Robustness and Security Testing in SOA
Challenges & Opportunities

Marco Vieira
mvieira@dei.uc.pt
University of Coimbra – Portugal

Outline
- The SOA trend...
- Robustness and security testing in Web Services
- Are Web Services robust and/or secure?
- Is it done? No, there are many challenges!

The SOA trend...
- Service Oriented Architectures (SOA)
  - Among the most important trends in modern software development
- Architectural style
  - Heterogeneous applications that exchange data
- Functionality decomposed into units (services)
  - Distributed over a network
- Supports business processes
  - Loosely coupled to their underlying implementations
  - Support business- and mission-critical apps

Web Services
- Key support for SOA
  - Provide a simple interface between a provider and a consumer

Web Services composition

Robustness and security testing
- Well known concepts!!!
  - But, are they being applied in the WS context?
- Robustness Testing
  - Test a system/component focusing on limit conditions and invalid (out-of-domain) inputs
  - Looks for input validation problems
- Security (penetration) Testing
  - Submit malicious inputs to the service under testing
  - Inputs may be valid in the input domain
  - Looks for input values used in such way that may create security problems
Robustness testing tools

- Well known tools
  - Ballista (Koopman and DeVale 1999)
  - MAFALDA (Rodríguez et al. 1999)
- Concept has been extended to the context of Web Services
  - e.g. wsrbench

Security testing tools

- HP WebInspect™
- IBM WebSphere
- WSDigger
- WSFuzzer

Questions are...

- Are Web Services and SOA-based apps robust and/or secure?
- Is testing being applied adequately?

What is the quality of the services that are being used to support business- and mission-critical apps?

Robustness testing case study

- 250 publicly available web services
  - More than 1200 operations
  - Obtained using Seekda – random selection
  - Tested using wsrbench
  - 420375 responses – double-checked by two developers

Security testing case study

- 300 Web Services tested - randomly selected

<table>
<thead>
<tr>
<th>Vulnerability Types</th>
<th>VS1.1</th>
<th>VS1.2</th>
<th>VS2</th>
<th>VS3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Injection</td>
<td>21</td>
<td>38</td>
<td>225</td>
<td>38</td>
</tr>
<tr>
<td>XPath Injection</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Code Execution</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Possible Parameter Based Buffer Overflow</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Possible Username or Password Disclosure</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>47</td>
</tr>
<tr>
<td>Possible Server Path Disclosure</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Total</td>
<td>238</td>
<td>40</td>
<td>216</td>
<td>40</td>
</tr>
</tbody>
</table>

Examples of SQL Injection vulnerability

```java
public String auth(String login, String pass)
    throws SQLException {
    String sql = "SELECT * FROM users WHERE " +
                "username='" + login + "' AND " + "password='" + pass + "$'";
```
SQL Injection without False Positives

VS1.1

103

VS1.2

15

VS3

102

Common vulnerabilities

- SQL Injection (149)
- Possible Server Path Disclosure (16)
- XPath Injection (10)
- Code Execution (1)
- Possible Parameter Based Buffer Overflow (1)
Are Web Services robust and/or secure?

- A large number of problems was observed
- Selecting a tool seems to be a very difficult task
  - e.g. different scanners detect different types of vulnerabilities
  - High false positives rates
  - Low coverage rates
- How effective are vulnerability detection tools?

Is it done?

- Facts:
  - Tools exist, but are not so good
  - Testing concepts are well know, but are not applied
- We need to educate developers
  - The human side of the development process!!

But, we are still missing the real challenges!!

The real challenges...

- Agility in the software development
  - Sparse documentation; no formalism
- Incremental software releases
  - Regression testing
- Dynamic composition of services
  - Runtime testing
- Use of third-party (unknown) software services and middleware
  - Testing the unknown!!

A Framework...

Design-time needs...

- How to support traceability to evolving requirements?
- How to cope with features of agile software development?
- How to cope with the development style trend based on successive software releases?
- How to support the use of open source software?

Run-time needs...

- How to apply testing to dynamic and evolving systems?
- How to monitor dynamic services and infrastructures?
  - How to identify changes in the architecture?
  - How to measure the "unknown"?
- How to test a deployed service without affecting the system behavior?
Thanks for your participation!

Marco Vieira
Center for Informatics and Systems
University of Coimbra
mvieira@dei.uc.pt