifies higher t-tag file.

**Data leak (DL):** Untrusted subject writes confidential data to network.

**Untrusted execution preparation (UP):** Memory/file objects with low data trustworthiness made executable.
Attack Detection Policies

**Untrusted exec (UE):** Subject w/ high code trustworthiness execs lower t-tag object.

**Suspicious modification (SM):** Subject with lower code tag modifies higher t-tag file.

**Data leak (DL):** Untrusted subject writes confidential data to network.

**Untrusted execution preparation (UP):** Memory/file objects with low data trustworthiness made executable.
Untrusted exec (UE): Subject w/ high code trustworthiness execs lower t-tag object.

Suspicious modification (SM): Subject with lower code tag modifies higher t-tag file.

Data leak (DL): Untrusted subject writes confidential data to network.

Untrusted execution preparation (UP): Memory/file objects with low data trustworthiness made executable.
**Attack Detection Policies**

**Untrusted exec (UE):** Subject w/ high code trustworthiness execs lower t-tag object.

**Suspicious modification (SM):** Subject with lower code tag modifies higher t-tag file.

**Data leak (DL):** Untrusted subject writes confidential data to network.

**Untrusted execution preparation (UP):** Memory/file objects with low data trustworthiness made executable.
**Attack Detection Policies**

**Untrusted exec (UE):** Subject w/ high code trustworthiness execs lower t-tag object.

**Suspicious modification (SM):** Subject with lower code tag modifies higher t-tag file.

**Data leak (DL):** Untrusted subject writes confidential data to network.

**Untrusted execution preparation (UP):** Memory/file objects with low data trustworthiness made executable.
**Attack Detection Policies**

**Untrusted exec (UE):** Subject w/ high code trustworthiness execs lower t-tag object.

**Suspicious modification (SM):** Subject with lower code tag modifies higher t-tag file.

**Data leak (DL):** Untrusted subject writes confidential data to network.

**Untrusted execution preparation (UP):** Memory/file objects with low data trustworthiness made executable.
Flexible Policy Framework

- **Tag assignment** and **propagation** can be customized using policies.
  - Policies invoked at trigger points:
    - object creation, removal, read, write, load, execute, chmod, and chown
  - Can refer to subject, object and event attributes

- **Tag initialization** example:
  
  \[
  \text{init}(o): \text{match}(o.\text{name}, "^\text{file|IP:(10\.0|127)}") \rightarrow o.ttag = \text{BENIGN\_AUTH} \\
  \text{init}(o) : \text{match}(o.\text{name}, "^\text{IP:}\) \rightarrow o.ttag = \text{UNKNOWN}
  \]

- **Tag propagation**:
  - Default is to propagate tags from input to output
  - Custom policies created to capture exceptions, e.g., upgrade tag after a hash/signature verification.
### Attack Detection Summary

<table>
<thead>
<tr>
<th>Data Set</th>
<th># of Events</th>
<th>Untrusted Execution</th>
<th>Suspicious Modification</th>
<th>Execution Preparation</th>
<th>Data Leak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win-1</td>
<td>0.1M</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>Win-2</td>
<td>0.4M</td>
<td>2</td>
<td>108</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>Lin-2</td>
<td>39M</td>
<td>5</td>
<td>1</td>
<td>8</td>
<td>159</td>
</tr>
<tr>
<td>Lin-3</td>
<td>19M</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>5300</td>
</tr>
</tbody>
</table>

#### Key Point
- Almost zero false positives and negatives (except for data leak)
- Typically filters out 99.99% to 99.99999% of events
# Effectiveness of Split Trustworthiness Tags

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Untrusted Exec</th>
<th>Suspicious Modification</th>
<th>Untrusted Exec Prep</th>
<th>Data Leak</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single</td>
<td>Split</td>
<td>Single</td>
<td>Split</td>
</tr>
<tr>
<td>Win-1</td>
<td>21</td>
<td>3</td>
<td>1.2 K</td>
<td>3</td>
</tr>
<tr>
<td>Win-2</td>
<td>44</td>
<td>2</td>
<td>3.7 K</td>
<td>108</td>
</tr>
<tr>
<td>Lin-1</td>
<td>60</td>
<td>2</td>
<td>53</td>
<td>5</td>
</tr>
<tr>
<td>Lin-2</td>
<td>1.5 K</td>
<td>5</td>
<td>19.5 K</td>
<td>1</td>
</tr>
<tr>
<td>Lin-3</td>
<td>695</td>
<td>5</td>
<td>26.1 K</td>
<td>2</td>
</tr>
<tr>
<td>Average Reduction</td>
<td>45.39x</td>
<td>517x</td>
<td>6.24x</td>
<td>112x</td>
</tr>
</tbody>
</table>

