Vulnerability Assessment Tool

AMOS project proposal

Hans Malte Kern & Dr. Christian Höppler*
Robert Bosch GmbH

January 25, 2013

This paper is a proposal to develop a Vulnerability Assessment Tool (VAT) to query security vulnerability databases. It should be developed as a library with a simple web interface to interact with the functionality. If time permits, the project also embarks on a maven integration.

1 Motivation

Both in commercial and free open source software development the developers usually know which open source components are integrated in their products. However, one aspect that is often neglected is watching for known security vulnerabilities of the used components and upgrading to have serious vulnerabilities fixed [1]. Of course, a manual search in major security vulnerability databases [2–4] can be a major effort. That is why we are looking for tool and automation support.

We are aware of two commercial tools—*Black Duck™ Code Center* and *Sonatype CLM for CI*—that also provide some of the functionality of VAT, but no open source solution exists as of now.

2 Main use cases

Web: Upload list of OSS components and download report. A user uploads a list of all OSS components she is interested in to the web interface of VAT. VAT provides the download of an output file with a list of known security vulnerabilities for each component.

Web: Display report for a specific OSS component. A user fills a web form with an OSS component's name and version. VAT then queries all configured databases and displays a web page with the list of security vulnerabilities.

*Authors' email addresses: hansmalte.kern@de.bosch.com and christian.hoeppler@de.bosch.com.

Web: Provide REST API for automated access to VAT functionality. The service can also be accessed automatically via its REST API.

Maven: Generate a security vulnerability report during build. When a java project is built with maven, the build system knows dependent software components. This data is used by the VAT-maven-plugin to report security vulnerabilities during the software build. The search for known security vulnerabilities is just another kind of static analysis. With this plugin the problem of unknowingly integrating software components with known security vulnerabilities [1, 7] could be considerably alleviated.

3 Proposed Architecture

To allow VAT to evolve and support future use cases we envision an architecture where the query library can be extended with different database connectors and input/output sources and formats as shown in figure 1.

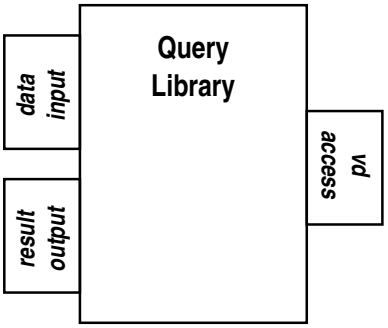

Figure 1: VAT library with interfaces to data input and output as well as to access vulnerability databases.

In a first version we would be interested in an implementation of the data input interface for XML and XLS. As for the result output, relevant formats would also be XLS and XML. For the latter we would like to see an XLST style sheet to be able to easily display the data as HTML.

Of the three databases, *The Open Source Vulnerability Database*, the *US National Vulnerability Database*, and the *Russian National Vulnerability Database*, we propose to start with the first one if possible, as it seems to have an accessible REST-API.

References

[1] Julia Schmidt. "Spring Framework: Alter Bug, neue Möglichkeiten." In: *heise Developer* (Jan. 18, 2013). URL: http://heise.de/-1787002 (visited on 01/22/2013).

[2] *The Open Source Vulnerability Database*. URL: http://osvdb.org/ (visited on 01/23/2013).

[3] *US National Vulnerability Database*. URL: http://nvd.nist.gov/ (visited on 01/22/2013).

[4] *Russian National Vulnerability Database*. URL: http://en.securitylab.ru/nvd/ (visited on 01/22/2013).

[5] *Black Duck™ Code Center*. URL: http://www.cm-logic.com/code-center.html (visited on 01/23/2013).

[6] *Sonatype CLM for CI*. URL: http://www.sonatype.com/Products/CLM-for-CI (visited on 01/23/2013).

[7] D. J. Walker-Morgan. "Repositories offer up vulnerable libraries says report." In: *The H open* (Mar. 30, 2012). URL: http://h-online.com/-1498138 (visited on 01/22/2013).