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
wget exploit
httpd
bash httpd exploit /etc/passwd /etc/shadow sshd bash
bash
bash
nmap

File Offset intervals
wget
httpd
bash
httpd
exploit
/etc/passwd
/tmp/*
Data Flow Analysis
• Sitaraman et al. 2005
• Dynamic Program Slicing
• Static Program Slicing
Memory Mapped Files
• Sarmoria et al. 2005

DFRWS 2006
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

DFRWS 2006
Methodology cont.

- A set of applications as a benchmark suite
- Regression test suite for each application
- Metrics
  - Average rate of false-positives
<table>
<thead>
<tr>
<th><strong>BenchMark Suite</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>gnuPG 1.4.2</td>
</tr>
<tr>
<td>gnu wget 1.10</td>
</tr>
<tr>
<td>find (findutils 4.2.25)</td>
</tr>
<tr>
<td>locate (findutils 4.2.25)</td>
</tr>
<tr>
<td>ls (coreutils 4.5.3)</td>
</tr>
<tr>
<td>cp (coreutils 4.5.3)</td>
</tr>
<tr>
<td>wc (coreutils(4.5.3))</td>
</tr>
<tr>
<td>tar 1.15.1</td>
</tr>
<tr>
<td>gzip 1.3.3</td>
</tr>
<tr>
<td>grep 2.5.1</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

<table>
<thead>
<tr>
<th>Command</th>
<th>BackTracker</th>
<th>Static Slicing</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpg</td>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>wget</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>find</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>locate</td>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>tar</td>
<td>100</td>
<td>80</td>
</tr>
<tr>
<td>gzip</td>
<td>120</td>
<td>100</td>
</tr>
<tr>
<td>wc</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>ls</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>cp</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>grep</td>
<td>80</td>
<td>60</td>
</tr>
</tbody>
</table>

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>57</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
Iterative and Recursive Behavior

```
while (pending_dirs)
{
    extract_files_from_dir(pending_dirs);
    print_files();
}
```

```
ls dir1 dir2 dir3

```

```
l | l | l
-----|---|---
f1  |  d1
f2  |  f3
```