Enhancing Fault/Intrusion Tolerance through Design and Configuration Diversity

Alysson Bessani, Alessandro Daidone, Ilir Gashi, Rafael Obelheiro, Paulo Sousa, Vladimir Stankovic

29th June 2009, Lisbon, Portugal

Outline

- Team and motivation
- FOREVER service
- Configuration diversity rules
- Preliminary assessment of the FOREVER service
- Conclusions and future work
The Team

- **FOREVER** – a mini-project inside the EU NoE ReSIST [http://www.resist-noe.org](http://www.resist-noe.org)
- **Institutions**
  - Universidade de Lisboa (Portugal)
  - City University London (UK)
  - Università di Pisa (Italy)
  - Universidade do Estado de Santa Catarina (Brazil) – Affiliate member
  - Universität Erlangen-Nürnberg (Germany) – Affiliate member
- **People**
  - Alysson Bessani @ Lisboa
  - Hans Reiser @ Lisboa
  - Paulo Sousa @ Lisboa
  - Ilir Gashi @ City
  - Vladimir Stankovic @ City
  - Alessandro Daidone @ Pisa
  - Rafael Obelheiro @ Santa Catarina
  - Tobias Distler @ Erlangen-Nürnberg
  - Rüdiger Kapitza @ Erlangen-Nürnberg

Addressing WRAITS Topics

- **Topics:**
  - automatic recovery and response techniques
  - use of Byzantine fault-tolerant algorithms in IT
  - diversity and failure independence
- “The workshop will be especially interested in ‘practical intrusion-tolerant systems’”
- “How to build information systems that are inherently resilient to intrusions“
- “Papers can present ongoing work and/or speculative/futuristic ideas”
Motivations

• Fault/intrusion tolerance is commonly the only viable way of improving the system dependability and (possibly) security.
• Designers are in need of advice about the architectural evolution of fault/intrusion-tolerant systems
  – Many solutions exist, but the current solutions concern (only) a specific snapshot in time, without considering how the system should evolve.
• Fault/intrusion tolerant systems need to counteract the evolution of threats and exploits against a running system.

Motivations (cont.)

• The defensive measures in the current practice are largely reactive.
  – E.g. issuance of patches
  – “at-risk-time” is (unnecessarily) long
• Proactive methods are needed to counteract the evolution of faults, threats and vulnerabilities.
  – With proactive methods, design novelty is used as a defence.
• We need a unifying, complementary approach: reactive approaches ought to be enhanced with a more dynamic response to evolving threats and vulnerabilities.
The FOREVER Service
(Fault/intrusiOn REmoVal through Evolution & Recovery)

- Byzantine fault-tolerant protocols against arbitrary failures have limited utility
  - They are “only” useful to delay corruptions.
- While a higher number of replicas allows for tolerating more compromised replicas, it also means that there are more individual replicas that need to be different from each other.
  - E.g., 3f+1 and f=1 → 4 (diverse) replicas needed
  - But, 3f+1 and f=2 → 7 (diverse) replicas needed.
- FOREVER’s main goal – increasing the number of tolerated arbitrary faults/vulnerabilities, without increasing the number of replicas.

The FOREVER Service (cont.)

- The ambitious goal is to be achieved through
  - Recovery
    - Event-triggered (when malicious behavior is detected or suspected)
    - Time-triggered (every replica is rejuvenated periodically)
  - Evolution
    - Through the use of configuration diversity rules
    - Recovered replicas are different from previous incarnations
    - This makes FOREVER different from the other recovery services.
The FOREVER Architecture

Hybrid model and architecture

Internet
(clients, attackers, …)

Fault/Tolerance application

BFT replication library

FOREVER

- asynchronous and exposed to arbitrary faults
- synchronous and secure

Can be compromised!

Cannot be compromised!

Diversity Management

- **Offline** diversity generation
  - Pool of pre-built OS images (e.g., Linux, OpenBSD)
  - Different OS image started in each recovery
    - Recent studies showed some off-the-shelf sw have **low** number of common faults/vulnerabilities\(^1\).
    - FOREVER selects the OS image that is **less similar** than the OS images running in the remaining replicas.

