What is Java2Shell?

Java2Shell, more widely called Java4Shell, is CVE-2026-114938: a heap-based buffer overflow (CWE-122) in the HotSpot JIT compiler's hardware-accelerated Base64 decoder on x86-64 CPUs with AVX-512 VBMI. Crafted Base64 with padding makes java.util.Base64 decoding write past the end of its output buffer, which researchers turned into remote code execution in several popular Java products. It affects Java 17 from 17.0.6 and later lines up to 17.0.21, 21.0.13, 25.0.5 and 27.0.1. Oracle and OpenJDK fixed it on November 17, 2026; the workaround is -XX:-UseBASE64Intrinsics.

HOW IT WORKS

01 How a Base64 string becomes code execution

- The entry point is ordinary. Java applications Base64-decode untrusted data all the time: tokens and credentials in HTTP headers, SAML and other XML payloads, uploaded files, API fields and messages on a queue. Any path where attacker input reaches `java.util.Base64` decoding is a candidate.
- The trigger is small. The public proof of concept is a run of 64 `A` characters followed by `=`; older Java 17 builds need 128.
- The intrinsic must be active. The bug only exists in the JIT-compiled fast path on x86-64 CPUs with AVX-512 VBMI. Interpreted code, ARM64 servers and CPUs without that instruction set extension take a safe path.
- From overflow to shell. The out-of-bounds write corrupts adjacent heap objects. Turning that into reliable code execution is application-specific (Oracle rates attack complexity high), but the researchers reported working exploits against products including Keycloak, Jira, Confluence, Jenkins and PingFederate without authentication, and against Kafka, Solr, Elasticsearch and Gerrit with it.

It fits the X2Shell pattern exactly: one bug in a runtime that sits under a large share of enterprise software, reached by input that looks completely normal. See image-based heap overflows for how a decoder write becomes code execution in general.

02 Who is exposed

HOW TO DEFEND

- Update the runtime. Install the November 17, 2026 update for your line (the OpenJDK advisory lists 17.0.21, 21.0.13, 25.0.5 and 27.0.1 and earlier as affected) from your JDK vendor, then restart every JVM. Rebuild container images and check products that bundle their own JRE; their vendors ship the fix on their own schedule.
- If you cannot patch yet, disable the vulnerable intrinsic. Add `-XX:-UseBASE64Intrinsics` to the JVM options, or `-XX:UseAVX=2` to stop the JVM using AVX-512 at all. Either blocks the bug; the first is narrower and costs only Base64 speed. Both need a JVM restart and are stopgaps, not fixes.
- Find your JVMs. `java -version` shows the runtime; on a running process, `jcmd <pid> VM.version` and `jcmd <pid> VM.flags` show the version and whether the workaround flag is active.
- Prioritise internet-facing services that decode attacker-supplied Base64: identity providers, SSO and SAML endpoints, API gateways, CI servers and collaboration tools.
- Watch for crashes. Unexpected JVM crashes (`hs_err_pid` files) or heap corruption errors on servers with AVX-512 CPUs are what failed exploitation attempts look like.

SOURCES

You are in scope if all of these hold:

- The Java version. OpenJDK and Java SE 17 from 17.0.6 onward, plus the 21, 25 and 27 lines, up to and including 17.0.21, 21.0.13, 25.0.5 and 27.0.1. That covers the common builds (Oracle Java SE, Eclipse Temurin, Amazon Corretto, Azul Zulu, Red Hat and Microsoft builds of OpenJDK) and Oracle GraalVM for JDK 17 and 21. Java 8 and 11 lines are not listed for this CVE, apart from one Oracle Java 8 performance build.
- The hardware. The JVM runs on an x86-64 CPU with AVX-512 VBMI, which includes many recent Intel Xeon and AMD EPYC server processors and the cloud instances built on them. Check with `grep -o avx512vbmi /proc/cpuinfo | head -1` on Linux.
- The input path. The application decodes Base64 that an attacker can influence.

Bundled runtimes matter here. Many products ship their own JRE inside an installer, appliance or container image, so the Java on the host is not the only one to check. As of November 17, 2026, no in-the-wild exploitation had been reported, but the memory-corruption proof of concept is public.

[1] OpenJDK Vulnerability Advisory: 2026/11/17

[2] NVD: CVE-2026-114938

[3] Oracle Critical Patch Update Advisory, November 2026

[4] CWE-122: Heap-based Buffer Overflow

Find the path to a shell before an attacker does.

securelayer7.net/learn/web-exploitation/what-is-java2shell

Open online