**Key Point**

- Without separating code and data tags, we will have 5x to 500x more alarms
False Positives in Benign Environment

- Untrusted execution (download+execute) plays a critical role in detection
  - What happens in an environment with legitimate software downloads?
- Experiment: Linux servers with automated security updates and manual upgrades
- Approach: Use custom policy to upgrade downloaded files before `apt-get` invokes `dpkg`
  - Note: `apt-get` verifies signatures, so this is safe.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of Events</th>
<th>Duration hh:mm:ss</th>
<th>Packages Updated</th>
<th>Binary Files Written</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server 1</td>
<td>2.17M</td>
<td>00:13:06</td>
<td>110</td>
<td>1.8K</td>
</tr>
<tr>
<td>Server 2</td>
<td>4.67M</td>
<td>105:08:22</td>
<td>4</td>
<td>4.2K</td>
</tr>
<tr>
<td>Server 3</td>
<td>20.9M</td>
<td>104:36:43</td>
<td>4</td>
<td>4.3K</td>
</tr>
<tr>
<td>Server 4</td>
<td>5.09M</td>
<td>119:13:29</td>
<td>4</td>
<td>4.3K</td>
</tr>
</tbody>
</table>

No (false) alarms reported.
Tag-Based Backward and Forward Analysis
Goal: Identify entry point of an attack.

- Entry point is a *source*, i.e., vertex with in-degree zero.

Starting points: Suspect vertices marked by attack detectors.

Problem: Find source vertices from which a suspect vertex is reachable.
Backward Analysis

**Goal:** Identify entry point of an attack.

- Entry point is a *source*, i.e., vertex with in-degree zero.

**Starting points:** Suspect vertices marked by attack detectors.

**Problem:** Find source vertices from which a suspect vertex is reachable.

**Complications:**

- **Multiple sources:** Suspect vertex is reachable from multiple sources.
- **Multiple suspect nodes:** Typically, many detectors go off during attacks, and numerous vertices end up looking suspicious.
Backward Analysis: Key Ideas

- Prefer shorter paths over longer ones
- Favor paths that avoid redundant edges
- Prefer edges corresponding to flow of untrusted code
- and, to a lesser extent, untrusted data
- Preference encoded using a custom edge-weight function to Dijkstra’s shortest path algorithm
Backward Analysis: Key Ideas

- Prefer shorter paths over longer ones
- Favor paths that avoid redundant edges
- Prefer edges corresponding to flow of untrusted code
- and, to a lesser extent, untrusted data
- Preference encoded using a custom edge-weight function to Dijkstra’s shortest path algorithm
Backward Analysis: Key Ideas

- Prefer shorter paths over longer ones
- Favor paths that avoid redundant edges
- Prefer edges corresponding to flow of untrusted code
- and, to a lesser extent, untrusted data
- Preference encoded using a custom edge-weight function to Dijkstra’s shortest path algorithm
Backward Analysis: Key Ideas

- Prefer shorter paths over longer ones
- Favor paths that avoid redundant edges
- Prefer edges corresponding to flow of untrusted code
- and, to a lesser extent, untrusted data
- Preference encoded using a custom edge-weight function to Dijkstra’s shortest path algorithm
Backward Analysis: Key Ideas

- Prefer shorter paths over longer ones
- Favor paths that avoid redundant edges
- Prefer edges corresponding to flow of untrusted code
  - and, to a lesser extent, untrusted data
- Preference encoded using a custom edge-weight function to Dijkstra’s shortest path algorithm

Diagram:

- 129.55.33.44:80
- firefox
- /home/bob/.bashrc
- cp
- bash
- sudo
- /lib/crt1.o
- ld
- /home/alice/test
- EXEC
- UE: /home/alice/test
Backward Analysis: Key Ideas

- Prefer shorter paths over longer ones
- Favor paths that avoid redundant edges
- Prefer edges corresponding to flow of untrusted code
- and, to a lesser extent, untrusted data
- Preference encoded using a custom edge-weight function to Dijkstra’s shortest path algorithm
• Prefer shorter paths over longer ones
• Favor paths that avoid redundant edges
• Prefer edges corresponding to flow of untrusted code
• and, to a lesser extent, untrusted data
• Preference encoded using a custom edge-weight function to Dijkstra’s shortest path algorithm
Backward Analysis: Key Ideas

- Prefer shorter paths over longer ones
- Favor paths that avoid redundant edges
- Prefer edges corresponding to flow of untrusted code
- and, to a lesser extent, untrusted data
- Preference encoded using a custom edge-weight function to Dijkstra’s shortest path algorithm
Forward Analysis

**Goal:** Identify attack impact, in terms of all objects/subjects affected by the attack.
- Generate a subgraph of provenance graph that only includes objects and subjects affected by the attack.

