sonar RULES

Products

- 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

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...

Vulnerability

Server hostnames should be verified during SSL/TLS connections
Vulnerability

Insecure temporary file creation methods should not be used
Vulnerability

Passwords should not be stored in plain-text or with a fast hashing algorithm
Vulnerability

Server certificates should be verified during SSL/TLS connections
Vulnerability

Persistent entities should not be used as arguments of "@RequestMapping" methods
Vulnerability

"HttpSecurity" URL patterns should be correctly ordered
Vulnerability

LDAP connections should be authenticated
Vulnerability

Cryptographic keys should be robust
Vulnerability

Weak SSL/TLS protocols should not be used
Vulnerability

"SecureRandom" seeds should not be predictable
Vulnerability

Cipher Block Chaining IVs should be unpredictable
Vulnerability

Resources should be closed

Analyze your code

Bug Blocker cwe leak denial-of-service cert

Connections, streams, files, and other classes that implement the `Closeable` interface or its super-interface, `AutoCloseable`, needs to be closed after use. Further, that `close` call must be made in a `finally` block otherwise an exception could keep the call from being made. Preferably, when class implements `AutoCloseable`, resource should be created using "try-with-resources" pattern and will be closed automatically.

Failure to properly close resources will result in a resource leak which could bring first the application and then perhaps the box the application is on to their knees.

Noncompliant Code Example

```
private void readTheFile() throws IOException {
  Path path = Paths.get(this.fileName);
  BufferedReader reader = Files.newBufferedReader(path, this
  // ...
  reader.close();  // Noncompliant
  // ...
  Files.lines("input.txt").forEach(System.out::println); //
}

private void doSomething() {
  OutputStream stream = null;
  try {
    for (String property : propertyList) {
      stream = new FileOutputStream("myfile.txt");  // Nonco
      // ...
    }
  } catch (Exception e) {
    // ...
  } finally {
    stream.close();  // Multiple streams were opened. Only t
  }
}
```

Compliant Solution

```
private void readTheFile(String fileName) throws IOException
    Path path = Paths.get(fileName);
    try (BufferedReader reader = Files.newBufferedReader(pat
      reader.readLine();
      // ...
    }
    // ..
    try (Stream<String> input = Files.lines("input.txt"))  {
      input.forEach(System.out::println);
    }
}
```

Basic authentication should not be used

Vulnerability

Regular expressions should not be vulnerable to Denial of Service attacks

Vulnerability

"HttpServletRequest.getRequestedSessio
should not be used

Vulnerability

Hashes should include an unpredictable salt

Vulnerability

```
private void doSomething() {
  OutputStream stream = null;
  try {
    stream = new FileOutputStream("myfile.txt");
    for (String property : propertyList) {
      // ...
    }
  } catch (Exception e) {
    // ...
  } finally {
    stream.close();
  }
}
```

Exceptions

Instances of the following classes are ignored by this rule because `close` has no effect:

- `java.io.ByteArrayOutputStream`
- `java.io.ByteArrayInputStream`
- `java.io.CharArrayReader`
- `java.io.CharArrayWriter`
- `java.io.StringReader`
- `java.io.StringWriter`

Java 7 introduced the try-with-resources statement, which implicitly closes `Closeables`. All resources opened in a try-with-resources statement are ignored by this rule.

```
try (BufferedReader br = new BufferedReader(new FileReader(f
  //...
}
catch ( ... ) {
  //...
}
```

See

- MITRE, CWE-459 - Incomplete Cleanup
- MITRE, CWE-772 - Missing Release of Resource after Effective Lifetime
- CERT, FIO04-J. - Release resources when they are no longer needed
- CERT, FIO42-C. - Close files when they are no longer needed
- Try With Resources

Available In:

sonarlint sonarcloud sonarqube