WHY YOU SHOULD CARE

JAVA VULNERABILITIES AND...
I AM A DEVELOPER
NOT A SECURITY EXPERT
24th November 2021
20th Century

Software landscape
20th Century

Software landscape

- Code was self written and closed source
- Source code was managed in a repository on a local server
- Manually build
- Delivered on hardware (CD, DVD, USB-Sticks)
- Ran on closed networks or local servers
- Large monolithic systems
- Connected systems only in government / banking / energy providers
- Full control over the source code
20th Century Vulnerabilities
20th Century Vulnerabilities

Vulnerabilities

- Password hacking / cracking
- Computer viruses (spread via floppy discs/usb sticks)
- Early days of hacking via internet
21st Century

Software landscape
21ST CENTURY

Software landscape

- A lot of open source software used
- Distributed source code management systems
- Automated builds by CI / CD systems
- Hosted in artifact repositories
- Running on public networks
- Accessible via browsers or api's
- "Everything" is connected
- No full control over the source code
- Today we have a whole software supply chain
21ST CENTURY VULNERABILITIES

Vulnerabilities

- Danger through Social Engineering (SIM swapping etc.)
- Malware / Ransomware (spread via mail / websites)
- Everything that is connected, will be hacked
- Spreading malicious code is way easier
- The whole software supply chain is target of attacks
SOME DEFINITIONS
CWE
Common Weakness Enumeration
Common Weakness Enumeration

Community developed list of software and hardware weakness types.

https://cwe.mitre.org/
National Vulnerability Database

U.S. government repository of standards based vulnerability management data, represented using the Security Content Automation Protocol (SCAP)

https://nvd.nist.gov/
CVE
Common Vulnerability + Exposure
CVE Program Mission

"Identify, define, and catalog publicly disclosed cybersecurity vulnerabilities"

https://cve.org/
CVE-2021-44228
CVE-2021-44228
Log4Shell

Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled.

From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.

https://nvd.nist.gov/vuln/detail/CVE-2021-44228
CVE-2021-44228
Log4Shell

Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled. From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.

https://nvd.nist.gov/vuln/detail/CVE-2021-44228
CVE-2021-44228

Log4Shell

Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled.

From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.

https://nvd.nist.gov/vuln/detail/CVE-2021-44228
CVE-2021-44228

Log4Shell

Apache Log4j2 2.0-beta9 through 2.15.0 (excluding security releases 2.12.2, 2.12.3, and 2.3.1) JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled.

From log4j 2.15.0, this behavior has been disabled by default. From version 2.16.0 (along with 2.12.2, 2.12.3, and 2.3.1), this functionality has been completely removed. Note that this vulnerability is specific to log4j-core and does not affect log4net, log4cxx, or other Apache Logging Services projects.

https://nvd.nist.gov/vuln/detail/CVE-2021-44228
CVSS

Common Vulnerability Severity Score
### CVSS

**Common Vulnerability Severity Score**

Vulnerability Severity Ratings (CVSS v2.0)

<table>
<thead>
<tr>
<th>Severity</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>0.0 - 3.9</td>
</tr>
<tr>
<td>Medium</td>
<td>4.0 - 6.9</td>
</tr>
<tr>
<td>High</td>
<td>7.0 - 10.0</td>
</tr>
</tbody>
</table>

https://nvd.nist.gov/vuln-metrics/cvss
# CVSS

Common Vulnerability Severity Score

<table>
<thead>
<tr>
<th>Severity</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0.0</td>
</tr>
<tr>
<td>Low</td>
<td>0.1 - 3.9</td>
</tr>
<tr>
<td>Medium</td>
<td>4.0 - 6.9</td>
</tr>
<tr>
<td>High</td>
<td>7.0 - 8.9</td>
</tr>
<tr>
<td>Critical</td>
<td>9.0 - 10.0</td>
</tr>
</tbody>
</table>

[https://nvd.nist.gov/vuln-metrics/cvss](https://nvd.nist.gov/vuln-metrics/cvss)

# Vulnerability Severity Ratings (CVSS v3.1)

CVSS v4.0

1st Nov 2023
CVE-2021-44228 (Log4Shell)

CVSS v2.0

Severity: High
Score: 9.3

CVSS v3.1

Severity: Critical
Score: 10.0

https://nvd.nist.gov/vuln/detail/CVE-2021-44228
IS JAVA SECURE?
OPENJDK
VULNERABILITY
GROUP
OPENJDK VULNERABILITY GROUP

What is it...?