- **Online** diversity generation
  - FOREVER applies a set of **configuration diversity rules** to the selected OS image.

---

Configuration Diversity Rules

- We produced a set of configuration diversity rules, which aim to enhance the resilience of software in between recoveries.
- The rules were generated using three approaches:
  - **Bottom-up** – scrutinising the implementations of the OSs and applications
  - **Top-down** – exploring reported vulnerabilities
  - **Lateral** – reviewing existing literature of configuration rules for protection against "malicious" behaviour

1 Full description of rules can be found at [http://www.csr.city.ac.uk/people/ilir.gashi/ConfigDiv/](http://www.csr.city.ac.uk/people/ilir.gashi/ConfigDiv/)

<table>
<thead>
<tr>
<th>ID</th>
<th>Rule name</th>
<th>Design Implication</th>
<th>Implication</th>
<th>Security Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Password change</td>
<td>W, B or G</td>
<td>Yes</td>
<td>C</td>
</tr>
<tr>
<td>2.1</td>
<td>Different authentication protocols</td>
<td>W, B or G</td>
<td>Yes</td>
<td>C</td>
</tr>
<tr>
<td>2.2</td>
<td>Different Trusted Third Parties</td>
<td>W, B or G</td>
<td>No</td>
<td>C and A</td>
</tr>
<tr>
<td>3</td>
<td>Different &quot;Factors&quot; in multi-factor authentication methods</td>
<td>W, B or G</td>
<td>Yes</td>
<td>C</td>
</tr>
<tr>
<td>4.1</td>
<td>Address Space Layout Randomisation (ASLR)</td>
<td>W</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Pointer obtipation</td>
<td>W</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Randomisation of global variables and local variables offsets</td>
<td>W</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>4.2</td>
<td>Address Space Partionning</td>
<td>W, B or G</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>4.3</td>
<td>Stack Frame Padding</td>
<td>W, B or G</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>4.4</td>
<td>Basic Block randomising</td>
<td>W, B or G</td>
<td>No</td>
<td>1</td>
</tr>
</tbody>
</table>

| 5.1| Instruction set randomisation     | W, B or G         | No          | 1                 |
| 5.2| Instruction set hardening        | W, B or G         | No          | 1                 |
| 5.3| Instruction Reordering           | W, B or G         | No          | 1                 |
| 6.1| Diverse Linux User IDs (UID)     | W, B or G         | No          | 1 and C           |
| 7  | Change IP addresses of the hosts | B | Yes | C and A |
| 8  | Changing listening port number   | W, B or G         | Yes         | C and A          |
| 9  | Adding or deleting non-functional code | W, B or G | No | 1 |
| 10.1| Varying dynamic libraries and system calls | W | No | 1 |
| 10.2| Varying unique names of system files | W | No | 1 |
| 10.3| Varying magic numbers in certain files (e.g., executables) | W | No | 1 |
FOREVER Service Preliminary Evaluation

- Preliminary assessment of the potential to enhance system resilience.
- Focus on the evaluation of the probability of system failure, through variation of several parameters:
  - The time between recoveries
  - The probability of common vulnerabilities
  - The mean effectiveness of configuration diversity rule-set
- Initial values chosen are pessimistic
  - The common practice when evaluating safety- and reliability-critical systems and/or parameter values are unknown.
- The quantitative evaluation uses a modelling methodology which assumes Multiple Phased Systems.

System Properties/Assumptions

- n = 4 replicas can tolerate up to f = 1 faults
  - n ≥ 3f+1
  - Win2k, Linux, Solaris, FreeBSD
- Both “space” (design diversity) and “time” (configuration diversity rules) diversity is considered.
- Arbitrary faults/vulnerabilities; a binary state space: “failed”, “OK”.
- Common faults/vulnerabilities exist
  - The ratio value used is based on the study¹ using National Vulnerability Database (NVD) data
- Proactive, sequential, fault-free recovery strategy has been assumed.

