JavaScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVASCRIPT code

Database queries should not be vulnerable to injection attacks

Analyze your code

Vulnerability Blocker

injection cwe owasp sans-top25 sql

User-provided data, such as URL parameters, should always be considered untrusted and tainted. Constructing SQL queries directly from tainted data enables attackers to inject specially crafted values that change the initial meaning of the query itself. Successful database query injection attacks can read, modify, or delete sensitive information from the database and sometimes even shut it down or execute arbitrary operating system commands.

Typically, the solution is to use prepared statements and to bind variables to SQL query parameters with dedicated methods like `setParameter`, which ensures that user-provided data will be properly escaped. Another solution is to validate every parameter used to build the query. This can be achieved by transforming string values to primitive types or by validating them against a white list of accepted values.

Noncompliant Code Example

```
var db = require('./mysql/dbConnection.js');

function (req, res) {
  var name = req.query.name; // user-controlled input
  var password = crypto.createHash('sha256').update(req.quer

  var sql = "select * from user where name = '" + name + "'

  db.query(sql, function(err, result) { // Noncompliant
     // something
  })
}
```

Compliant Solution

```
var db = require('./mysql/dbConnection.js');

function (req, res) {
  var name = req.query.name; // user-controlled input
  var password = crypto.createHash('sha256').update(req.quer

  var sql = "select * from user where name = ? and password

  db.query(sql, [name, password], function(err, result) { //
     // something
  })
}
```

See

- OWASP Top 10 2021 Category A3 - Injection scheme

Vulnerability

Server hostnames should be verified during SSL/TLS connections

Vulnerability

Server certificates should be verified during SSL/TLS connections

Vulnerability

Cryptographic keys should be robust

Vulnerability

Weak SSL/TLS protocols should not be used

- OWASP Top 10 2017 Category A1 - Injection
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-89 - Improper Neutralization of Special Elements used in an SQL Command
- MITRE, CWE-943 - Improper Neutralization of Special Elements in Data Query Logic
- OWASP SQL Injection Prevention Cheat Sheet
- SANS Top 25 - Insecure Interaction Between Components

Available In:

sonarcloud | sonarqube Developer Edition