Cheatsheet: 8 best practices to prevent SQL injection attacks

1. Do not rely on client-side input validation
   - Client-side validation can be bypassed by executing raw HTTP calls using curl or tools like postman.
   - Always perform server-side validation.

2. Restrict database users
   - Create specific database users for your application with limited privileges.
   - Application users don't need to DROP or TRUNCATE tables generally.

3. Prepared statements and query parameterization
   - Don't concatenate user input with the query literal.
   - Use real prepared statements if possible.
   - Add untrusted input as parameters to the query.

4. Scan your code for SQLi
   - Use a SAST tool like Snyk Code to detect SQL injection in your custom code.

5. ORM layer
   - Use an ORM layer to map database results to objects. This prevents a lot of explicit SQL queries.
   - Be aware of custom queries also in specific dialects like HQL.
   - Scan used ORM libraries with Snyk Open Source for hidden SQL injection vulnerabilities.

6. Prevent blocklisting
   - Don't rely on blocklisting user input to prevent SQL injection.
   - Maintaining a blocklist is challenging, and takes a lot of effort. Some keywords or characters can also be legitimate names.

7. Input validation
   - Validation input is in general a good practice to lower security risk.
   - Might be a good alternative when prepared statements are not an option.
   - Good practice in a multi-layer defense strategy.

8. Watch out with stored procedures
   - Stored database procedures are not by default safe.
   - Be aware that stored procedures can also be vulnerable to SQL injection when implemented wrongly.
   - Check the documentation if you need to resort to this method.

www.snyk.io

Brian Vermeer
@BrianVerm
Developer Advocate at Snyk