Measures of Interest and Parameters

- \( p_F \) - System failure probability, i.e., probability of having more than \( f \) failed replicas.
- \( t \) - Mission time; we are interested in investigating how system failure probability changes over time.
- \( T_P \) - Recovery period, based on the waiting time \( T_W \) between the recovery of replica \( i \) and the recovery of replica \( i+1 \).
  \[ T_P = (T_R + T_W) \times n \]
- \( \delta_x \) - Base value of the penalty of replica failure rate, if diversity is not applied.
  - Discovery of vulnerabilities is assumed to be progressing
- \( \delta_{ij} \) - Probability of common faults/vulnerabilities among different replicas
  - Since \( n = 4 \), the overall system fails as soon as one common fault/vulnerability, affecting a pair of replicas, occurs.
- \( \delta_x \) - Mean value for the effectiveness of the configuration diversity rule-set.

Parameter Values and Common Faults Prob.

<table>
<thead>
<tr>
<th>Study</th>
<th>( T_W )</th>
<th>( \delta_x )</th>
<th>( \delta_x )</th>
<th>( \delta_{ij} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{0, 360, 480, 840}</td>
<td>0</td>
<td>0</td>
<td>( \delta_{ij} )</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>(0, 10^{-6}, 5 \times 10^{-7}, 10^{-7})</td>
<td>0.8</td>
<td>( \delta_{ij} )</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>{0, 1, 5, 10} \times \delta_{ij}</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>10^{-4}</td>
<td>{0, 0.2, 0.4, 0.6, 0.8, 1}</td>
<td>( \delta_{ij} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>O.S.</th>
<th>Vulns</th>
<th>Common Vulnerabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreeBSD</td>
<td>229</td>
<td>Win2K 3</td>
</tr>
<tr>
<td>Solaris</td>
<td>411</td>
<td>Win2K 3</td>
</tr>
<tr>
<td>Linux</td>
<td>437</td>
<td>Win2K 3</td>
</tr>
<tr>
<td>Win2K</td>
<td>347</td>
<td>Win2K 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \delta_{ij} )</th>
<th>FreeBSD</th>
<th>Solaris</th>
<th>Linux</th>
<th>Win2K</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreeBSD</td>
<td></td>
<td>0.029</td>
<td>0.017</td>
<td>0.005</td>
</tr>
<tr>
<td>Solaris</td>
<td>0.029</td>
<td></td>
<td>0.006</td>
<td>0.004</td>
</tr>
<tr>
<td>Linux</td>
<td>0.017</td>
<td>0.006</td>
<td></td>
<td>0.004</td>
</tr>
<tr>
<td>Win2K</td>
<td>0.005</td>
<td>0.004</td>
<td>0.004</td>
<td></td>
</tr>
</tbody>
</table>
Preliminary Evaluation Results (Study 1)
Waiting Time ($T_w$) Sensitivity Analysis

\[ \delta_x = 0 \]
\[ \delta_A = 0 \]

Preliminary Evaluation Results (Study 4)
Rule(s) Effectiveness ($\delta_x$) Sensitivity Analysis

$T_w = 0$
$\delta_A \text{ NOT changed}$
Diversity Configuration Rules - Design Implications

- Interface and input language complexity
- Possibility of introducing new faults/vulnerabilities
- Security issues
- Performance issues
- Data consistency issues

Conclusions

- We performed initial study about system evolution wrt. tolerating ever-growing and ever-present threats.
- We defined a set of configuration diversity rules to be used as part of the “preventive” FOREVER architecture
  – FOREVER uses both reactive and proactive recovery
- FOREVER uses online and offline diversity generation mechanisms
- Performed preliminary evaluation of the FOREVER
Future Work

- Extending configuration diversity rules
  - Possibility of using an alternative categorisation
- Developing proof-of-concept implementations
  - Possibility of using fault-injection
- Extending/improving the evaluation
  - Include the reactive recoveries; and thus e.g. explore trade-off between reactive and proactive strategies.
  - More precise evaluation of the effectiveness of the configuration diversity rules (currently, only a single value represents the effectiveness).
  - Possibility to compare with other (e.g., non-redundant) architecture
- Evaluating the effects on the system performance
  - Aiding the analysis of the “eternal” trade-off between the improvement of dependability/security and performance.

Thank you!