Risk-Driven Vulnerability Testing: Results from eHealth Experiments using Patterns and Model-Based Approach

A. Vernotte, C. Botea, B. Legeard, A. Molnar, and F. Peureux
Outline

- Context and motivation
- Case Study: MediPedia The Medical Encyclopedia
- Pattern-driven and Model-based Vulnerability Testing
- Experimentation results
- Discussion and conclusion
Context and Motivation

- Continued growth and complexity of the internet:
  - increasing ubiquity of uses (eHealth, eBanking, eCommerce, social...)
  - increasing combination of technologies (server, client)

  Maintaining security is a challenging and critical task

- Focus put on Web applications vulnerabilities:
  - Cross-Site Scripting
  - SQL Injections
  - Cross-Site Request Forgery
  - ... (see OWASP Top 10$^1$ – CWE/SANS TOP 25$^2$)

Security testing: State of the practice

Static Techniques
- Code review
- Static Application Security Testing (SAST)

Manual Techniques
- Manual Penetration Testing

Automated Techniques
- Intrusive proxies (Burp suite, WebScarab, ...)
- Vulnerability scanners, Fuzzing tools, ...

Dynamic Techniques
- Dynamic Application Security Testing (DAST)

A. Vernotte, C. Botea, B. Legeard, A. Molnar, and F. Peureux
Risk-Driven Vulnerability Testing: Results from eHealth Experiments
Existing DAST Approaches

Manual Penetration Testing
- Performed by specialists (Pentesters, Security Testing Engineer)
- Use of multifunction toolboxes (Burp, Metasploit, Fuzzing, ...)
- Based on insights and experience
- Being systematic is challenging (compromise between available time / risk)

Automated Testing / Scanners
- Point-and-shoot solutions
- Efficient for a lot of vulnerability types
- Struggle when it comes to complex and logical vulnerability types
- Crawling often misses parts of an application:
  - false positives
  - false negatives
Objectives of the PMVT approach

- To provide a systematic guidance for DAST techniques from risk assessment

- To automate test case derivation and execution using model-based security testing techniques

- To support compositional analysis to manage large scale networked system in complex environments
Use Case: MediPedia

- Complex eHealth Romanian web portal developed by Info Worl (http://www.medipedia.ro)
- 125k+ weekly visits, 36k+ active accounts
- Stores highly sensitive personal data
- Selection of features:
  - Users can manage their electronic health records
  - Integrated with nation-wide Medcenter labs (analyses are automatically uploaded to user’s account)
  - Allows scheduling appointments
  - Forum for interacting with peers and medical practitioners
Use Case: MediPedia

RISK 2015
15.06.2015
Berlin

A. Vernotte, C. Botea, B. Legoard, A. Molnar, and F. Peureux

Risk-Driven Vulnerability Testing: Results from eHealth Experiments
Consultatii Specialitati
NA2041

<table>
<thead>
<tr>
<th>Pacient: COJOCARU AURELIAN</th>
<th>CNP: 165101510021</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data naşerii: 15 octombrie 1965 (49 ani)</td>
<td>Sex: Masculin</td>
</tr>
<tr>
<td>Medic: DAVID CRISTIAN</td>
<td>Data: 26.09.2007 08:30:33</td>
</tr>
</tbody>
</table>

Bilet de trimitere primit

<table>
<thead>
<tr>
<th>Detaliu vizită:</th>
<th>Data: 10.06.2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serie și număr:</td>
<td>Motivul trimiterii: -</td>
</tr>
<tr>
<td>PNS:</td>
<td></td>
</tr>
<tr>
<td>Unitate medicală:</td>
<td>Medic trimițător:</td>
</tr>
<tr>
<td>Denumire:</td>
<td>Nume:</td>
</tr>
<tr>
<td>CUI:</td>
<td>Cod parăță:</td>
</tr>
<tr>
<td>Sediu:</td>
<td>Număr contract:</td>
</tr>
<tr>
<td>Județul:</td>
<td>Casa de asigurări:</td>
</tr>
<tr>
<td></td>
<td>Specialitate:</td>
</tr>
</tbody>
</table>

Proceduri:

<table>
<thead>
<tr>
<th>Procedura efectuată</th>
<th>De la</th>
<th>Pana la</th>
</tr>
</thead>
<tbody>
<tr>
<td>Studiu electrofiziologic al inimii cu cel mult 3 catetere (38209-00)</td>
<td>26.09.2007 08:48:00</td>
<td>26.09.2007 10:48:41</td>
</tr>
</tbody>
</table>

Antecedente medicale:

<table>
<thead>
<tr>
<th>Diagnostic</th>
<th>Tip Diagnostic</th>
<th>Diagnostic asociat</th>
<th>Data diagnosticări</th>
<th>Data rezolvări</th>
<th>Anameza</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hipertensiunea esențială (primară)</td>
<td></td>
<td></td>
<td>01.04.1996</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HEMOLEUCOGRAAMA COMPLETA