- Private forum (trusted members of the OpenJDK community)
- Receives/reviews reports of vulnerabilities in the OpenJDK code base
- Collaborates on fixing the issues
- Coordinates the release of such fixes
- Maintains list of CVE's patched for each release
- Tracks CVE's by component (not all Java users leverage every component)
- Discusses OpenJDK security related issues
- Does not actively test the OpenJDK source code
OPENJDK VULNERABILITY ADVISORIES
OpenJDK Vulnerability Advisories

Published 4x a year

Describing
- Severity
- Area
- Affected versions

https://openjdk.org/groups/vulnerability/advisories/
# OpenJDK Vulnerability Advisories

**Example 17th of October 2023**

## OpenJDK Risk Matrix

<table>
<thead>
<tr>
<th>CVE ID</th>
<th>Component</th>
<th>CVSSv3.1 Vector</th>
<th>Affects ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2023-22067</td>
<td>other-libs/corba</td>
<td>5.3</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NLNNUNLN</td>
<td>11</td>
</tr>
<tr>
<td>CVE-2023-22081</td>
<td>security-libs/javax.net.ssl</td>
<td>5.3</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NLNNUNLN</td>
<td>21</td>
</tr>
<tr>
<td>CVE-2023-22025</td>
<td>hotspot/compiler</td>
<td>3.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NHNNUNLN</td>
<td></td>
</tr>
</tbody>
</table>

## OpenJFX Risk Matrix

<table>
<thead>
<tr>
<th>CVE ID</th>
<th>Component</th>
<th>CVSSv3.1 Vector</th>
<th>Affects ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>

## Acknowledgements

We acknowledge the following parties for their reports and contributions: Carter Kozak, and Dinglijie.

We also thank the Leads of the [JDK 8 Updates](#), [JDK 11 Updates](#), [JDK 17 Updates](#), and [OpenJFX Projects](#) for providing the risk-matrix information for their releases.

## How to report a vulnerability

Please see the [reporting instructions](#) for information about how to report a vulnerability.
JAVA RELEASE CYCLE
JAVA RELEASE CYCLE

New Cadence

- 6 MONTHS
- 3 YEARS
- 2 YEARS

2017 LTS
2018
2019
2020 LTS
2021 LTS
2022
2023 LTS
2024
2025
BUT HOW DOES THAT HELP?
Java Release Cycle

Features per release

- JDK 8 LTS in Mar 2014: 56 features
- JDK 9 in Sep 2017: 91 features
- JDK 9 in Mar 2018: 91 features
- JDK 9 in Sep 2018: 91 features
- JDK 9 in Mar 2019: 91 features
- JDK 9 in Sep 2019: 91 features
- JDK 9 in Mar 2020: 91 features
- JDK 9 in Sep 2020: 91 features
- JDK 9 in Mar 2021: 91 features
- JDK 9 in Sep 2021: 91 features
- JDK 9 in Mar 2022: 91 features
- JDK 9 in Sep 2022: 91 features
- JDK 9 in Mar 2023: 91 features
Java Release Cycle

