Semesterarbeit: SQL Injection Attacks and Defenses

Keywords

DHTML, HSQL Database Engine, HTML, input validation, J2EE, Java, Java Enterprise Platform, JavaScript, Java Servlet, JDBC, JSP, regular expression, SQL, SQL injection, SSL, TLS, Tomcat

Introduction & Goal

SQL (standard query language) injections are a form of attack against Web-based applications. These include allowing the attacker to log into otherwise restricted Web sites by "successful authentication", disclosing private information stored in a database, and modifying one or more database tables.

The purpose of this Semesterarbeit is to explain in detail what SQL injections are, what kinds of attack can be mounted with them, and how to defend against them by performing input validation. Results are to be delivered in written form, in form of a talk, and potentially in form of a Java class library.

An optional ("if time left") task is to explain means of defense other than input validation. (E.g., how can it be avoided that the attacker learns the relations in the database [assuming she/he does not know them already] by sanitizing error messages? Or what is the gain of using prepared statements instead of plain statements?)

Environment

Web-based applications typically have a three-tier architecture with a presentation tier, a processing tier, and a persistence tier:

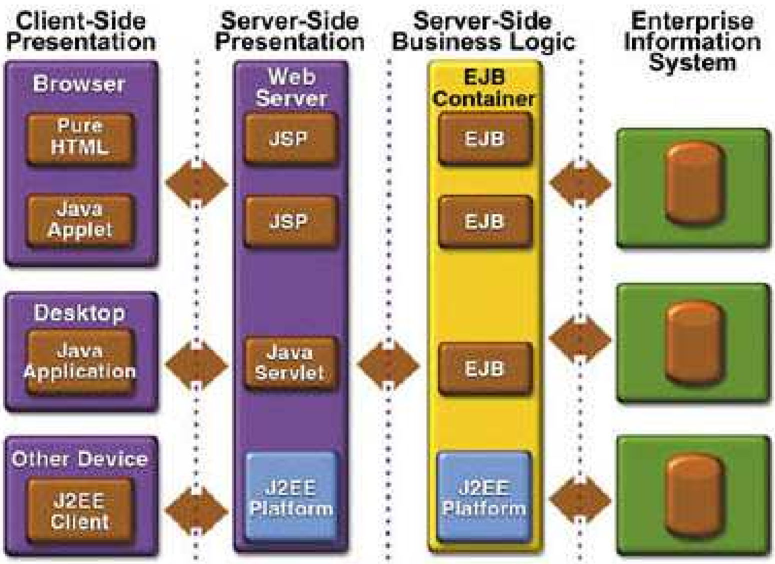

Illustration 1: J2EE Application Model

- The presentation tier is the interface to the user. It consists of (D)HTML pages the user accesses through her/his Web browser.
- The processing tier consists of a Web server and an application server. (Both roles can be played by one process.)
- The persistence tier consists of one or more enterprise information systems (EISs).

An enterprise resources planning (ERP) system such as SAP is an example for an EIS. Another example is a system providing naming and directory services.

The Web server generates and delivers the Web pages. The application server contains the business logic. The Web server generates and delivers the next page based on the (HTTP) request it receives from the presentation tier and through interaction with the application server and the persistence tier.

Hereafter, we confine ourselves to a simple Web-based application with only a Web server and no application server in its processing tier and only one kind of EIS in its persistence tier, namely the most common case of a relational database management system (DBMS). The Web server interacts with the DBMS by submitting SQL requests.

More specifically, we use Tomcat [3] as the Web server and the HSQL Database Engine [2] as the DBMS. Java ServerPages (JSPs) and Java Servlets accept HTTP requests and generate and deliver Web pages consisting of HTML and JavaScript. They submit SQL requests through JDBC (Java DataBase Connectivity).

Benefits for the Student

The student learns how to set up a development and test environment for Web-based applications, including generating a server certificate and enabling SSL/TLS. She/He gains experience in using popular open-source solutions such as the HSQL Database Engine and Tomcat as well as with the Java Enterprise Platform. In particular, she/he gets intimately familiar with SQL injection attacks and learns how to defend against them by performing regular expression-based input validation.

Prerequisites

- Java development experience
- familiarity with a relational DBMS and SQL

Potential Benefits for the Student and the InfSec Group

Use the results of the Semesterarbeit in the context of the applied security laboratory.

Publish an article based on the Semesterarbeit in a magazine such as IEEE's brand-new *Security & Privacy*.

Make the input-validation classes available to third parties.

Supervisors

Prof. Dr. David Basin and Paul E. Sevinç, Chair of Information Security. Please contact Paul (paul.sevinc@inf.ethz.ch) if you are interested in this Semesterarbeit.

Possibility for Subsequent Diplomarbeit

This Semesterarbeit allows the student to get started with J2EE security. And if she/he is interested in further working on this topic, we can leverage this knowledge in a Diplomarbeit. For instance, the J2EE standard merely defines authentication and authorization mechansims, whereas defining the confidentiality mechanism is left to the application server providers, which might lead to vendor lock-in. Therefore, proposing and realizing a confidentiality mechanism that works with any J2EE-compliant application server could be the topic of a Diplomarbeit.

References

1. Chapters 10-14 in M. Howard and D. LeBlanc. *Writing Secure Code.* Microsoft Press, Redmond, 2nd edition 2003.
2. The HSQL Database Engine available at http://hsqldb.sourceforge.net/.
3. The Tomcat servlet container available at http://jakarta.apache.org/tomcat/.