Recovery from Intrusions in PaaS Clouds

Miguel P. Correia
(joint work with Dário Nascimento)

IFIP WG Meeting
Jan. 2015
Motivation

Number of **critical applications** in the Cloud is increasing

Number of **Intrusions** in these applications is increasing
Motivation

Intrusions compromise:
• Integrity
• Availability
• Confidentiality

Intrusion/fault causes:
• Software flaws
• Configuration and usage mistakes
• Corrupted legitimate requests (e.g. SQL injection)
Motivation

• Personal motivation:
• I’ve been working on masking faults and intrusions for 15 years
• Industry seems not to care
• Industry does care about recovering from intrusions/faults when they happen
Recover the application’s integrity when intrusions happen
Related Work

Backups
works but removes both bad and good actions
Related Work

Intrusion recovery: remove bad, not good actions
- Operating systems: Taser, Retro
- Databases: ITDB, Phoenix
- Web applications: Goel et. al, Warp, Aire
- Others: Undo for Operators

Limitations:
- All require setup and configuration
- Max. complexity: 1 app server, 1 database instance
- Cause application downtime during recovery
Objective

Platform as a Service (PaaS)

• Cloud service = to run applications
• Consumer develops application to run in that environment, using
  – Supported languages, e.g., Java, Python, Go, PHP
  – Supported components, e.g., SQL/noSQL databases, load balancers
Objective

Intrusion recovery system for PaaS

- Supported by the PaaS: available without setup
- Remove the intrusion effects
- Support applications deployed in various instances
- Avoid application downtime
- Cost effective
- Recover fast
Architecture

Shuttle

User requests

Legend:
- Scalling
- User Requests
- Replay Requests
- Control Messages
Architecture

Replay Process

1. Identify the malicious actions
2. Start new application and database instances
3. Load a snapshot previous to intrusion instant
   Create a new branch
4. Replay requests
   Database operations shall replay in same order as original
5. Block incoming requests; replay last requests
6. Change branch
## Replay Modes

<table>
<thead>
<tr>
<th></th>
<th>Full-Replay</th>
<th>Selective-Replay</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Cluster (Serial)</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Clustered</td>
<td>✔</td>
<td>✗</td>
</tr>
</tbody>
</table>

**Full-Replay:** Replay every operation after snapshot  
**Selective-Replay:** Replay only affected (tainted) operations  

**Serial:** Replay all dependency graph sequentially  
**Clustered:** Independent clusters can be replayed concurrently

![Diagram of replay modes](image)
Evaluation

Environment

• Amazon EC2, c3.xlarge instances, Gb Ethernet
• WildFly (formerly JBoss) application servers
• Voldemort database
• Ask Q&A application; data from Stack Exchange
Evaluation

**Accuracy** with intrusion scenarios:
1. Malicious requests
2. Software vulnerabilities
3. External channels (e.g. SSH)

<table>
<thead>
<tr>
<th></th>
<th>#tampered intrusion</th>
<th>#tainted</th>
<th>#replayed (selective rep.)</th>
<th>#replayed (full replay)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>110</td>
<td>0</td>
<td>[0, 605]</td>
<td>&gt; 38 620</td>
</tr>
<tr>
<td>1b</td>
<td>58</td>
<td>14</td>
<td>[0, 379]</td>
<td>&gt; 38 620</td>
</tr>
<tr>
<td>1c</td>
<td>48</td>
<td>52</td>
<td>[0, 253]</td>
<td>&gt; 38 620</td>
</tr>
<tr>
<td>2a</td>
<td>4338</td>
<td>0</td>
<td></td>
<td>&gt; 38 620</td>
</tr>
<tr>
<td>2b</td>
<td>18 286</td>
<td>1 278</td>
<td></td>
<td>&gt; 38 620</td>
</tr>
<tr>
<td>3</td>
<td>&gt; 2 000</td>
<td>-</td>
<td></td>
<td>&gt; 38 620</td>
</tr>
</tbody>
</table>
## Evaluation

### Performance overhead in normal execution

<table>
<thead>
<tr>
<th></th>
<th>Workload A</th>
<th>Workload B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shuttle</td>
<td>6325 ops/sec [5.78 ms]</td>
<td>15346 ops/sec [3.62 ms]</td>
</tr>
<tr>
<td>No Shuttle</td>
<td>7148 ops/sec [5.07 ms]</td>
<td>17821 ops/sec [3.01 ms]</td>
</tr>
<tr>
<td>overhead</td>
<td>13% [14%]</td>
<td>16% [20%]</td>
</tr>
</tbody>
</table>
Evaluation Performance

Recovery time
1 million requests
Evaluation Performance

Restrain duration

![Graph showing the change in requests per second over time. The graph indicates a drop in requests after a certain point, with a note indicating the restrain duration as 46 seconds.]

**Restrain**: 46 seconds
## Evaluation Performance

### Storage overhead for 1 million requests

<table>
<thead>
<tr>
<th></th>
<th># objects</th>
<th>size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shuttle Storage:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request</td>
<td>1 million</td>
<td>212</td>
</tr>
<tr>
<td>Response</td>
<td>1 million</td>
<td>8,967</td>
</tr>
<tr>
<td>Start/end timestamps</td>
<td>2 million</td>
<td>16</td>
</tr>
<tr>
<td>Keys</td>
<td>137 million</td>
<td>488</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>9,684</strong></td>
</tr>
<tr>
<td><strong>Database node:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version List</td>
<td>14,593</td>
<td>1.4</td>
</tr>
<tr>
<td>Operation list</td>
<td>9 million</td>
<td>277</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>282</td>
</tr>
<tr>
<td><strong>Manager:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graph</td>
<td>1 million</td>
<td>718</td>
</tr>
</tbody>
</table>
Conclusion

• New intrusion recovery service to be integrated in PaaS offerings

• Supports applications running in various instances backed by distributed databases

• Leverages the resource elasticity and pay-per-use model to reduce the recovery time and costs