<table>
<thead>
<tr>
<th>Denumire</th>
<th>7/9/2007 12:00:00 AM</th>
<th>Valori normale</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERITROCITE (10^12/L)</td>
<td>5,32</td>
<td>4,3-5,8</td>
</tr>
</tbody>
</table>
Pattern-driven and Model-based Vulnerability Testing Process

1. Risk Assessment, Risk model design
2. Test purpose definition
3. Behavioral model design
4. Test generation
5. Concretization, test execution and verdict assignment
1. Risk Assessment Inputs

1. **Risk Assessment, Risk model design**
2. Test purpose definition
3. Behavioral model design
4. Test generation
5. Concretization, test execution and verdict assignment

**RASSEN**

A. Vernotte, C. Botea, B. Legear, A. Molnar, and F. Peureux

Risk-Driven Vulnerability Testing: Results from eHealth Experiments
Risk Identification and Prioritization

- Using the CORAS approach to provide test objectives identification and prioritization based on the risk analysis:
  - Selection of the test purposes from identified threat scenarios
  - Prioritization of the test purposes regarding risk assessment
Based on a risk analysis of Medipedia using CORAS, we were able to orientate Vulnerability Testing towards five types of flaws:
- SQL Injection
- Cross-Site Scripting
- Cross-Site Request Forgery
- Auth Bypass & Privilege Escalation
- Insecure Direct Object Reference

The present experimental report and presentation focus on the first three mentioned vulnerability types: XSS, SQLi, CSRF.
Security Test Patterns

Security test patterns are typically related to vulnerability catalogues
- MITRE CWE & CAPEC
- OWASP Top 10

Solution
- one or more test design technique and corresponding strategies, test effort and effectiveness

Test Data
- instructions for crafting test data
- references to test data libraries or generators

Tools
- references tools that can be used to generate and execute such test cases

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>A meaningful name for the pattern, e.g. the name of the weakness.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE-ID(s)</td>
<td>The IDs of a weakness from the Common Weakness Enumeration.</td>
</tr>
<tr>
<td>Weakness Description</td>
<td>A high-level description of the weakness.</td>
</tr>
<tr>
<td>Solution</td>
<td>How the weakness could be revealed manually.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Design Technique</th>
<th>Test design technique that is able to find the weakness.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Strategies</td>
<td>Test strategies specific for a certain test design technique that shall be applied in order to generate test cases for the weakness in question.</td>
</tr>
<tr>
<td>Effort</td>
<td>The effort to generate and execute such test cases on a scale with the values 'low', 'medium', and 'high'-</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>How effective is the test design technique in finding such a weakness (how many test cases are necessary to find one weakness, how many weaknesses might be missed).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description of Test Coverage Items</th>
<th>Informal description of items to be covered by test cases created on basis of a pattern.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metrics</td>
<td>Appropriate test and coverage metrics. These will be developed in Task T4.3. This field is omitted within this deliverable.</td>
</tr>
</tbody>
</table>

Discussion
- A short discussion on the pitfalls of applying the pattern and the potential impact it has on test design in general and on other patterns applicable to that same context in particular.

<table>
<thead>
<tr>
<th>Test Data</th>
<th>Actual or references to test data and test data generation practices.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tools</td>
<td>References to tools appropriate for test case generation.</td>
</tr>
<tr>
<td>Generalization of</td>
<td>References to other security test patterns that are similar.</td>
</tr>
<tr>
<td>References</td>
<td>References to OWASP Top 10 weaknesses CWE descriptions and CAPEC attack patterns</td>
</tr>
</tbody>
</table>

Generic

A. Vernotte, C. Botea, B. Legear, A. Molnar, and F. Peureux
Risk-Driven Vulnerability Testing: Results from eHealth Experiments
2. Test Purpose Definition

1. Risk Assessment, Risk model design
2. **Test purpose definition**
3. Behavioral model design
4. Test generation
5. Concretization, test execution and verdict assignment

**R A S E N**

A. Vernotte, C. Botea, B. Legeard, A. Molnar, and F. Peureux

Risk-Driven Vulnerability Testing: Results from eHealth Experiments
Vulnerability Test Patterns (vTP) are the entry-point of PMVT.

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

**Smarttesting Test Purpose Language**:  
- Textual language based on regular expressions  
- Reasons in term of states to be reached and operations to be called  
- Drive the (security) test generation process

Three test purposes were used to generate test cases for Medipedia:

- Cross-Site Scripting in all its forms
- Standard SQL injection (first and second order)
- Cross-Site Request Forgeries

```groovy
for_each_instance $page from "Page.allInstances().--select(p:Page|not(p.all_outputs--isNotEmpty()))" on_instance sut,
for_each_instance $param from "self.all_outputs" on_instance $page,
use any_operation any_number_of_times to_reach
"WackPick.allInstances()--any(true).webAppStructure.ongoingAction.all_inputs--exists(d:Data|d=self)"
on_instance $param
then use threat.injectXSS($param)
then use any_operation any_number_of_times to_reach
then use threat.checkXSS()
```
2. Behavioral Model Design

1. Risk Assessment, Risk model design
2. Test purpose definition
3. Behavioral model design
4. Test generation
5. Concretization, test execution and verdict assignment
Behaviors modeling notation is based on UML metamodel:

- **Class diagrams** specify the static structure (points of control and observation)
- **Object diagrams** specify concrete business entities
- **State diagrams** graphically describe its behavioural characteristics
The Medipedia model contains:

- 25 pages
- 20 actions
- 28 navigation links
- 47 data

All three UML diagrams are generated from a dedicated DSML called DASTML.

