Java4Shell (CVE-2026-114938): Critical OpenJDK Flaw Enables Remote Code Execution



A critical memory-corruption flaw in OpenJDK's Base64 decoder, CVE-2026-114938, can lead to remote code execution in widely used enterprise applications. Oracle has issued patches; temporary workarounds exist.

- CVE-2026-114938 is a memory-corruption flaw in the hardware-accelerated Base64 decoder of OpenJDK 17.0.6 and later, and in Java distributions built from that code.
- Calif rated it 9.2 (CVSS 4.0); Oracle rated it 8.1 High (CVSS 3.1).
- Researchers achieved unauthenticated RCE against Keycloak, Jira, Confluence, Artifactory, Jenkins and PingFederate in default configurations.
- Exploitation needs an application-specific path to the decoder and x86-64 CPUs with AVX-512 VBMI support; there is no universal exploit.

> - Oracle patched the flaw on November 17; a proof-of-concept of the memory corruption is public, but weaponized exploits are not.

A critical vulnerability nicknamed Java4Shell, tracked as CVE-2026-114938, has been disclosed in OpenJDK. Because many unrelated applications share the same Java runtime, a flaw in a core component can reach a wide range of enterprise systems.

What is affected

The flaw is a memory-corruption bug in the hardware-accelerated Base64 decoding function. Specially crafted Base64 data can overwrite parts of Java's memory, which may expose sensitive information or allow remote code execution. It affects OpenJDK 17.0.6 and later and distributions built from the same code, including Oracle Java SE, Amazon Corretto, Eclipse Temurin, Microsoft Build of OpenJDK, Azul Zulu and Red Hat OpenJDK.

Calif, the research firm that disclosed it, demonstrated unauthenticated code execution against Keycloak, Atlassian Jira and Confluence, JFrog Artifactory, Jenkins and PingFederate in default configurations. Authenticated execution was shown against Apache Kafka Broker, Apache Solr, Elasticsearch, Gerrit and Spinnaker.

Important limits

There is no universal exploit. Attackers need an application-specific way to deliver malicious data to the decoder, and the vulnerable function is only used on x86-64 processors with AVX-512 VBMI support. A proof-of-concept showing the memory corruption is public; the weaponized exploits are not.

Fixes and workarounds

Oracle released patches on November 17 in its November Critical Security Patch Update and urges customers to apply them promptly. Where updating is not immediately possible, Calif suggests launching Java with -XX:-UseBASE64Intrinsics, or -XX:UseAVX=2 on x86-64 systems to disable AVX-512 instructions. These are temporary measures; the runtime should still be updated.

What to do now

1. Inventory every Java runtime and Java-based application (including Jenkins, Jira, Confluence, Keycloak, Artifactory, PingFederate) and identify those on OpenJDK 17.0.6 or later.
2. Apply Oracle's November patches or the equivalent update from your Java distribution vendor, prioritizing internet-facing and identity systems.
3. Where patching must wait, apply -XX:-UseBASE64Intrinsics or -XX:UseAVX=2 as a temporary measure and record it as an exception with an owner and end date.

4. Check which hosts run x86-64 CPUs with AVX-512 VBMI to focus prioritization.
5. Increase monitoring on exposed Java applications for unusual Base64 input and unexpected process activity, and handle patching under your change and vulnerability-management procedures.