**Starting point:** Sources identified by backward analysis

**Challenge:** Straight-forward dependence analysis may yield a graph with hundreds of thousands (if not millions) of edges.
Forward Analysis: Key Ideas

- Use cost metric to prune off distant nodes, i.e., nodes at a distance $\geq d_{th}$
Forward Analysis: Key Ideas

- Use cost metric to prune off distant nodes, i.e., nodes at a distance $\geq d_{th}$

- Cost metric favors edges with untrusted \textit{code} trustworthiness (cost=0);
- and, to a lesser extent, edges with untrusted \textit{data} trustworthiness (cost=1)
Forward Analysis: Key Ideas

- Use cost metric to prune off distant nodes, i.e., nodes at a distance $\geq d_{th}$
- Cost metric favors edges with untrusted code trustworthiness (cost=0);
- and, to a lesser extent, edges with untrusted data trustworthiness (cost=1)
- Define simplifications on output
  - Prune nodes lacking “interesting” descendants
  - Merge “similar” entities
  - Remove repetitions

![Diagram](image-url)
## Campaign Reconstruction Summary

<table>
<thead>
<tr>
<th>Campaign</th>
<th>Entry Points</th>
<th>Programs Executed</th>
<th>Key Files Involved</th>
<th>Exit Points</th>
<th>Correctly Identified</th>
<th>False Positives</th>
<th>Missed Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win-1</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>3</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Win-2</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>18</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lin-1</td>
<td>2</td>
<td>10</td>
<td>6</td>
<td>2</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lin-2</td>
<td>2</td>
<td>20</td>
<td>11</td>
<td>4</td>
<td>37</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lin-3</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>18</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bsd-1</td>
<td>4</td>
<td>13</td>
<td>9</td>
<td>2</td>
<td>13</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>Bsd-2</td>
<td>2</td>
<td>10</td>
<td>7</td>
<td>3</td>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bsd-3</td>
<td>4</td>
<td>14</td>
<td>7</td>
<td>1</td>
<td>26</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>19</strong></td>
<td><strong>89</strong></td>
<td><strong>57</strong></td>
<td><strong>24</strong></td>
<td><strong>174</strong></td>
<td><strong>15</strong></td>
<td><strong>1</strong></td>
</tr>
</tbody>
</table>
Generated Graph for Scenario Bsd-3

The graph illustrates the execution flow of a scenario involving the usage of various commands and files. Here is a textual representation of the main actions:

1. `receive` at 128.55.12.167
2. `fork`
3. `fork`
4. `fork`
5. `fork`
6. `fork`
7. `fork`
8. `fork`
9. `fork`
10. `write`
11. `fork`
12. `fork`
13. `read`
14. `write`
15. `write`
16. `write`
17. `fork`
18. `read`
19. `fork`
20. `read`
21. `fork`
22. `fork`
23. `execute`
24. `write`
25. `fork`
26. `fork`
27. `execute`
28. `read`
29. `receive`
30. `fork`
31. `fork`
32. `write`
33. `fork`
34. `write`
35. `fork`
36. `write`
37. `fork`
38. `write`
39. `fork`
40. `fork`
41. `write`
42. `fork`
43. `execute`
44. `read`
45. `send`

The nodes represent commands and files involved in the scenario. For instance, commands like `scp`, `bash`, `sudo`, `whoami`, `date`, `ps`, `hostname`, `ls`, `vi`, `sh`, `whoami`, `uname`, `ls`, and `archiver` are depicted, along with their interrelations through the execution flow.
Generated Graph for Scenario Lin-2
## Forward Analysis Selectivity

