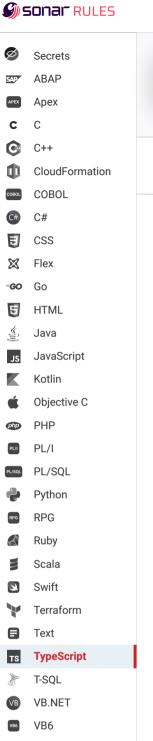
TypeScript static code analysis



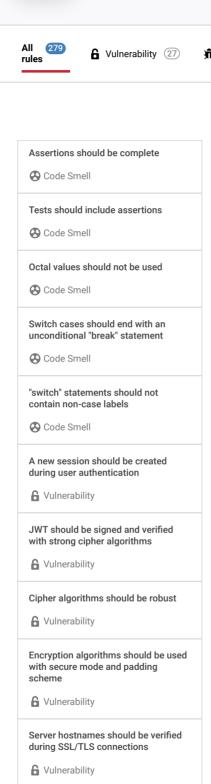
Products >

Analyze your code

injection cwe owasp sans-top25



XML



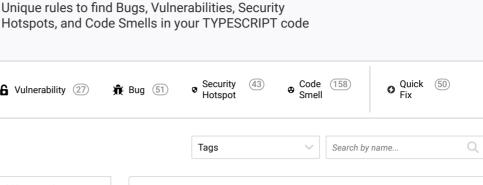
Server certificates should be verified

Cryptographic keys should be robust

during SSL/TLS connections

6 Vulnerability

Vulnerability



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 guery. This can be achieved by transforming string values to primitive types or by validating them against a white list of accepted

Noncompliant Code Example

Database queries should not be

vulnerable to injection attacks

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

Weak SSL/TLS protocols should not be used

Vulnerability

Origins should be verified during cross-origin communications

Vulnerability

Regular expressions should not be vulnerable to Denial of Service attacks

Vulnerability

File uploads should be restricted

Vulnerability

Regular expressions should be

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

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Privacy Policy