Introduction to Autonomic Computing

DCSL Presentation
Matthew Tan Creti
03/06/13

IBM’s Properties of Autonomic Computing

1. Self-Configuring – configures itself according to high level goals
2. Self-Optimization – optimizes its use of resources
3. Self-Healing – detects and diagnoses problems
4. Self-Protection – protects itself
Is Autonomic Computing a Buzzword or Legitimate Field of Study?

• Argument for buzzword, traditional examples:
  – DHCP network protocol is self-configuring
  – Query optimizer in a database system is self-optimizing
  – Routing protocol in network is self-healing
  – Intrusion detection and response system is self-protecting

• Self-management community autonomic definition [Huebscher 08]
  – the system changes itself to reflect the current environmental context; reflecting dynamism in the system
  – it exhibits more than one of the self-management properties

• For example, TCP is self-optimizing and self-healing, however it is not autonomic because it is a fixed protocol

Overview

• Introduction
• Two examples of self-protecting
  – Rollback and huddle
  – Automatically patching errors in deployed software
• Conclusion
Architectural Support for Automated Software Attack Detection, Recovery, and Prevention

Embedded and Ubiquitous Computing (EUC) 2008
Jesse Sathre, Alex Baumgarten, and Joseph Zambreno

Problem

- Problem: when an intrusion is detected on a node, how to gracefully respond and recover?
- One approach is to halt execution of the application and send an alert to the administrator
- This is not a good solution for systems that need to operate autonomously (e.g., embedded networks)
Approach

- Approach: “rollback and huddle”
  - System maintains checkpoints
  - Upon attack detection the application is restarted at the last known safe checkpoint (rollback)
  - Additional hw/sw module is loaded protect system from attack (huddle)

Lightweight Monitoring

- During normal execution a lightweight detection monitoring is used
- The example use in paper is a “secondary stack”
  - When a function is called, the return address is stored on both stacks, along with a timestamp
  - On function return both addresses are checked
  - If they do not match the timestamp is used to indicate the time of the attack
Continuous Checkpointing

- As program executes, state is saved to logs
- Non deterministic IO not logged so deterministic replay not possible
- Reduces amount that must be logged, but at the cost of not being able to exactly reproduce attacks

Heavy Weight Monitoring

- NOP place holders are inserted into the code at compile time
- After rollback, heavy weight monitoring inserted into the NOPs
- Examples are bounds-checking arrays and type-checking pointers
- The assumption is that program degradation is better than program termination
**Heavyweight Protection Unit (HPU)**

- HPU must have write access to the instruction space
- Two different methods of applying protection
  1. HPU contains instruction templates that protect against known types of attacks, these are programmed into the NOPs
  2. The NOPs are already programed with encrypted instructions that are decrypted to enable heavyweight protection

**Evaluation**

- Behavioral model of the LPU and HCU modules in PTLSim Classic
- Simulates a single-threaded application with realistic branch prediction and cache behaviors

- L1 Instruction Cache: 32 KB, 4-way set associative, 1 cycle latency
- L1 Data Cache: 16 KB, 4-way set associative, 1 cycle latency
- L2 Cache: 256 KB, 16-way set associative, 6 cycle latency
- L3 Cache: 4 MB, 32-way set associative, 16 cycle latency
- Main Memory: 140 cycle latency.
**CPU Overhead**

- Two sources of overhead: lightweight monitoring and checkpoint logging
- In CRC and blowfish.encode there are many small function calls so monitoring dominates overhead
- In quicksort, tiff2rgba, and tiffmedian checkpoint logging dominates overhead

**Maximum Rollback Distance**

- Entries dictates how many writes to unique address can happen before a log (checkpoint) is full
- Log is the number of checkpoints that can be recorded
- CRC and sha make few memory writes
- Trend is that a smaller number of large checkpoint logs leads to a longer achievable rollback distance
Real World Vulnerabilities

- Taken from US-CERT Cyber Security Bulletin SB04-357

<table>
<thead>
<tr>
<th>Application</th>
<th>Version</th>
<th>Cycles from input to detection</th>
<th>Recoverable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>vItemList</td>
<td>2.6.6</td>
<td>6,745,763</td>
<td>Yes</td>
</tr>
<tr>
<td>ringtoneetools</td>
<td>2.22</td>
<td>3,640,082</td>
<td>Yes</td>
</tr>
<tr>
<td>mp3lplayer</td>
<td>1.0pre5</td>
<td>2,890,044</td>
<td>Yes</td>
</tr>
<tr>
<td>evi2xml</td>
<td>0.5.1</td>
<td>1,841,282</td>
<td>Yes</td>
</tr>
<tr>
<td>zfax</td>
<td>3.0.4</td>
<td>727,625</td>
<td>Yes</td>
</tr>
<tr>
<td>bash2ppm</td>
<td>0.6</td>
<td>665,687</td>
<td>Yes</td>
</tr>
<tr>
<td>jpeg2avri</td>
<td>1.5</td>
<td>432,591</td>
<td>Yes</td>
</tr>
<tr>
<td>elfread</td>
<td>0.0.3</td>
<td>304,964</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Paper 1 Conclusion

- Self-protection using rollback and huddle
- Insert heavy weight protections and resume execution for safe checkpoint
Self-defending software: Automatically patching errors in deployed software

SOSP 2009
Saman Amarasinghe, Jonathan Bachrach, Michael Carbin, Sung Kim, Samuel Larsen, Carlos Pacheco, Jeff Perkins, Martin Rinard, Frank Sherwood, Stelios Sidiroglou, Greg Sullivan, Weng-Fai Wong, Yoav Zibin