Model-Based Vulnerability Testing for Web Applications

F. Lebeau, B. Legeard, F. Peureux, A. Vernotte

FEMTO-ST Institute / University of Franche-Comté
UMR CNRS 6174, 25030 Besancon, France.

Smartesting R&D Center, 25000 Besancon, France.

SecTest 2013, Luxembourg
March 22, 2013
# Table of contents

<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>Current Dynamic Application Security Testing Techniques</td>
</tr>
<tr>
<td>Model-Based Vulnerability Testing</td>
</tr>
<tr>
<td>Detailed Process with Example</td>
</tr>
<tr>
<td>Discussion</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Web evolution and Security

Continued growth and complexity of the internet:

▶ increasing ubiquity of **uses** (banking, e-commerce, social...)
▶ increasing combination of **technologies** (server, client)

→ maintaining **security** is a real challenge

*An urgent issue: Web Application vulnerabilities*

The later a vulnerability is found, the more expensive it is to fix.

⇒ Vulnerability discovery part of the development process.

Two main techniques:

▶ **SAST**: Static Application Security Testing
▶ **DAST**: Dynamic Application Security Testing
Continued growth and complexity of the internet:

- increasing ubiquity of uses (banking, e-commerce, social...)
- increasing combination of technologies (server, client)

→ maintaining security is a real challenge

An urgent issue: Web Application vulnerabilities

The later a vulnerability is found, the more expensive it is to fix.
⇒ Vulnerability discovery part of the development process.

Two main techniques:

- SAST: Static Application Security Testing
- DAST: Dynamic Application Security Testing
Current DAST techniques

Manual/Tool-Based Penetration testing

**Strengths**
- Precise and Reliable to detect design flaws
- Based on insights and experience

**Weaknesses**
- Fastidious (thorough testing for XSS)
- Constant need of manpower and expertise
- Based on insights and experience

Current DAST techniques (2)

Vulnerability Scanners

<table>
<thead>
<tr>
<th>Strengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Point-and-shoot solutions</td>
</tr>
<tr>
<td>▶ Efficient for a majority of technical vulnerabilities</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Suffer from a fair amount of false positives</td>
</tr>
<tr>
<td>▶ Struggle with complex vulnerabilities (Multi-step XSS)</td>
</tr>
<tr>
<td>▶ Struggle with vulnerabilities related to business logic</td>
</tr>
</tbody>
</table>

## Current DAST techniques (3)

### Black-box Fuzzing

<table>
<thead>
<tr>
<th><strong>Strengths</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Low cost solution</td>
<td></td>
</tr>
<tr>
<td>▶ Efficient to spot unattended behaviors (or “black swans”)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Weaknesses</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Weak Oracle (crashes, freezes)</td>
<td></td>
</tr>
<tr>
<td>▶ Improving the Oracle is challenging</td>
<td></td>
</tr>
</tbody>
</table>

**R. McNally et. al., Fuzzing: The State of the Art:**

*DSTO-TN-1043*, DSTO Formal Reports, 2012
Existing DAST techniques (4)

Model-Based Security Testing

**Strengths**
- Efficient to address functional security properties
- Automation capacity
- Handle well application evolution

**Weaknesses**
- Needed effort to provide models
- Needed effort to develop a concretization layer

Model-Based Vulnerability Testing (MBVT)

**MBVT**: Vulnerability testing based on **models** and **test patterns**.

**Goal**: Improve the **accuracy** and **precision** of vulnerability testing.

**Accuracy**

Capability to focus on the **relevant part of the software** (e.g. from a risk assessment point of view) depending on the **targeted vulnerability types**.

**Precision**

Capability to avoid both **false positive** and **false negative**.
This MBVT approach is composed of 4 activities:

1. Test purpose definition
2. Model design
3. Test generation
4. Concretization, test execution and verdict assignment
Multi-step XSS is a challenging vulnerability for automated tools\(^1\).

→ It requires **knowledge** from the targeted application.

This MBVT approach deals with this class of vulnerability, by applying a “Def/Use” approach (All-def criterion).

Experiments have been conducted on WackoPicko.

Vulnerability Test Patterns (vTP) are the **entry-point** of MBVT:

- Based on a study from the ITEA2 DIAMONDS project
- Express testing needs and procedure to highlight a breach

The goal is to **translate** vTP into a machine-readable language.

