sonar RULES Products

Java static code analysis

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

All rules 632 | Vulnerability 53 | Bug 154 | Security Hotspot 36 | Code Smell 389 | Quick Fix 42

Tags | Search by name...

Hard-coded credentials are security-sensitive

Analyze your code

Security Hotspot | Blocker | cwe cert 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 allows 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

```
Connection conn = null;
try {
  conn = DriverManager.getConnection("jdbc:mysql://localhost
        "user=steve&password=blue"); // Sensitive
  String uname = "steve";
  String password = "blue";
  conn = DriverManager.getConnection("jdbc:mysql://localhost
        "user=" + uname + "&password=" + password); // Sensi

  java.net.PasswordAuthentication pa = new java.net.Password
```

Compliant Solution

Hashes should include an unpredictable salt

Vulnerability

Calls to methods should not trigger an IllegalArgumentException

Bug

Unsupported methods should not be called on some collection implementations

Bug

Cast operations should not trigger a ClassCastException

```
Connection conn = null;
try {
  String uname = getEncryptedUser();
  String password = getEncryptedPass();
  conn = DriverManager.getConnection("jdbc:mysql://localhost
        "user=" + uname + "&password=" + password);
```

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
- CERT, MSC03-J. - Never hard code sensitive information
- SANS Top 25 - Porous Defenses
- Derived from FindSecBugs rule Hard Coded Password

Available In:

sonarcloud sonarqube