Java static code analysis

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

Abstract class names should comply with a naming convention
Code Smell

Strings literals should be placed on the left side when checking for equality
Code Smell

Files should contain an empty newline at the end
Code Smell

Source code should be indented consistently
Code Smell

A close curly brace should be located at the beginning of a line
Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines
Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line
Code Smell

An open curly brace should be located at the beginning of a line
Code Smell

An open curly brace should be located at the end of a line
Code Smell

Tabulation characters should not be used
Code Smell

Functions should not be defined with a variable number of arguments
Code Smell

Using hardcoded IP addresses is security-sensitive

Analyze your code

Security Hotspot   Minor   cert owasp

Hardcoding IP addresses is security-sensitive. It has led in the past to the following vulnerabilities:

- CVE-2006-5901
- CVE-2005-3725

Today's services have an ever-changing architecture due to their scaling and redundancy needs. It is a mistake to think that a service will always have the same IP address. When it does change, the hardcoded IP will have to be modified too. This will have an impact on the product development, delivery, and deployment:

- The developers will have to do a rapid fix every time this happens, instead of having an operation team change a configuration file.
- It misleads to use the same address in every environment (dev, sys, qa, prod).

Last but not least it has an effect on application security. Attackers might be able to decompile the code and thereby discover a potentially sensitive address. They can perform a Denial of Service attack on the service, try to get access to the system, or try to spoof the IP address to bypass security checks. Such attacks can always be possible, but in the case of a hardcoded IP address solving the issue will take more time, which will increase an attack's impact.

Ask Yourself Whether

The disclosed IP address is sensitive, e.g.:

- Can give information to an attacker about the network topology.
- It's a personal (assigned to an identifiable person) IP address.

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

Recommended Secure Coding Practices

Don't hard-code the IP address in the source code, instead make it configurable with environment variables, configuration files, or a similar approach. Alternatively, if confidentially is not required a domain name can be used since it allows to change the destination quickly without having to rebuild the software.

Sensitive Code Example

```
String ip = "192.168.12.42"; // Sensitive
Socket socket = new Socket(ip, 6667);
```

Compliant Solution

```
String ip = System.getenv("IP_ADDRESS"); // Compliant
Socket socket = new Socket(ip, 6667);
```

Exceptions

No issue is reported for the following cases because they are not considered sensitive:

- Loopback addresses 127.0.0.0/8 in CIDR notation (from 127.0.0.0 to 127.255.255.255)
- Broadcast address 255.255.255.255
- Non routable address 0.0.0.0
- Strings of the form `2.5.<number>.<number>` as they often match Object Identifiers (OID).

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- CERT, MSC03-J. - Never hard code sensitive information

Available In:
sonarcloud sonarqube

Local-Variable Type Inference should be used

Code Smell

Migrate your tests from JUnit4 to the new JUnit5 annotations

Code Smell

Track uses of disallowed classes

Code Smell

Track uses of "@SuppressWarnings" annotations

Code Smell