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...

HTTP request redirections should not be open to forging attacks
Vulnerability

Deserialization should not be vulnerable to injection attacks
Vulnerability

Endpoints should not be vulnerable to reflected cross-site scripting (XSS) attacks
Vulnerability

Database queries should not be vulnerable to injection attacks
Vulnerability

XML parsers should not be vulnerable to XXE attacks
Vulnerability

A secure password should be used when connecting to a database
Vulnerability

XPath expressions should not be vulnerable to injection attacks
Vulnerability

I/O function calls should not be vulnerable to path injection attacks
Vulnerability

LDAP queries should not be vulnerable to injection attacks
Vulnerability

OS commands should not be vulnerable to command injection attacks
Vulnerability

"@SpringBootApplication" and "@ComponentScan" should not be used in the default package
Bug

Extracting archives should not lead to zip slip vulnerabilities

Analyze your code

Vulnerability | Blocker | injection cwe owasp sans-top25

File names of the entries in a zip archive should be considered untrusted, tainted and should be validated before being used for file system operations. Indeed, file names can contain specially crafted values, such as '../', that change the initial path and, when accessed, resolve to a path on the filesystem where the user should normally not have access.

A successful attack might give an attacker the ability to read, modify, or delete sensitive information from the file system and sometimes even execute arbitrary operating system commands. This special case of path injection vulnerabilities is called "zip slip".

The mitigation strategy should be based on the whitelisting of allowed paths or characters.

Noncompliant Code Example

```
public static List<String> zipSlipNoncompliant(ZipFile zip
  Enumeration<? extends ZipEntry> entries = zipFile.entrie
  List<String> filesContent = new ArrayList<>();

  while (entries.hasMoreElements()) {
    ZipEntry entry = entries.nextElement();
    File file = new File(entry.getName());
    String content = FileUtils.readFileToString(file, Stan
    filesContent.add(content);
  }

  return filesContent;
}
```

Compliant Solution

```
public static List<String> zipSlipCompliant(ZipFile zipFil
  Enumeration<? extends ZipEntry> entries = zipFile.entrie
  List<String> filesContent = new ArrayList<>();

  while (entries.hasMoreElements()) {
    ZipEntry entry = entries.nextElement();
    File file = new File(entry.getName());
    String canonicalDestinationPath = file.getCanonicalPat

    if (!canonicalDestinationPath.startsWith(targetDirecto
      throw new IOException("Entry is outside of the targe
    }

    String content = FileUtils.readFileToString(file, Stan
    filesContent.add(content);
  }
```

"@Controller" classes that use "@SessionAttributes" must call "setComplete" on their "SessionStatus" objects

Bug

"wait" should not be called when multiple locks are held

Bug

"PreparedStatement" and "ResultSet" methods should be called with valid indices

Bug

Files opened in append mode should not be used with ObjectOutputStream

```
    return filesContent;
  }
```

See

- OWASP Top 10 2021 Category A1 - Broken Access Control
- OWASP Top 10 2021 Category A3 - Injection
- OWASP Top 10 2017 Category A1 - Injection
- snyk - Zip Slip Vulnerability
- MITRE, CWE-20 - Improper Input Validation
- MITRE, CWE-22 - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- MITRE, CWE-99 - Improper Control of Resource Identifiers ('Resource Injection')
- MITRE, CWE-641 - Improper Restriction of Names for Files and Other Resources
- SANS Top 25 - Risky Resource Management

Available In:

sonarcloud | sonarqube Developer Edition