CYBER THREAT INTELLIGENCE

CRITICALSTART® Security Advisory

TLP CLEAR // [CS-SA-26-1102] Java4Shell Vulnerability

Executive Summary

The CRITICALSTART® Cyber Research Unit (CRU) is tracking Java4Shell (CVE-2026-114938), a memory-corruption flaw in the Open Java Development Kit (OpenJDK) Base64 decoder. Malformed Base64 input makes an optimized routine write past the end of a heap buffer on x86-64 servers that support Advanced Vector Extensions 512 Vector Byte Manipulation Instructions (AVX-512 VBMI).[1] Because servers decode Base64 in login headers and tokens, the flaw is often reachable without credentials. Calif, which discovered the flaw, demonstrated unauthenticated Remote Code Execution (RCE) in six enterprise applications including Atlassian's Jira and Confluence, while independent labs have so far reproduced crashes and memory leaks.[1, 2] Oracle and OpenJDK released fixes on November 17, 2026, several downstream builds are still pending, and no exploitation in the wild is confirmed yet. Organizations should find affected Java runtimes on affected hardware, then patch immediately per CISA or apply the startup flag mitigation, starting with internet-facing systems.

Overview

Calif researcher Quang Luong found the flaw through AI-assisted auditing of the Java runtime and reports building a first working exploit within hours, for under $200 in AI costs. Calif coordinated disclosure with Oracle and the OpenJDK project, which published fixes at 1:00 PM Pacific Time on November 17.[1] The trigger was partly public earlier: an unmerged OpenJDK code proposal posted on September 9, 2026, described the same overflow and included a triggering input.[2] CRU assesses with moderate confidence that capable actors could have identified the bug before disclosure. No threat actor has been linked to exploitation at the time of writing.

Technical Details

- **The bug:** Once Java's Base64 decoder runs often enough, the HotSpot Just-In-Time (JIT) compiler swaps in a hand-written AVX-512 routine. A malformed padded tail, such as 64 `A` characters followed by `=`, makes an internal length counter go negative. The routine then writes 62 or 63 bytes past the end of the output array. Java rejects the input only after the write.[1]
- **From crash to code execution:** The overflow writes mostly fixed bytes, so attackers control only where they land. Calif arranges the heap so the overflow enlarges a neighboring array, which yields controlled write and read primitives. The read primitive, which Calif calls JavaBleed, can leak keys, tokens, and personal data from Java Virtual Machine (JVM) memory before any code runs. The write primitive then reaches the JVM's executable code region. Calif also reports a client-side variant that compromises a Java program connecting over Transport Layer Security (TLS) to a malicious server.[1]
- **Constraints:** There is no universal exploit yet as each application needs its own heap profile. The chains are sensitive to network latency, and failed attempts can crash the JVM. Reverse proxies that buffer complete requests, as nginx does by default, block some chains but not all.[1] Crashing a server is far easier: Blackpoint Cyber crashed a test Spring Boot application with unauthenticated login requests in about 20 seconds.[3]

© 2026 Critical Start. All rights reserved.

Relevant Vulnerability Context

CVE	CVSS	Description	Relevance
CVE-2026-114938	8.1 (v3.1, Oracle) 9.8 (RedHat)	Heap out-of-bounds write in the HotSpot x86-64 AVX-512 VBMI Base64 decoder	Unauthenticated network attack; memory disclosure, denial of service, and RCE

Scores diverge mainly on attack complexity; Red Hat's preliminary Common Vulnerability Scoring System (CVSS) score is 9.8.[2] Calif demonstrated pre-authentication RCE in default configurations of Keycloak, Atlassian Jira, Atlassian Confluence, JFrog Artifactory, Jenkins, and PingFederate, and post-authentication RCE in Apache Kafka, Apache Solr, Elasticsearch, Gerrit, and Spinnaker.[1] Exposure also requires Intel Xeon Ice Lake or newer, or AMD Zen 4 or newer. Standard Java 8 and 11 builds and all ARM64 systems are not affected.[1, 2]

MITRE ATT&CK Mapping

The mapping below, drawn from the MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework, describes behaviors that are in scope:

ID	Name	Relevance
T1595.002	Active Scanning: Vulnerability Scanning	Java4Shell-shaped probing observed by GreyNoise sensors[4]
T1190	Exploit Public-Facing Application	Pre-authentication RCE through Base64 in login headers and tokens
T1203	Exploitation for Client Execution	Compromise of a Java TLS client connecting to a malicious server
T1212	Exploitation for Credential Access	JavaBleed heap disclosure of keys and tokens
T1499.004	Endpoint Denial of Service: Application or System Exploitation	Unauthenticated JVM crash

