A Pattern-driven and Model-Based Test Generation Toolchain for Web Vulnerability

Alexandre Vernotte¹, Bruno Legeard¹,², and Fabien Peureux¹,²

¹ Institut FEMTO-ST, UMR CNRS 6174 – Route de Gray, 25030 Besançon, France
{avernotte,blegeard,fpeureux}@femto-st.fr
² Smartesting R&D Center – 2G, Avenue des Montboucons, 25000 Besançon, France,
bruno.legeard@smartesting.com

Abstract. The purpose of this demonstration is to present a tooled Pattern-driven and Model-based Vulnerability Testing approach (PMVT for short) to improve the capability of detection of various vulnerability types such as injections (Cross-Site Scripting, SQL injections, etc.). This approach relies on generic vulnerability test patterns, which are applied on a behavioral model of the application under test, in order to generate vulnerability test cases. Hence, we propose to demonstrate the toolchain especially regarding XSS and SQLI vulnerabilities. This prototype has been experimented and validated on real-life Web applications, showing a strong improvement of detection ability w.r.t. Web application scanners for these vulnerabilities. This work is partially supported by the FP7 European project RASEN (http://www.rasenproject.eu).

1 Context

Based on the current state of the art on security and on all the security reports like OWASP Top Ten 2013 CWE/SANS 25 and WhiteHat Website Security Statistic Report 2013, Web applications are the most popular targets when speaking of cyber-attacks. The mosaic of technologies used in current Web applications increases the risk of security breaches. This situation has led to significant growth in application-level vulnerabilities.

Application-level vulnerability testing is first performed by developers, but they often lack the sufficient in-depth knowledge in recent vulnerabilities and related exploits. This kind of tests can also be achieved by companies specialized in security testing, in penetration testing for instance. But they mainly use manual approaches, making the dissemination of their techniques very difficult, and the impact of this knowledge very low. Finally, Web application vulnerability scanners can be used to automate the detection of vulnerabilities, but since they often generate many false positive and false negative results, human investigation is also required [2].

The approach we propose, called Pattern-driven and Model-based Vulnerability Testing (PMVT for short) aims to improve the accuracy and precision of vulnerability testing, by proposing an automated testing approach driven by vulnerability test patterns composed with a behavioral model of the system under test (SUT). The next sections respectively introduce the principles of the PMVT approach and describe the architecture of the toolchain implementing it.
2 Principles of the PMVT Approach

The PMVT process, shown in Fig. 1, is composed of the four following activities.

1. **The Test Purpose design activity.** It consists of formalizing a test procedure from vulnerability test patterns (vTP) that the generated test cases have to cover. Each procedure, targeting a dedicated vulnerability, is given by a test purposes, which is a high-level expression. It formalizes a testing objective and drives the automated test generation on the behavioral model of SUT.

2. **The Modeling activity.** As for every Model-Based Testing (MBT) approach, the modeling activity consists of designing a test model that will be used to automatically generate abstract test cases. To ease and accelerate this modeling activity, we have developed a Domain Specific Modeling Language (DSML), called DASTML, that allows to model the global structure of a Web application.

3. **The Test Generation activity.** It consists of automatically producing abstract test cases, including expected results, from the artifacts defined during the two previous activities.

4. **The Adaptation, Test Execution and Observation activity.** During the modeling activity, all data used by the application are modeled at an abstract level. As a consequence, the test cases are abstract and cannot thus be executed as they are. To bridge the gap, this activity mainly consists of linking the abstract data to concretes one in order to provide executable test scripts. It should be underlined that these test scripts embed the test sequence and the observation procedure in order to automate the verdict assignment.

Basically, the main contribution (that we aim to illustrate and emphasize during demonstration) of this tooled approach concerns [4]:

- The formalization of vulnerability test patterns using generic test purposes to drive the test generation engine.
- The use of a DSML to ease and accelerate the functional modeling of the Web application under test.
- The full automation of the testing process, including test generation, test execution and verdict assignment.
3 Overview of the PMVT toolchain

All the PMVT activities are supported by a dedicated toolchain, based on an existing Model-Based Testing (MBT) software named CertifyIt [3] provided by the company Smartesting3. CertifyIt is a test generator that takes as input a test model (written with a subset of UML called UML4MBT [1]), capturing the behavior of the SUT. This model is managed using a dedicated plugin in IBM Rational Software Architect4.

The process supported by the toolchain takes as input one or several vulnerability test purpose (vTP) and a UML4MBT model of the SUT. A test purpose formalizes a given vulnerability in relation with associated generic vulnerability test pattern catalogues. The demonstration will address Cross-Site Scripting (XSS) and SQL injections (SQLI), but the security test pattern catalogue is currently being extended to address the OWASP Top 10 weaknesses. Basically, such a test purpose is a sequence of significant steps that has to be exercised by the test case scenario in order to assess the robustness of the application under test w.r.t. the related vulnerability. Each step takes the form of a set of operations to execute, or specific state to be reached on the UML4MBT behavioral model. For instance, Fig. 2 shows the test purpose formalizing the vTP for SQLI.

![Fig. 2. SQLI Vulnerability Test Purpose](image)

During test generation, this test purpose drives the vulnerability test generation on the UML4MBT behavioral model. This model is designed using a dedicated DSML (called DASTML) that allows to describe all the structural entities and behaviors, of the Web application under test, which are needed to generate vulnerability test cases: pages, available actions on each page, etc.

At this stage, the test generator automatically combines model and test purposes to compute abstract test cases, as shown in Fig. 3(a), regarding SQLI. To automate test execution, the test engineer has to match each abstract value with a concrete value. This consists of automatically creating a JUnit test suite, in which each abstract test case is exported as a JUnit test case. The test case generator also produces a file containing the signature of each operation that the test engineer has to implement using a Web navigation library such as HtmlUnit or Selenium. Once this task is completed, test cases can be executed and verdict is assigned automatically, as shown in Fig 3(b).

![Fig. 3. Test Case Generation and Execution using PMVT Approach](image)

In order to illustrate the PMVT approach during Demonstration Session, we will use the toy example WackoPicko. This deliberately-unsecured Web application allows users to authenticate themselves, share comment and buy pictures. To keep it simple, we will focus on Cross-Site Scripting vulnerability: a malicious user injects a script and forces victims visiting the page to execute them.

References


[https://github.com/adamdoupe/WackoPicko](https://github.com/adamdoupe/WackoPicko) [October 2014]