## Are Security Leaders Overconfident About the State of Their Java Security?

azu

CRITICAL

**CVEs** 

IN 3RD PARTY JAVA APPS

AG11 10

e<sup>d</sup>11 100

101101

**310 111111 0**1

9111101

10111000 1010

0 0000 101 1

101 11

010110 0011010

01010

11111101000 011110111

**00 11010** 

001100

991191 999191111

0011010

HIGH RISK

**CVEs** 

11011010110

9901101 0 0 00 100111

011010

0100

01101

10

1

01111

1010001111000001011010

3111000 1

110001 11001 0000110100

01 0

1101 0 0

1 00 100111 10000011010

10 00000111000 1

**00 01** 

101101 00011000

0011 000101 0 1

01101 010000101110001101000010 111111 010

0100011000010111010

**01**0 11 101101000 101100101 010

1 1011 10

1 01 1

01110 011

1004 00001101000010110 001101000110

011110111

991911**199911919** 

110010

#### Security professionals are overconfident

Security professionals think they are prepared for anything

- **95%** feel confident in the accuracy of their security tools<sup>1</sup>
- 100% say they are confident in their company's state of Java security<sup>2</sup>
- **93%** think Java application developers have adequate security skills and expertise<sup>2</sup>

Data and recent trends tell a different story. Java libraries, the Log4Shell vulnerability, data breaches, unsupported Java versions and unpatched vulnerabilities are all sources of danger. False positives and irrelevant notifications overwhelm users with alert fatigue.

### Third-party code is proliferating

- The number of libraries is growing, and 35% of Java developers say managing third-party libraries is becoming more difficult<sup>2</sup>
- CVEs are growing 13 high-risk and 3 critical CVEs over a 3-year period (2018-20)
- In Q1 2022 alone there were 200+ CVEs in 3rd party Java applications and components, many with the highest risk score, by thousands of library authors, and each library has its own release schedule

### Vulnerabilities are increasing in number and severity, as evidenced by Log4Shell

- In its first weekend in Q4 2021, there were **37,000** attempts to allocate the Spring4Shell vulnerability
- About **58%** of Java applications package a vulnerable version of Log4Shell<sup>3</sup>
- Check Point researchers have already identified more than 60 variations of the

original exploit code<sup>4</sup>

× C

# Teams have not universally adopted software bills of material (SBOMs)

- 100% of security professionals say they know who is responsible for in-house SBOMs<sup>2</sup>
- But they don't agree on WHO is responsible (and they can't all be right)
  - o 43% say Application Developers
  - o **30%** say line of Business Executives
  - o 23% say IT Security
  - o 3% say Business Operations

Azul Vulnerability Detection provides a new way to keep Java environments safe. For more information, visit azul.com/products/vulnerability-detection

1 Orca Security, 2022 2 Azul, 2022 3 Contrast Security, 2021 4 TechTarget, 2021

Copyright © 2022 Azul Systems, Inc. 385 Moffett Park Drive Suite 115, Sunnyvale, CA 940891306. All rights reserved. Azul Systems, the Azul Systems logo, Zulu and Zing are registered trademarks, and ReadyNow! is a trademark of Azul Systems Inc. Java and OpenJDK are trademarks of Oracle Corporation and/or its affiliated companies in the United States and other countries. Monotype is a trademark of Monotype Imaging Inc. registered in the United States Patent and Trademark Office and may be registered in certain other jurisdictions. The Monotype logo is a trademark of Monotype Imaging Inc. and may be registered in certain jurisdictions. Other marks are the property of their respective owners and are used here only for identification purposes. Products and specifications discussed in this document may reflect future versions and are subject to change by Azul Systems without notice.

#### Contact Azul

385 Moffett Park Drive, Suite 115 Sunnyvale, CA 94089 USA +1.650.230.6500 www.azul.com

azu