The Design of a Generic Intrusion-Tolerant Architecture for Web Servers

Ayda Saidane, Vincent Nicomette, and Yves Deswarte, Member, IEEE
Outline

• INTRODUCTION
• AN INTRUSION-TOLERANT WEB SERVER
• DETECTION MECHANISMS
• ALERT MANAGER
• PERFORMANCE MEASUREMENTS
• CONCLUSION
INTRODUCTION

• Everybody agrees now that the Internet has become essential in everyday life.

• A growth of malicious activity in the Internet.

• More and more vulnerabilities are discovered, and nearly every day, new security advisories are published.
AN INTRUSION-TOLERANT WEB SERVER

Diagram depicting a network architecture with proxies, mediators, a leader, an adjudicator, and a database. The Internet is connected to a firewall, which leads to the proxies network. The Web servers network includes mediators for Linux/i386, Apache, Solaris/Sparc NetEnterprise, Windows XP/i386 IIS, and MacOSX/PowerPC Apache. The leaders and adjudicators are connected to the database, which is described as an accidental fault tolerant database.
AN INTRUSION-TOLERANT WEB SERVER (2/4)

• The architecture is based on the principles of redundancy and diversification.

  – **Redundancy** is used to increase system availability.

  – **Diversification** is used to increase independence between redundant subsystems.
• Adaptive Redundancy Level
  – In order to minimize the performance degradation of the system.

  – The **regime** is the number of Web servers that process each client request.
AN INTRUSION-TOLERANT WEB SERVER (4/4)

Duplex and triplex regimes.
• **Agreement Protocol**
  
  – It is used to validate server responses when the system is running in a *nonsimplex* regime and is executed by the leader.
Detection Mechanisms(2/8)

• **Intrusion Detection**
  – **SNORT**
    • an open source software based on misuse detection.
  – **EMERALD**
    • developed at SRI International
    • which combines misuse detection and anomaly detection.
Detection Mechanisms (3/8)

• Challenge/Response Protocol

  – The IDSs are efficient in detecting known attacks but are less efficient in detecting new attacks with a slow propagation or low frequency.
Detection Mechanisms (4/8)
Detection Mechanisms (5/8)

• The number of challenges should be such that
  \[ M > \frac{fc}{fr} \]
  
  – 1. \( fc \): the frequency of the CRP for the same file
  
  – 2. \( fr \): the reboot frequency.
Detection Mechanisms(6/8)

• **Runtime Verification**
  
  – A runtime verifier checks the behavior of each proxy during its execution.

  – This technique detects with good credibility any injection of malicious code in the proxies.
Detection Mechanisms (7/8)

State machine for the leader.
Detection Mechanisms (8/8)

• Database Access

Connections between the leader, the adjudicator, and the mediator.
Alert Manager (1/2)

- illustrates the proxy architecture
Alert Manager (2/2)
PERFORMANCE MEASUREMENTS (1/3)

• Performances According to the Regime

<table>
<thead>
<tr>
<th></th>
<th>Direct</th>
<th>Simplex</th>
<th>Duplex</th>
<th>Triplex</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 byte</td>
<td>0.0037 sec</td>
<td>0.0074 sec</td>
<td>0.0087 sec</td>
<td>0.0096 sec</td>
</tr>
<tr>
<td>44Kb</td>
<td>0.0115 sec</td>
<td>0.0145 sec</td>
<td>0.0167 sec</td>
<td>0.0170 sec</td>
</tr>
<tr>
<td>1Mb</td>
<td>0.14 sec</td>
<td>0.316 sec</td>
<td>0.321 sec</td>
<td>0.322 sec</td>
</tr>
</tbody>
</table>
PERFORMANCE MEASUREMENTS (2/3)

• Performance of Database Accesses

<table>
<thead>
<tr>
<th></th>
<th>$GPT_{DB}$</th>
<th>APT (leader)</th>
<th>$GPT_{HTTP}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>using our database library</td>
<td>0.028 sec</td>
<td>0.038 sec</td>
<td>0.045 sec</td>
</tr>
<tr>
<td>using a standard MySQL library</td>
<td>-</td>
<td>0.015 sec</td>
<td>0.020 sec</td>
</tr>
</tbody>
</table>
PERFORMANCE MEASUREMENTS (3/3)

• Performance of Isolation and Reboot of a Corrupted Server

<table>
<thead>
<tr>
<th></th>
<th>APT</th>
<th>TI</th>
<th>$GPT_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.341 sec</td>
<td>0.344 sec</td>
<td>73.2 sec</td>
</tr>
<tr>
<td>Min</td>
<td>0.00616 sec</td>
<td>0.00876 sec</td>
<td>70.3 sec</td>
</tr>
<tr>
<td>Max</td>
<td>1.010 sec</td>
<td>1.015 sec</td>
<td>75.8 sec</td>
</tr>
</tbody>
</table>
CONCLUSION

• In this paper propose a generic intrusion tolerant architecture based on redundancy and diversification.

• The efficiency of intrusion tolerance is strongly dependent on the deployed detection mechanisms.