Implications for Organizations

Organizations running affected Java versions should treat Java4Shell as a critical risk, particularly when hosting internet-facing applications such as Jira, Jenkins, Keycloak, and Confluence. Exposure depends on both the Java runtime and the underlying processor, not just the application version. The same container image can be vulnerable on one server but unaffected on another. Denial of service is the most immediate concern, as attackers can trigger JVM crashes through malformed Base64 inputs. Remote code execution requires more targeted exploitation, but successful attacks could compromise enterprise applications, expose sensitive data, and provide access to credentials or service tokens. Importantly, patching does not automatically eliminate the consequences of prior exploitation, as stolen credentials and active sessions may remain valid. Organizations must therefore assess potential compromise and credential exposure alongside software remediation.

© 2026 Critical Start. All rights reserved.

Prioritized Mitigation Strategies

To improve your organization's security posture in light of Java4Shell potential exploitation, CRU recommends the following:

Immediate (Within 24 Hours)

- Identify vulnerable runtimes. Inventory Java 17.0.6 and later, Java 21, 25, and 27, and the affected Java 8 Performance Pack builds. Include embedded runtimes, container images, and x86-64 systems supporting AVX-512 VBMI.
- Install the fixed Java builds listed in Appendix A, prioritizing internet-facing applications and services that process unauthenticated Base64 input. Restart Java services and rebuild affected container images.
- Apply temporary mitigations where no fix exists. Where patches are unavailable, launch the JVM with `-XX:-UseBASE64Intrinsics` to disable the vulnerable Base64 implementation. Alternatively, use `-XX:UseAVX=2` on supported x86-64 systems. Verify that the mitigation takes effect and remove the flags after installing fixed builds.

Short-Term (Within Two Weeks)

- Hunt for exploitation attempts by reviewing application logs for malformed Base64 inputs, unexpected `IllegalArgumentException` errors, JVM crashes, and unusual service restarts. Use the indicators in Appendix B to guide investigations.
- Investigate potential compromise by examining affected systems for suspicious activity, unauthorized access, and signs of memory corruption or code execution. Rotate potentially exposed credentials, signing keys, and service tokens, and revoke active sessions where compromise is suspected.
- Track outstanding vendor updates. Monitor security advisories from Red Hat, Eclipse Temurin, Microsoft, BellSoft, and other Java distributors. Also check products that bundle their own Java runtimes.

Structural

- Improve runtime visibility. Record Java versions, processor architectures, and hardware capabilities in asset inventories and Software Bills of Materials (SBOMs).
- Strengthen build security. Automate container image rebuilds when updated Java base images become available and verify that deployed workloads use patched runtimes.
- Expand security monitoring. Monitor Java applications for abnormal input patterns, unexpected crashes, and suspicious access to authentication and application data.

Conclusion

Java4Shell presents a significant risk to organizations running vulnerable Java runtimes on supported x86-64 hardware, particularly those exposing affected applications to unauthenticated network traffic. Although reliable RCE requires application-specific exploitation, the demonstrated memory corruption and relatively straightforward denial-of-service attacks make delayed remediation a serious concern. The September 9 disclosure of a triggering input also raises the possibility that capable threat actors could have identified the flaw before its public disclosure. Organizations should prioritize patching or applying temporary mitigations, validate remediation across embedded runtimes and container deployments, and investigate potentially exposed systems for signs of exploitation. Importantly, successful remediation must also address potential credential and session compromise, as patching alone does not invalidate secrets stolen before the update.

For more threat reports, including OPTIMJACK and Artex, visit Critical Start's Intel Hub. This advisory was written using the best intelligence available at the time and is subject to change as additional information becomes available.

© 2026 Critical Start. All rights reserved.

What Critical Start is Doing

The CRITICALSTART® Cyber Research Unit (CRU) disseminated a brief of the threat as part of the 11-18-2026 CTI Update and is providing this Security Advisory as a follow up. CRU will continue to monitor the situation and work closely with the SOC and Security Engineering team to implement any relevant detections and increase vigilance. Critical Start customers are encouraged to reach out to support@criticalstart.com for any further enquiries.