<table>
<thead>
<tr>
<th>Campaign</th>
<th>Initial # of Events</th>
<th>Final # of Events</th>
<th>Reduction Factor</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Single t-tag</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Split t-tag</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SLEUTH Simplif.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td>Win-1</td>
<td>100 K</td>
<td>51</td>
<td>4.4x</td>
<td>1951x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1394x</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.4x</td>
<td></td>
</tr>
<tr>
<td>Win-2</td>
<td>401 K</td>
<td>28</td>
<td>3.6x</td>
<td>14352x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>552x</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>26x</td>
<td></td>
</tr>
<tr>
<td>Lin-2</td>
<td>38.5 M</td>
<td>130</td>
<td>7.3x</td>
<td>297100x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2971x</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>100x</td>
<td></td>
</tr>
<tr>
<td>Lin-3</td>
<td>19.3 M</td>
<td>45</td>
<td>7.6x</td>
<td>430048x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1208x</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>356x</td>
<td></td>
</tr>
<tr>
<td>Bsd-2</td>
<td>5.86 M</td>
<td>39</td>
<td>1.9x</td>
<td>150202x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>689x</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>218x</td>
<td></td>
</tr>
<tr>
<td>Bsd-3</td>
<td>5.68 M</td>
<td>45</td>
<td>6.7x</td>
<td>125800x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>740x</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>170x</td>
<td></td>
</tr>
<tr>
<td><strong>Average Reduction</strong></td>
<td></td>
<td></td>
<td>4.68x</td>
<td>54517x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1305x</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>41.8x</td>
<td></td>
</tr>
</tbody>
</table>
### Memory Use and Runtime Performance

<table>
<thead>
<tr>
<th>Campaign</th>
<th>Events</th>
<th>Memory Usage (MB)</th>
<th>Bytes/event</th>
<th>Duration (hh:mm:ss)</th>
<th>Runtime (Time)</th>
<th>Speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win-1</td>
<td>100K</td>
<td>3</td>
<td>30</td>
<td>06:22:42</td>
<td>1.19 s</td>
<td>19.3 K</td>
</tr>
<tr>
<td>Win-2</td>
<td>400K</td>
<td>10</td>
<td>25</td>
<td>19:43:46</td>
<td>2.13 s</td>
<td>33.3 K</td>
</tr>
<tr>
<td>Win-Mean</td>
<td></td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td>26.3 K</td>
</tr>
<tr>
<td>Lin-1</td>
<td>2.7M</td>
<td>26</td>
<td>10</td>
<td>07:59:26</td>
<td>8.71 s</td>
<td>3.3 K</td>
</tr>
<tr>
<td>Lin-2</td>
<td>38.5M</td>
<td>329</td>
<td>9</td>
<td>79:06:39</td>
<td>114.14s</td>
<td>2.5 K</td>
</tr>
<tr>
<td>Lin-3</td>
<td>19.3M</td>
<td>175</td>
<td>9</td>
<td>79:05:13</td>
<td>74.14 s</td>
<td>3.9 K</td>
</tr>
<tr>
<td>Lin-Mean</td>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td>3.2 K</td>
</tr>
</tbody>
</table>
Related Work

Intrusion detection: Numerous anomaly & misuse detection techniques developed since 80s.

- **Sleuth** advances: novel use of provenance and policies to obtain *application-independent, training-free* detection with very low false positive rate.

Alert correlation: Link alarms using statistical [Qin03], graph-based clustering [Wang08, Pei16], attack specifications [Ning03], and so on

- In contrast, **Sleuth** uses provenance tags and policies to obtain accurate, analyst-friendly scenario descriptions

“Backtracking Intrusions:” Backtracker, Taser, Forensix, ...

- Target forensic analysis, assisted by external detectors.
- **Sleuth** targets fully automated, real-time scenario construction with built-in detectors

Tackling dependence explosion: Orthogonal to (and can benefit) **Sleuth**.

- Fine-grained taint-tracking
- Forensics-targeted: BEEP, ProTracer, ...

Information flow control: [Biba, Bell-LaPadula, PIP, SPIF, ...]

- Goal is to block illegal flow, while minimizing failures.
- In contrast, **Sleuth** needs to distinguish attacks from benign policy violations.
Summary

- Presented techniques that a security analyst can use to understand an ongoing attack campaign, and respond in real-time.

  Automatically generated visual representation that compactly summarizes an ongoing campaign

- Experiments show high accuracy and performance for **SLEUTH**

- Effectiveness evaluated using realistic adversarial engagements.

  Key point: Given millions of events in an unknown environment, **SLEUTH** consistently managed to be spot-on, zooming in on the 0.01% or less of the events actually involved in attacks.