Features per release

<table>
<thead>
<tr>
<th>Year</th>
<th>JDK</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 2014</td>
<td>JDK 8</td>
<td>56</td>
</tr>
<tr>
<td>Sep 2017</td>
<td>JDK 9</td>
<td>91</td>
</tr>
<tr>
<td>Mar 2017</td>
<td>JDK 10</td>
<td>17</td>
</tr>
<tr>
<td>Sep 2018</td>
<td>JDK 11 LTS</td>
<td>12</td>
</tr>
<tr>
<td>Mar 2018</td>
<td>JDK 12</td>
<td>17</td>
</tr>
<tr>
<td>Sep 2019</td>
<td>JDK 13</td>
<td>8</td>
</tr>
<tr>
<td>Mar 2019</td>
<td>JDK 14</td>
<td>5</td>
</tr>
<tr>
<td>Sep 2019</td>
<td>JDK 15</td>
<td>16</td>
</tr>
<tr>
<td>Mar 2020</td>
<td>JDK 16</td>
<td>14</td>
</tr>
<tr>
<td>Sep 2020</td>
<td>JDK 17 LTS</td>
<td>17</td>
</tr>
<tr>
<td>Mar 2021</td>
<td>JDK 18</td>
<td>14</td>
</tr>
<tr>
<td>Sep 2021</td>
<td>JDK 19</td>
<td>5</td>
</tr>
<tr>
<td>Mar 2022</td>
<td>JDK 20</td>
<td>7</td>
</tr>
<tr>
<td>Sep 2022</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>Mar 2023</td>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>
Less features per release means less potential vulnerabilities per release.
JAVA
UPDATES
CPU

Critical Patch Update
Critical Patch Update

Contains

- Fixes vulnerabilities
- Fixes critical issues

Safe to use in production
PSU

Patch Set Update
Could possibly introduce new vulnerabilities !!!

PSU
Patch Set Update

Superset of CPU

Contains

- Fixes vulnerabilities
- Fixes critical issues
- Fixes non critical issues
- New features
Four times a year

January
**UPDATES**

Four times a year

January  |  April

17.0.2 PSU CPU | 17.0.3 PSU CPU

Vulnerability Fixes

Fixes and Features

17.0.2 CPU | 17.0.3 CPU
Four times a year

January

April

July
Updates

Four times a year

January
17.0.2 PSU CPU
17.0.2 CPU

April
17.0.3 PSU CPU
17.0.3 CPU

July
17.0.4 PSU CPU
17.0.4 CPU

October
17.0.5 PSU CPU
17.0.5 CPU

Fixes and Features

Vulnerability Fixes
Keep in mind

- Updates are available 4 times a year (every 3 months starting from January).
- Patch Set Updates (PSU) contain the CPU plus non-critical fixes and small features.
- Critical Patch Updates (CPU) contain only critical vulnerability fixes and are feature-wise always one step behind the PSU.
Why CPUs matter

- PSU 8u252 introduced a change that prevented Hadoop cluster and Solr from running
- CPU 8u251 only contained security fixes from PSU 8u242 and did not introduce this change
IMPACT WITHOUT UPDATES
JDK 17

14.09.2021

17.0.0

19.10.2021

17.0.1

CVE-2021-35567 6.8
CVE-2021-35586 5.9
CVE-2021-35564 5.3
CVE-2021-35561 5.3
CVE-2021-35559 5.3
CVE-2021-35578 5.3
CVE-2021-35556 5.3
CVE-2021-35603 3.7

8 CVE's
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>CVEs</th>
</tr>
</thead>
</table>
If you stick to 17.0.0 you are vulnerable to 54 CVE's !!!
If it ain't broke at least keep it up to date!
MODULAR
RUNTIME
IMAGES
Reducing risk by removing modules

JLink makes this possible (since JDK9 introduced the Java Platform Module System JPMS)

Removing unused modules means reducing risk for vulnerabilities

Hackers cannot attack what isn't there

Your application doesn't need to be modular
MODULAR RUNTIME IMAGES

Java Platform Module System

JDK 21.0.1

69 Modules

339 MB
modular runtime images

Java Platform Module System (JRE 21)

50 Modules
150 MB
Modular Runtime Images

Java Platform Module System (JLINK JRE 21)

11 Modules
48 MB
MODULAR RUNTIME IMAGES

Java Platform Module System

JDK 21.0.1
69 Modules
339 MB

JRE 21.0.1
50 Modules
150 MB

JLINK 21.0.1
11 Modules
48 MB
SOFTWARE
SUPPLY CHAIN
SOFTWARE SUPPLY CHAIN

Central Repo
Development
Repository
submit
build
CI/CD
Artifact Repo
deploy
Production
SOFTWARE SUPPLY CHAIN

