An Empirical Study of Automatic Event Reconstruction Systems

By

Sundararaman Jeyaraman, Mikhail Atallah

Presented At

The Digital Forensic Research Conference

DFRWS 2006 USA  Lafayette, IN (Aug 14th - 16th)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

http://dfrws.org
An Empirical Study of Automatic Event Reconstruction Systems

Sundar Jeyaraman
Mikhail J Atallah
Event Reconstruction

- Identify the underlying conditions and chain of events that led to the security event

- Necessary for effective incident response and recovery
Event Reconstruction cont.

• Ex-post evidence
  – Disk, Memory dumps. Network logs
  – TCT, Sleuthkit, Encase, Ethereal...

• Ex-ante logging
  – Audit trails (hopefully tamper proof)
  – Back Tracker, Forensix...
Why an empirical study?

- Guidance for investigators in choosing the right tool
- Likelihood calculation for hypotheses
- Towards standardization and thwarting Trojan Horse Defense [Carney et al. 2004, ]
- Directions for future research
A really quick survey of event reconstruction systems
BackTracker [king et al. 2003]

- At run time:
  - Monitor system objects and events
  - Record dependences between system objects

- Post-mortem:
  - Build dependence graph
  - Traverse graph to reconstruct the events
Data Flow Analysis
- Sitaraman et al. 2005
  - Dynamic Program Slicing
  - Static Program Slicing

Memory Mapped Files
- Sarmoria et al. 2005
In Summary

- Tracking OS-enabled dependences
  - BackTracker, Forensix, CIDS, Process Labels
  - Improved BackTracker
    - File Offset Intervals

- Tracking process-enabled dependences
  - Improved BackTracker
    - Static Program Slicing
    - Dynamic Program Slicing
  - Memory mapped files
Methodology

• Equivalent ability in tracking causal relationships enabled by the OS.

• Difference arises in the ability to track those enabled by the process address space

• Use *dynamic slicing* to determine false-positives and false-negatives
Reconstruction Systems

- Tracking OS-enabled dependences
  - **BackTracker, Forensix, CIDS, Process Labels**
  - Improved BackTracker
    - File Offset Intervals

- Tracking process-enabled dependences
  - **Improved BackTracker**
    - **Static Program Slicing**
    - **Dynamic Program Slicing**
  - Memory mapped files
Methodology cont.

- A set of applications as a benchmark suite
- Regression test suite for each application
- Metrics
  - Average rate of false-positives
## BenchMark Suite

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gnuPG 1.4.2</td>
<td>GNU’s replacement for PGP</td>
</tr>
<tr>
<td>gnu wget 1.10</td>
<td>Program for retrieving files through HTTP(S), FTP</td>
</tr>
<tr>
<td>find (findutils 4.2.25)</td>
<td>Search for files in a directory hierarchy</td>
</tr>
<tr>
<td>locate (findutils 4.2.25)</td>
<td>List files in a database that matches pattern</td>
</tr>
<tr>
<td>ls (coreutils 4.5.3)</td>
<td>List directory contents</td>
</tr>
<tr>
<td>cp (coreutils 4.5.3)</td>
<td>Copy files</td>
</tr>
<tr>
<td>wc (coreutils(4.5.3))</td>
<td>Print the number of bytes, words and lines in a file</td>
</tr>
<tr>
<td>tar 1.15.1</td>
<td>Archiving software</td>
</tr>
<tr>
<td>gzip 1.3.3</td>
<td>A popular data compression program</td>
</tr>
<tr>
<td>grep 2.5.1</td>
<td>Search files for a given input pattern</td>
</tr>
</tbody>
</table>
Experimentation

- Dynamic slicing implemented using PIN
- Static Slicing implemented using CodeSurfer
- Approx. 100,000 system calls and 11 Billion instructions executed as part of the test cases
Avg rate of False-positives

Avg Rate of False positives

- gpg
- wget
- find
- locate
- tar
- gzip
- wc
- ls
- cp
- grep

BackTracker
Static Slicing

DFRWS 2006
## Overhead of Dynamic Slicing

<table>
<thead>
<tr>
<th>Application</th>
<th>CPU overhead</th>
<th>Wallclock overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpg</td>
<td>8458</td>
<td>7646</td>
</tr>
<tr>
<td>wget</td>
<td>4933</td>
<td>45</td>
</tr>
<tr>
<td>find</td>
<td>648</td>
<td>648</td>
</tr>
<tr>
<td>locate</td>
<td>43298</td>
<td>48</td>
</tr>
<tr>
<td>tar</td>
<td>12808</td>
<td>14149</td>
</tr>
<tr>
<td>gzip</td>
<td>32894</td>
<td>1510</td>
</tr>
<tr>
<td>wc</td>
<td>28719</td>
<td>760</td>
</tr>
<tr>
<td>ls</td>
<td>22153</td>
<td>8140</td>
</tr>
<tr>
<td>cp</td>
<td>10502</td>
<td>11525</td>
</tr>
<tr>
<td>grep</td>
<td>53</td>
<td><strong>57</strong></td>
</tr>
</tbody>
</table>
Limitations & Discussion

- Incomplete coverage of reconstruction systems
- Limitations of benchmark suite
  - No multi-threaded applications
  - No application > 100K LOC
- No statement coverage statistics for testcases
- Implicit dependences
- Better analysis of the results
Comments/Questions/Brickbats?

Sundar Jeyaraman

jsr@cerias.purdue.edu
while (pending_dirs)
{
    extract_files_from_dir(pending_dirs);
    print_files();
}

`ls dir1 dir2 dir3`