⇒ Reuse of the *Smartesting Test Purpose Language*:\(^2\):

- Designed for security means
- Textual language based on regular expressions
- Reasons in term of states to be reach and operations to be called

---

\(^2\) J Botella et. al., *MBT of Cryptographic Components, Lessons Learned from Experience*, ICST 2013.
# Test Purpose for Multi-step XSS

<table>
<thead>
<tr>
<th>Name</th>
<th>Multi-step XSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>...</td>
</tr>
<tr>
<td>Objective(s)</td>
<td>Detect if an input can embed malicious datum enabling a Multi-step XSS attack.</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>N/A</td>
</tr>
<tr>
<td>Procedure</td>
<td>Identify a sensible user input, inject the malicious datum <code>&lt;script&gt;alert(rxss)&lt;/script&gt;</code>.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Find the page where the input is rendered, and check if a message box 'rxss' appears.</td>
</tr>
<tr>
<td>Variant(s)</td>
<td>...</td>
</tr>
<tr>
<td>Known Issue(s)</td>
<td>...</td>
</tr>
<tr>
<td>Affiliated vTP</td>
<td>Reflected XSS</td>
</tr>
<tr>
<td>Reference(s)</td>
<td>...</td>
</tr>
</tbody>
</table>
2 - Model Design

MBVT behavioral notation is based on UML4MBT\(^3\).

A **generic class diagram** depicts the structure of the SUT: pages, actions, and in/out data (following the def/use concept).

MBVT also requires:

- **Class Diagrams**: defines the static aspect of the SUT
- **Object Diagrams**: defines the initial state of the SUT
- **State Diagrams**: defines the dynamic of the SUT

\(^3\)F Bouquet et. al., *A subset of precise UML for model-based testing*, 2007.
Modeling: WackoPicko Example

Class Diagram

Statemachine
3 - Test Generation

Test cases are generated using **Smartesting CertifyIt**.

Test generation process is driven by **test purposes** and **models**:

- Test generator unfolds test purposes
- Models give the path to follow in order to reach each stages of a given test purpose

**Result**: A suite of abstract vulnerability test cases
Human Intervention during concretization:

- List of malicious vectors (xml file)
- Body of the SUT’s operations (HTTP level, Browser level)

Observation Technique for XSS: crawl the source page to see if the injected vector has been sanitized.

Test terminology dedicated to Vulnerability Testing:

- Test verdict is OK → **Attack-pass**: System is vulnerable
- Test verdict is KO → **Attack-fail**: System is resistant
Early Results on WackoPicko

Experiment on Multi-Step XSS testing:

- 1 test purpose
- 2 abstract test cases:
  - login input
  - comment input
- 210 test executions:
  - 105 variants retrieved from OWASP’s XSS Cheat Sheet
- Results:
  - All failed for login input
  - 85 Successes / 20 fails for comment input

Concordant with our manual experiments.

⇒ 0 false positive, 0 false negative
Introduction SoTA MBVT Process Discussion Conclusion

Early Results on a real application

Application Under Test presentation:
  ▶ Virtual Learning Environment
  ▶ Highly used by french learning academies (> 90%)
  ▶ More than 15000 users

Experiment on Multi-Step XSS testing:
  ▶ 1 test purpose
  ▶ 11 abstract test cases
  ▶ 1155 test executions:
    ▶ 105 variants retrieved from OWASP’s XSS Cheat Sheet
    ▶ 16 steps per test case
    ▶ 8 steps between injection and observation

Concordant with our manual experiments.
⇒ 0 false positive, 0 false negative
Discussion

MBVT appears as an accurate and precise technique.

It also has its limitations, inherited from MBT:

- Needed effort to provide Models
- Needed effort to design Adaptation

Potential solutions:

- Use of a behavioral crawler to infer most parts of models
- Use of User traces to complement the results of the crawler
- Identify the reusability capacity of each artifact
MBVT does not suit every vulnerability type.

$\Rightarrow$ MBVT Scope based on OWASP TOP 10 2013:

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 - Injection</td>
<td>Done</td>
</tr>
<tr>
<td>A2 - Broken Authentication and Session Management</td>
<td>Doable</td>
</tr>
<tr>
<td>A3 - Cross-Site Scripting (XSS)</td>
<td>Under study</td>
</tr>
<tr>
<td>A4 - Insecure Direct Object References</td>
<td>Out of scope</td>
</tr>
<tr>
<td>A5 - Security Misconfiguration</td>
<td>Under study</td>
</tr>
<tr>
<td>A6 - Sensitive Data Exposure</td>
<td>Done</td>
</tr>
<tr>
<td>A7 - Missing Function Level Access Control</td>
<td>Doable</td>
</tr>
<tr>
<td>A8 - Cross-Site Request Forgery (CSRF)</td>
<td>Under study</td>
</tr>
<tr>
<td>A9 - Using Components with Known Vulnerabilities</td>
<td>Out of scope</td>
</tr>
<tr>
<td>A10 - Unvalidated Redirects and Forwards</td>
<td>Out of scope</td>
</tr>
</tbody>
</table>
Conclusion and Future Works

MBVT is a novel technique for Dynamic Application Security Testing.

Goal
To improve the precision and accuracy of vulnerability testing.

First Approach
- vTP into Test purposes
- Modeling of the SUT
- Test cases fed with a vectors battery

Limitations
Needed effort to provide models and design adaptation layer.

Work in Progress
Model inference, User traces, generic artefacts, test purposes extensions, real-life applications experiments.
Thank you for your attention

“Testing is always model-based!”
Robert Binder

Source - http://model-based-testing.info