Security Testing with Selenium

Vidar Kongsli

Bekk Consulting AS
Vippetangen, Skur 39
N-0150 Oslo, Norway
+47 982 19 329
vidar.kongsli@bekk.no

Abstract

Selenium is a tool for creating and running automated web tests and is a good fit for agile projects where it can be used for creating acceptance tests corresponding to the web application's user stories.

This demonstration will show how Selenium additionally can be leveraged to create security tests. First, we model security threats as *misuse stories*, similar to user stories except that we focus on illegal or non-normative use of the application. Subsequently, we create security tests in Selenium that manifest the misuse stories by exploiting vulnerabilities in the application.

This approach can be seen as a contribution to strengthening the security focus in agile projects by trying to apply familiar agile concepts, methods, and tools to the security aspects of the application.

We have found that several of the most common security vulnerabilities in web applications can be addressed with this approach, such as cross site scripting (XSS), broken authentication and access management, information leakage, and improper error handling. This demonstration will show examples of such vulnerabilities and corresponding tests, in addition to discussing the cases where there are shortcomings.

Categories and Subject Descriptors D.2.5 [**Software Engineering**]: Testing and debugging – *testing tools*; K.6.3 [**Software Engineering**]: Software management – *software process*; K.6.5 [**Management of Computing and Information Systems**]: Security and protection

General Terms Security

Keywords Agile; security; testing; misuse stories

OOPSLA'07 October 21–25, 2007, Montréal, Québec, Canada.
ACM 978-1-59593-865-7/07/0010.

1. Introduction

Security is an increasingly important concern in web applications, as web applications are becoming more complex and business critical. Equally important, web based attacks have become more sophisticated.

Agile and iterative methodologies focusing on refactoring, team co-location, and automated testing, seem to run counter to the view of traditional security methodologies [1]. For instance, security methodology is often sequential rather than iterative; it often requires "big design up front" in order to assess the security of the system.

In this demonstration, we will show how agile methods and tools can be leveraged to improve the security of a web application during development. Specifically, misuse stories will be used to model security threats and automated security tests will be created and run using Selenium, a popular open source web testing tool.

2. Misuse Stories

A user story is a software requirement formulated in up to three sentences in everyday language. The user stories drive the creation of the acceptance tests [2]. While a user story has a low level of detail and is just accurate enough to create a proper estimate, an acceptance test specifies how the system shall behave very precisely.

In a similar manner, we utilize *misuse stories* to describe illegal or non-normative use of the system. (For projects utilizing use cases, misuse cases or 'abuse cases' [3], have been suggested for modeling misuse ([4], [5], [6] and others.)

When a user story is selected for implementation, we derive possible misuse stories associated with that user story. The main question during this session is '*how can the functionality described by this user story be misused?*'

3. Enter Selenium

After identifying a misuse story, the next step is to create an acceptance test that manifests the story using the web testing tool Selenium (http://www.openqa.org/selenium/).

The rationale for using Selenium for security testing is to leverage a tool that is already in use by the development team, avoid learning and maintaining a separate tool for security. Furthermore, a familiar tool could help to narrow the gap between 'a typical' developer and a security specialist. Having security requirements written as web tests also supports and enhances collective ownership.

4. Creating Security Tests

In order for a misuse story to be executed, there has to be vulnerabilities in the application that can be exploited. Furthermore, to be able to successfully create a Selenium test that executes a misuse story, there has to be a vulnerability exposed through the web interface of the application.

The OWASP Top Ten [7] is used for identifying vulnerabilities for this demonstration.

4.1 Testing for insecure input handling

One of the most important classes of vulnerabilities is insecure handling of input to the application. Basically, all input to the application have to be handled appropriately using one of, or a combination of the following:

- Input validation to ensure that the input is within a valid range, has an appropriate format, and adheres to business rules
- Meta character escaping to avoid command injections

We will demonstrate how Selenium tests can be constructed to verify appropriate handling of inputs.

4.2 Cross Site Scripting (XSS)

Cross site scripting attacks typically exploit vulnerabilities that allow a malicious user to insert JavaScript into the web application user interface. We will show how a Selenium test for such vulnerability can be created leveraging the JavaScript event model.

4.3 Cross Site Request Forgery (CSRF)

A CSRF attack forces a logged-in victim's browser to send a request to a vulnerable web application. Very often, an XSS vulnerability in the web application is used as one of the steps taken to perform such attacks.

We will demonstrate how a test for CSRF vulnerabilities can be constructed, and how these vulnerabilities can be removed.

4.4 Information Leakage/Improper Error Handling

Error conditions can be used by a malicious user to gain information about the system. This typically occurs when the application exposes information about an error that is helpful for developers or system administrators. Too often, such information used for development has not been removed in the production environment.

We will show a strategy where we try to provoke an error situation, and check what type of output is generated.

4.5 Broken Authentication and Access Management

We will demonstrate how Selenium tests can check for vulnerabilities such as rights elevation and session fixation.

4.6 Insecure Communications

Tests that verify that SSL is forced on pages that convey sensitive information or credentials will be demonstrated.

5. Shortcomings

We have found that in many cases vulnerabilities exposed through the user interface can be exploited to carry out an attack, which makes it possible to create a test for those vulnerabilities using Selenium.

However, there are vulnerabilities that are very hard to expose through the web interface, for instance vulnerabilities in the "insecure cryptographic storage" category, as noted in [7].

Furthermore, Selenium core being a JavaScript based framework, it is subject to the normal restrictions on JavaScript when running in a browser.

References

[1] Kongsli, V. *Towards Agile Security in Web Applications*, OOPSLA 2006.

[2] Extreme Programming: a gentle introduction, http://www.extremeprogramming.org.

[3] McGraw, G. Software Security - Building Security In, Addison-Wesley, 2006.

[4] Sindre, G., Opdahl, A. *Templates for Misuse Case Description*, Seventh International Workshop on Requirements Engineering, Interlaken, Switzerland, 2001.

[5] Alexander, I., *Initial Industrial Experience of Misuse Cases in Trade-Off Analysis*, Proceedings of IEEE Joint International Requirements Engineering Conference, 9-13 September 2002, Essen, pp 61-68.

[6] Firesmith, D. OPEN Process Framework, http://donald-firesmith.com/

[7] OWASP: Ten Most Critical Web Application Security Vulnerabilities 2007 update, http://www.owasp.org/.