




 Secrets


 ABAP


 Apex


 C


 C++


 CloudFormation


 COBOL


 C#


 CSS


 Flex


 Go


 HTML


 Java


 **JavaScript**


 Kotlin


 Objective C


 PHP


 PL/I


 PL/SQL


 Python


 RPG


 Ruby


 Scala


 Swift


 Terraform


 Text


 TypeScript

 T-SQL

 VB.NET

 VB6


 XML





## JavaScript static code analysis


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


All rules285

 Vulnerability29

 Bug62

 Security Hotspot43


 Code Smell151

 Quick Fix41


Tags ▾

Search by name... 🔍


Server certificates should be verified during SSL/TLS connections

 Vulnerability


Cryptographic keys should be robust

 Vulnerability


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


Function calls should not pass extra arguments

 Bug


Regular expressions should be syntactically valid

 Bug


Getters and setters should access the expected fields

 Bug

"super()" should be invoked appropriately

 Bug


"Symbol" should not be used as a constructor


 Bug


Results of "in" and "instanceof" should be negated rather than operands


Hard-coded credentials are security-sensitive


Analyze your code

 Security Hotspot

 Blocker

 cwe

 sans-top25

 owasp

Because it is easy to extract strings from an application source code or binary, credentials should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.

In the past, it has led to the following vulnerabilities:

- [CVE-2019-13466](#)
- [CVE-2018-15389](#)

Credentials should be stored outside of the code in a configuration file, a database, or a management service for secrets.

This rule flags instances of hard-coded credentials used in database and LDAP connections. It looks for hard-coded credentials in connection strings, and for variable names that match any of the patterns from the provided list.

It's recommended to customize the configuration of this rule with additional credential words such as "oauthToken", "secret", ...

Ask Yourself Whether

- Credentials allow access to a sensitive component like a database, a file storage, an API or a service.
- Credentials are used in production environments.
- Application re-distribution is required before updating the credentials.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Store the credentials in a configuration file that is not pushed to the code repository.
- Store the credentials in a database.
- Use your cloud provider's service for managing secrets.
- If a password has been disclosed through the source code: change it.

Sensitive Code Example

```
var mysql = require('mysql');

var connection = mysql.createConnection(
{
  host:'localhost',
  user: "admin",
  database: "project",
  password: "mypassword", // sensitive
  multipleStatements: true
});

connection.connect();
```

Compliant Solution

https://rules.sonarsource.com/javascript/RSPEC-2068

1/2

 Bug
"in" should not be used with primitive types
 Bug
A compare function should be provided when using "Array.prototype.sort()"
 Bug
Jump statements should not occur in "finally" blocks
 Bug
Using slow regular expressions is security-sensitive

```
var mysql = require('mysql');

var connection = mysql.createConnection({
  host: process.env.MYSQL_URL,
  user: process.env.MYSQL_USERNAME,
  password: process.env.MYSQL_PASSWORD,
  database: process.env.MYSQL_DATABASE
});

connection.connect();
```

See

- [OWASP Top 10 2021 Category A7](#) - Identification and Authentication Failures
- [OWASP Top 10 2017 Category A2](#) - Broken Authentication
- [MITRE, CWE-798](#) - Use of Hard-coded Credentials
- [MITRE, CWE-259](#) - Use of Hard-coded Password
- [SANS Top 25](#) - Porous Defenses
- Derived from FindSecBugs rule [Hard Coded Password](#)

Available In:  
 