And it's vulnerabilities

- Central Repo
- Development
- Repository
- CI/CD
- Artifact Repo
- Production

Submit bad code
Compromise source control
Build from modified source
Compromise build platform
Upload modified package
Compromise package repo
Use compromised package

Use compromised dependency
SOME FACTS
Software Supply Chain attacks

742% increase over the past 3 years

(Sonatype State of the Software Chain report)
The year 2021 saw 20,142 unique bugs and security vulnerabilities recorded.
Of all Log4j downloads, 20% are still vulnerable to CVE 2021-44228, even 21 months after Log4j has been patched!

(Christian Grobmeier, Log4j maintainer)
About 6 out of 7 project vulnerabilities come from transitive dependencies

(Sonatype State of the Software Chain report)
At least one vulnerability in 84% of all scanned codebases and 48% contained high-risk vulnerabilities.

(Synopsys OSS security and risk analysis report)
Open Source is everywhere

Open Source used in 96% of all scanned codebases and 76% of code in codebases was Open Source

(Synopsys OSS security and risk analysis report)
OPEN SOURCE
THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
MEANS...
I OWE YOU NOTHING!
Keep in mind that:

- OpenSource maintainers are not suppliers!
- You don't have a business relationship with them!
- If you use their code, it's up to you to make sure it's up to date and secure!

https://xkcd.com/2347/
WHAT CAN WE DO?
SHIFT LEFT
SHIFT LEFT

Software Supply Chain
SHIFT LEFT

Security should start more on the left side of the diagram

Software Supply Chain
DEV OPS LOOP

Plan
DevOps Loop

Code
DevOps Loop

Build
Release

Dev Ops Loop

Code

Plan

Build

Release
DEV OPS LOOP
Deploy

CODE
PLAN
RELEASE
BUILD
DEPLOY
DevOps Loop
Operate

- Code
- Plan
- Deploy
- Operate
- Build
- Release
DEV OPS LOOP

Monitor
DevOps Loop

Plan

- Code
- Build
- Release
- Deploy
- Monitor
- Operate
What about security?
DEV SEC OPS

- Code
- Plan
- Deploy
- Operate
- Build
- Release
- Monitor

SEC

azul
SECURITY APPLIES TO ALL AREAS

CODE | PLAN | DEPLOY
---|---|---
BUILD | RELEASE | OPERATE

SEC

DEV | OPS
YES...BUT

ALSO
VALIDATE
RIGHT
UPDATE YOUR JDK
SDKMAN

Command line application

https://sdkman.io/

Facts

- Supports many JDK distributions
- Commandline only
- Linux, MacOS
- Downloads and installs JDK's
Desktop application

https://github.com/HanSolo/JDKMon/releases

Facts

- Info about JDK updates
- Supports "all" JDK distributions
- Downloads JDK's
- Taskbar application
- Windows, Linux, MacOS
- Shows CVE's in OpenJDK
Desktop application

Installed JDK distributions

Vulnerabilities found for JDK

- CVE-2021-2368 (CVSS 3.1 Score 7.6, Severity High)
- CVE-2021-2369 (CVSS 3.1 Score 4.3, Severity Medium)
- CVE-2021-2341 (CVSS 3.1 Score 3.1, Severity Low)
STATIC CODE ANALYSIS
STATIC CODE ANALYSIS

What is it?

- Usually part of a code review (white-box testing)
- Identifies vulnerabilities in source code
- At the implementation phase
- Inexpensive because adjustments can be done easily
- Standalone tools / IDE plugins
STATIC CODE ANALYSIS

Source Code Security Analyzers

- AppSonar/CodeSonar by CyberTest
- Codiga by Codiga
- DerScanner by DerSecur Ltd.
- FindSecurityBugs free
- Snyk Code by Snyk Limited
- SonarQube by SonarSource
- Static Reviewer by Security Reviewer

FIND SECURITY BUGS
Find security bugs

SpotBugs plugin

Facts

- Free of charge
- Extends SpotBugs
- 400+ bug patterns
- Plugin

https://find-sec-bugs.github.io/
FIND SECURITY BUGS

Eclipse Plugin
VULNERABILITY SCANNERS
VULNERABILITY SCANNERS

What is it?

