Data Sheet

Azul Vulnerability Detection: Continuously Detect Known Vulnerabilities in Your Java Applications

Detect Vulnerabilities with No Performance Penalty and Eliminate False Positives

Businesses are under relentless pressure to

- speed up application innovation cycles
- accelerate time to market
- fortify application security

while grappling with scant developer resources. DevOps productivity suffers from alert fatigue due to out-of-control vulnerability false positives.

Inefficient prioritization of vulnerabilities wastes effort, hampers agility, and reduces developer productivity due to constant context switching and unproductive remediation tasks.

Azul Intelligence Cloud is a DevOps analytics solution that provides actionable intelligence from production Java runtime data, pinpointing what actually runs to efficiently prioritize vulnerable code for remediation (Vulnerability Detection) and identify unused code for removal (Code Inventory), boosting DevOps productivity.

Azul Vulnerability Detection at-a-glance

Continuous Detection at Point-of-Use in Production.
Accurately assesses custom and 3rd party applications' exposure to vulnerabilities in production. Compares code run to a Java-specific CVE database.

Eliminates False Positives
Eliminates false positives by monitoring the code loaded by the JVM.

Runs in Production with no Performance Impact
Efficiently captures Java runtime data that exists within a JVM when running a Java application, resulting in no performance impact.

Detects without Source Code
Recognizes components using unique Hash-based identifiers.

Java CVE Knowledge Base
Updated daily with the latest CVEs filtered to focus on Java-specific vulnerabilities

Azul Vulnerability Detection, a feature of Azul Intelligence Cloud, uses information the JVM inherently has when running a Java application to identify vulnerable code that actually runs, generating accurate results unattainable by traditional application security tools. It enables DevOps to prioritize vulnerabilities based on actual risk, saving time while reducing security issue backlogs and improving production security posture.

Azul Vulnerability Detection monitors all your Java software and applications to accurately identify components loaded and in use in production. Azul Vulnerability Detection uniquely identifies each component using bytecode-aware hashing techniques. It maps these components accurately to vulnerabilities in a knowledge base updated daily with the latest CVEs from external databases, publicly available information, and more.

This enables accurate, continuous assessment of custom and vendor applications exposure to vulnerabilities in production without the need for source code. Azul Vulnerability "just works" to detect vulnerabilities in all Java applications – whether you built it or not, haven't built it in years, or are introducing a regression with a recent change.

Azul Vulnerability Detection focuses scarce human effort by eliminating false positives. It does this by monitoring the code loaded and in use by applications running on the JVM vs looking at static file listings and source code.

Inside Azul Vulnerability Detection

Azul Vulnerability Detection uses runtime information the JVM already has and sends it to a backend cloud service for detection of vulnerabilities.

- Forwarder: JVMs connect to the Intelligence Cloud Service through a Forwarder. The Forwarder is a piece of software Azul provides that acts as an application-specific proxy so that JVMs can reach the Intelligence Cloud Service without needing special firewall rules.

Azul Vulnerability Detection Advantages

- Continuous detection of vulnerabilities in production
- Checks all Java software – custom and vendor applications, 3rd party libraries
- Supports any JVM from any vendor or distribution: Azul, Oracle, Amazon, Microsoft, Red Hat, Temurin
- Focuses human effort by eliminating false positives
- Runs in production with no performance penalty
- Uses information the JVM inherently has when running a Java application to identify vulnerable code that actually runs
- Generates complete results – works on all major packaging structures including shaded jars, fat jars

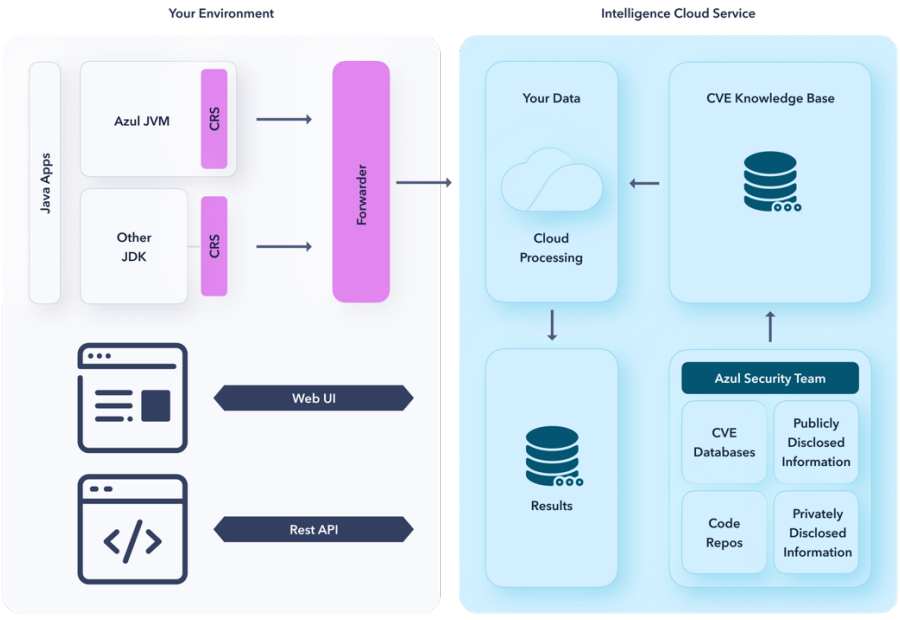
Data Sheet

Azul Vulnerability Detection: Continuously Detect Known Vulnerabilities in Your Java Applications

- CVE Knowledge Base contains information about known CVEs. Azul's security team filters these down to understand which relate to Java and which customers should pay attention to. As a result of the CVE knowledge base, customers can focus on risks unique to Java.
- Component Knowledge Base is a recognition set that Azul Vulnerability Detection uses to identify many open-source components. Once recognized, Azul Vulnerability Detection checks the known component against the CVE Knowledge Base to see if it is vulnerable.
- Azul Vulnerability Detection is not another dashboard for customers to look at. Users can access data on which components are in use, and vulnerable, using either the product's API or an intuitive UI. The role of the web UI is to show the information we have and guide customers to the REST API.

Azul Vulnerability Detection Features

- Supports Java SE 21, 17, 15, 13, 11, or 8
- Supports any JVM from any JDK vendor or distribution, including Azul Zulu and Zing Builds of OpenJDK, Oracle JDK, Amazon Corretto, Microsoft Build of OpenJDK, Red Hat build of OpenJDK and Eclipse Temurin
- Detection API to access present, used, vulnerable component level information
- Web UI for looking at data quickly and as a guide to the data available in the Detection API.
- Forwarder component that facilitates communication between JVMs on an internal network and Azul Intelligence Cloud.
- Component Knowledge Base for component recognition using byte-code aware hashing and unique component identifiers
- CVE Knowledge Base updated daily with Java related CVEs in accordance with National Vulnerability Database



Contact Azul

385 Moffett Park Drive, Suite 115

Sunnyvale, CA 94089 USA

+1.650.230.6500

www.azul.com

Azul Vulnerability Detection Advantages Cont.

- Catches regressions and reintroductions of vulnerabilities
- More precise component recognition using bytecode-aware hashing techniques
- Focus on Java related vulnerabilities using Knowledge Base updated daily with latest CVEs from National Vulnerability Database
- Creates an accurate SBOM of components loaded and in use in production