Security

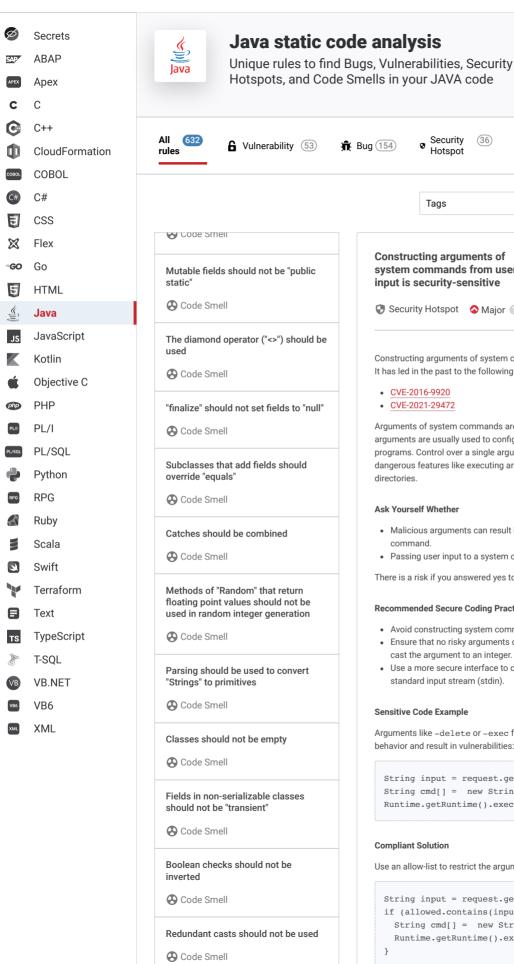
Hotspot

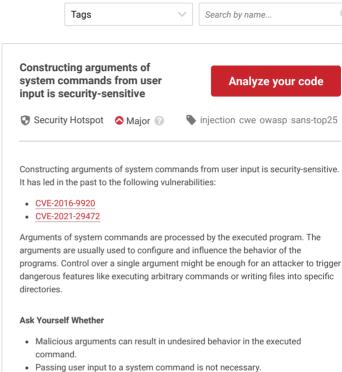
(36)



Products >

Quick 42





⊗ Code

Smell

(389)

## **Recommended Secure Coding Practices**

· Avoid constructing system commands from user input when possible.

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

- Ensure that no risky arguments can be injected for the given program, e.g., typecast the argument to an integer.
- Use a more secure interface to communicate with other programs, e.g., the standard input stream (stdin).

## Sensitive Code Example

Arguments like -delete or -exec for the find command can alter the expected behavior and result in vulnerabilities:

```
String input = request.getParameter("input");
String cmd[] = new String[] { "/usr/bin/find", input };
Runtime.getRuntime().exec(cmd); // Sensitive
```

## **Compliant Solution**

Use an allow-list to restrict the arguments to trusted values:

```
String input = request.getParameter("input");
if (allowed.contains(input)) {
 String cmd[] = new String[] { "/usr/bin/find", input };
 Runtime.getRuntime().exec(cmd);
```

See

"@Deprecated" code should not be

"toString()" should never be called on a String object

Code Smell

Annotation repetitions should not be wrapped

Code Smell

Multiple variables should not be declared on the same line

Code Smell

Strings should not be concatenated using '+' in a loop

OWASP Top 10 2021 Category A3 - Injection
 OWASP Top 10 2017 Category A1 - Injection
 MITRE, CWE-88 - Argument Injection or Modification
 SANS Top 25 - Insecure Interaction Between Components
 CVE-2021-29472 - PHP Supply Chain Attack on Composer

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