- Detect vulnerabilities *(using a database / probing for common flaws)*
- Monitor misconfigurations and coding flaws
- Help using only artifacts from reliable sources
- Help using only latest secure version *(without known vulnerabilities)*
- Monitor appearance of new packages with fixed vulnerabilities
- Update dependencies *(as soon as new versions are available)*
VULNERABILITY SCANNERS

How they work

SCAN, FIX

SCAN, FIX, MONITOR

SCAN, FIX, MONITOR

SCAN

MONITOR

SourceCode → Repository → CI/CD → Artifact Repo → Production

submit → build → deploy

often agent based
VULNERABILITY SCANNERS

Agent based monitoring
VULNERABILITY SCANNERS

Agent based monitoring
Agent based monitoring

**CLASS LOADER SUBSYSTEM**
- **LOADING**
  - Bootstrap Class Loader
  - Extension Class Loader
  - Application Class Loader
- **LINKING**
  - Verification
  - Preparation
  - Resolution
- **INITIALISING**
  - Initialisation

**JVM MEMORY**
- **STACK**
  - Thread #1
  - Thread #2
  - Thread #3
- **PC REGISTER**
  - Thread #1 PC
  - Thread #2 PC
  - Thread #3 PC

**EXECUTION ENGINE**
- **JIT COMPILER**
  - C1
  - C2
- **PROFILER**
  - Serial
  - Parallel
  - CMS
  - Z
- **GARBAGE COLLECTOR**
  - Native Method Interface (JNI)
  - Native Method Libraries
VULNERABILITY SCANNERS

Agent based monitoring

Performance hit of 10% or more!
VULNERABILITY SCANNERS

For Java development

- Azul Vulnerability Detection by Azul
- Black Duck by Synopsis
- Xray by JFrog
- Snyk by Snyk Limited
- SonarQube by SonarSource
- Trivy by Aqua
Snyk Code
Static application Security Testing

Facts
- Free and paid version
- 9+ languages supported
- Developer first
- Standalone
- IDE Plugin available
- CI/CD integration

https://snyk.io/product/snyk-code/
IntelliJ Plugin

SQL injection
Vulnerability | CVE-99

Unsanitized input from the HTTP request body flows into find, where it is used in an SQL query. This may result in injection vulnerability.

Data Flow - 2 steps
1. 
2. 

External example fixes

This issue was fixed by 91 projects. Here are 3 example fixes:

WHERE?

WHAT?
SONARQUBE
SONARQUBE
Automatic code review tool

https://www.sonarsource.com/products/sonarqube/

Facts

- Free and paid version
- 30+ languages
- 4800+ analysis rules
- Standalone
- Plugin available
- CI/CD integration
Intellij Plugin

WHERE?
WHAT?
WHY?
AZUL
VULNERABILITY DETECTION
Azul Vulnerability Detection

Facts

- Runs in production
- JVM only
- Fewer false positives
- Does code inventory
- No Java agent → no performance overhead

https://www.azul.com/products/vulnerability-detection/
AZUL VULNERABILITY DETECTION

Web UI
A secure software supply chain
EXAMPLE

Central Repo

Development

submit

Repository

build

CI/CD

Artifact Repo

deploy

Production
TAKEAWAY
Follow an automated patch schedule (in line with your OpenJDK vendors quarterly patch cycle)

Automate application packaging with jlink (removing modules that are not used by your application)

Watch for CVE's in libraries (automate their updates in the line with the OpenJDK quarterly patch schedule)

Use vulnerability scanners (not only in development and CI/CD but also in production)
NEED TO BE A SECURITY EXPERT?
No...
YOU NEED TO BE AWARE
STAY SECURE