SQL-Injection

Eine SQL-Injection kann dadurch gelingen, dass in einem nicht ausreichend geschützten System in einer Eingabeaufforderung SQL-Code eingegeben und ausgeführt werden kann. Ein Beispiel für solchen Code wäre, wenn die in einer Eingabemaske eingetragenen Werte in einer Variable gespeichert würden und diese Variable einfach in eine vorgefertigte SQL-Anweisung nach folgender Art eingefügt wird:

String input = JOptionPane.showInputDialog(„Insert Titel“);

…

*rs = stmt.executeQuery(„select * from Books where title = `“ + input + “`“);*

Bei dem hier gezeigten Beispiel kann der Nutzer des Programms zwar in der Eingabemaske den Titel eines Buches eingeben und somit danach suchen, er hat jedoch auch die Möglichkeit, noch zusätzliche SQL-Anweisungen einzugeben.
Eine solche SQL-Injection könnte hier folgendermaßen aussehen:

titelDesBuchs`; DELETE FROM Books WHERE title = `titelEinesBuches

In diesem Fall wird zwar eine Variable “titelDesBuchs“ in die SQL-Anweisung eingefügt, aber zusätzlich auch noch die SQL-Anweisung “DELETE FROM Books WHERE title = `titelEinesBuches“, die von der Inputvariable hier einfach durch ein “;“ getrennt wird. Neben der Möglichkeit, SQL-Anweisungen durch ein Semikolon zu trennen, können jedoch zusätzliche Anweisungen z.B. auch durch AND oder OR eingebunden werden. Von dem Programm würde in dem hier gezeigten Beispiel die folgende SQL-Anweisung ausgeführt:

*rs = stmt.executeQuery(„select * from Books where title = `titelDesBuchs`; DELETE FROM Books WHERE title = `titelEinesBuchs`“);*

Auf diese Art kann man beliebig viele SQL-Anweisungen in die Eingabeaufforderung und somit in die SQL-Anweisung “rs = stmt.executeQuery(„select * from Books where title = `“ + input + “`“); “ einfügen, da das Programm keinerlei Überprüfung oder Beschränkung der Eingabe vornimmt.
In dem hier gezeigten Beispiel würde es einem Angreifer somit gelingen, Einträge in der Tabelle Books zu löschen.

Um eine SQL-Injection zu verhindern, sollte man ein PreparedStatement in Verbindung mit einem “?“ als Platzhalter für eine Variable verwenden.
Der entsprechende Java-Code würde dann folgendermaßen aussehen:

String input = JOptionPane.showInputDialog(„Insert Titel);

…

*PreparedStatement p = con.prepareStatement("select * from Books where title = ?");*
p.setString(1, input);

Der Vorteil des PreparedStatement in Verbindung mit dem “?“ besteht darin, dass die Eingabe automatisch in doppelte Anführungszeichen eingeschlossen wird und somit maskiert ist. Eine SQL-Injection wird somit erschwert.
Neben dieser Technik gibt es jedoch noch weitere Maßnahmen, die unternommen werden können, um eine noch höhere Sicherheit zu gewährleisten. So kann man den Java-Code noch dahingehend optimieren, dass er nur Eingaben bis zu einer gewissen Länge zulässt. Auf diese Art, wird die Wahrscheinlichkeit verringert, dass ein Angreifer SQL-Anweisungen, die manchmal relativ lang sein können, eingeben kann. Eine Weitere Technik, die man außerdem noch einsetzen kann, ist Sonderzeichen, wie “;“ oder “=“ in der Eingabe zu verbieten. Diese Technik kann jedoch z.B. bei Passwörtern auf Schwierigkeiten stoßen, da solche Sonderzeichen Passwörter sicherer machen.