Using DASTML, modeling the targeted Medipedia web pages took approximately 3 hours.

[...]

"PAT_VIEW_TOPIC" {
  ACTIONS {
    "PAT_POST_ANSWER" {
      "PAT_PA_CONTENT" = "PAT_PA_CONTENT_1"
      => {"ANON_VIEW_TOPIC","PAT_VIEW_TOPIC"}
    },
    "PAT_EDIT_POST" {
      "PAT_PE_CONTENT" = "PAT_PE_CONTENT_1"
      => {"ANON_VIEW_TOPIC","PAT_VIEW_TOPIC"}
    }
  }
  NAVIGATIONS {
    "GOTO_PAT_FORUMS"
    -> "PAT_FORUMS"
  }
}[...]
4. Security Test Generation

1. Risk Assessment, Risk model design
2. Test purpose definition
3. Behavioral model design
4. **Test generation**
5. Concretization, test execution and verdict assignment
Test generation strategies

Test cases are automatically generated using Smartesting Certifylt by composing behavioral models and test purposes:

- For one Test Purpose, several (or many) test cases by:
  - Applying usual Test Purpose coverage criteria
  - Applying behavioral fuzzing strategy (Fuzzino) given from Test Patterns

- Risk-based test selection criteria: the highest the priority is, the more combination of iterator values are generated

- Traceability management from security requirements to generated tests is build-in

**Result:** a suite of abstract vulnerability test cases
5. Test Concretization for Execution

1. Risk Assessment, Risk model design
2. Test purpose definition
3. Behavioral model design
4. Test generation
5. Concretization, test execution and verdict assignment

A. Vernotte, C. Botea, B. Legeard, A. Molnar, and F. Peureux

Risk-Driven Vulnerability Testing: Results from eHealth Experiments
JUnit test scripts are automatically generated by CertifityIt using an adaptation layer concretizing abstract data into concrete values:

- For one abstract test case, several (or many) executable test cases by:
  - Using a set of selected test data given from Test Patterns
  - Applying data fuzzing strategy given from Test Patterns

- Traceability management from security requirements to executable tests is built-in

**Result:** a set of executable vulnerability test scripts
Test results for MediPedia

Test generation took **25 minutes**
Test concretization took **6 hours** (including Selenium primitives customization + data mapping)
Test execution took about **50 minutes**

<table>
<thead>
<tr>
<th></th>
<th>SQLi</th>
<th>XSS</th>
<th>CSRF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstract Test Cases</strong></td>
<td>47</td>
<td>27</td>
<td>11</td>
</tr>
<tr>
<td><strong>Attack Vectors</strong></td>
<td>10</td>
<td>105</td>
<td>1</td>
</tr>
<tr>
<td><strong>Executable Test Cases</strong></td>
<td>470</td>
<td>2845</td>
<td>11</td>
</tr>
<tr>
<td><strong>Detected Vulnerabilities</strong></td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td><strong>Falses Positives</strong></td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td><strong>False Negatives</strong></td>
<td>0</td>
<td>24</td>
<td>0</td>
</tr>
</tbody>
</table>
This Risk-Based Vulnerability Testing technique appears to be **accurate** and **precise**.

Two previously unknown XSS vulnerabilities were found in the forum section of Medipedia.

It also has its limitations, inherited from MBT:
- Needed effort to provide Models
- Needed effort to provide Concrete Data

Test verdict assignment is not precise enough for CSRF

Two tests came back positive but we were not able to reproduce the attack.
Conclusion and future work

- PMVT approach combines RASEN partners risk assessment and testing techniques:
  - Risk identification and prioritization using CORAS method
  - Import of risk assessment results from CORAS tool into Certiflyt tool
  - Test purpose generation method (Certiflyt)
  - Behavioral and data fuzzing strategies (Fuzzino)
- Extension of security test patterns for risk-based vulnerability test case generation
- Generic formalization of the security test patterns into dedicated test purposes to drive the risk-based test generation
- Promising experimental results on real-life application (undiscovered XSS have been found)

Future investigations:
- Definition of more accurate testing strategies regarding risk prioritization
- Extension of security test patterns and related test purposes
- Enhancement of the MBT modeling activity (e.g., by recording and using execution traces)
- Improvement of the tool integration (especially Test Purpose / fuzzing)
- Improvement of the verdict assignment in order to avoid false positives (cf. CSRF detection)
Thank you for your attention!

Questions and Comments?

“Testing is always model-based!”
Robert Binder

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