Further Reading

[1] Calif (Nov. 17, 2026). Java4Shell: Write once, RCE everywhere. https://calif.io/java4shell

[2] The Hacker News (Nov. 18, 2026). Java4Shell Flaw (CVE-2026-114938) Enables Pre-Auth RCE in Jira, Jenkins, Keycloak and More. https://thehackernews.com/2026/11/java4shell-cve-2026-114938-enables-pre.html

[3] Blackpoint Cyber (Nov. 2026). Java4Shell (CVE-2026-114938): Inside a Critical Flaw in Java's Base64 Decoder. https://blackpointcyber.com/blog/java4shell-cve-2026-114938-java-base64-decoder-flaw/

[4] Precursor Security (Nov. 18, 2026). Java4Shell: Critical OpenJDK RCE Vulnerability (live updates citing GreyNoise). https://www.precursorsecurity.com/blog/java4shell-critical-openjdk-rce-vulnerability

Appendices

Appendix A: Affected and Fixed Builds

This table reflects vendor patch availability as of November 18, 2026. Release status may change as vendors publish additional updates. The availability of a newer build does not necessarily confirm that it addresses CVE-2026-114938. Organizations should verify CVE-specific fixes against official vendor security advisories before considering affected systems remediated.

Distribution	Affected Versions	Fixed Release
Oracle Java SE 8 Performance Pack	8u513	8u513-perf
Oracle Java SE 17	17.0.6 through 17.0.21.0.3	17.0.21.1
Oracle Java SE 21	21.0.13.0.3 and earlier	21.0.13.1
Oracle Java SE 25	25.0.5.0.3 and earlier	25.0.5.1
Oracle Java SE 27	27.0.1.0.3 and earlier	27.0.1.1
Oracle GraalVM for JDK 17 and 21	17.0.21.0.3 and 21.0.13.0.3	17.0.21.1 and 21.0.13.1
Oracle GraalVM Enterprise Edition	21.3.20.0.3	21.3.20.1
Amazon Corretto 17, 21, 25, 27	Earlier builds; Java 17 from 17.0.6	17.0.21.1.2, 21.0.13.1.2, 25.0.5.1.2, 27.0.1.1.4
Azul Zulu 17, 21, 25, 27	Earlier builds; Java 17 from 17.0.6	17.70.201, 21.54.201, 25.38.201, 27.30.201
Eclipse Temurin 17, 21, 25, 27	Earlier builds; Java 17 from 17.0.6	17.0.21.1+2, 21.0.13.1+2, 25.0.5.1+2, 27.0.1.1+4
SAP SapMachine 17, 21, 25, 27	Earlier builds; Java 17 from 17.0.6	17.0.21.1, 21.0.13.1, 25.0.5.1, 27.0.1.1 (CVE-specific fix not confirmed by SAP)
Red Hat Enterprise Linux 8, 9, 10; Red	Java 17 (from 17.0.6), 21, and 25	Pending as of November 18, 2026

© 2026 Critical Start. All rights reserved.

Distribution	Affected Versions	Fixed Release
Hat build of OpenJDK	builds	
Microsoft Build of OpenJDK	Java 17 (from 17.0.6), 21, 25, and 27 builds	Verify vendor advisory
BellSoft Liberica	Java 17 (from 17.0.6), 21, 25, and 27 builds	Verify vendor advisory
Not affected	Standard Java 8 and 11; ARM64 systems; x86-64 without AVX-512 VBMI	Not applicable

Appendix B: Detection and Hunting

- Confirm processor support (Linux; all four flags must appear)

  ```
  grep -m1 '^flags' /proc/cpuinfo | tr ' ' '\n' | grep -E '^(avx512f|avx512bw|avx512vbmi|bmi2)$'
  ```

- Confirm the mitigation flag is active

  ```
  java -XX:-UseBASE64Intrinsics -XX:+PrintFlagsFinal -version | grep UseBASE64Intrinsics
  ```

Hunting indicators

None of these signs proves compromise, and their absence does not prove a host is safe.[2]

- Base64 values of 32 characters or more that end in `=` and whose length is not a multiple of 4, especially in `Authorization` headers
- Bursts of `IllegalArgumentException` errors from `java.util.Base64$Decoder`
- Unexplained `hs_err_pid` JVM crash logs or bursts of service restarts
- Java processes spawning shells or scripting interpreters, or opening new outbound connections, after any of the above

© 2026 Critical Start. All rights reserved.