Solr Docker Image CVE Report

Generated 2026-08-15. Images built via `./gradlew :solr:docker:dockerTag` from freshly-pulled branches. Scanned with `docker scout cves`, both raw and with `solr.openvex.json` (downloaded fresh from https://solr.apache.org/solr.openvex.json) applied via `--vex-location solr.openvex.json --vex-author '.*'`, per https://solr.apache.org/security-dependency-cves.html.

Branch	Image	Scan	Total	Critical	High	Medium	Low	
main	apache/solr:11.0.0-SNAPSHOT	No VEX	28	0	5	19	4	
main	apache/solr:11.0.0-SNAPSHOT	With VEX	27	0	5	18	4	
branch_10x	apache/solr:10.1.0-SNAPSHOT	No VEX	32	0	6	22	4	
branch_10x	apache/solr:10.1.0-SNAPSHOT	With VEX	31	0	6	21	4	
branch_9x	apache/solr:9.11.0-SNAPSHOT	No VEX	54	1	17	26	10	
branch_9x	apache/solr:9.11.0-SNAPSHOT	With VEX	49	1	17	22	9	

Released Docker Hub images (solr:10.0.0, solr:9.8.1, solr:9.10.1)

Same process, but pulling the official published images (`docker pull solr:<version>`, Docker Hub `library/solr`) instead of building from source, reusing the same `solr.openvex.json`.

Image	Scan	Total	Critical	High	Medium	Low	Unspecified	Packages affected
solr:10.0.0	No VEX	136	6	52	68	8	2	35

Image	Scan	Total	Critical	High	Medium	Low	Unspecified	Packages affected
solr:10.0.0	With VEX	107	6	42	51	6	2	28
solr:9.10.1	No VEX	226	12	95	103	14	2	56
solr:9.10.1	With VEX	194	12	85	83	12	2	45
solr:9.8.1	No VEX	370	13	108	190	57	2	81
solr:9.8.1	With VEX	340	13	96	174	55	2	73

Why these numbers are so much higher than the from-source SNAPSHOT builds: the base OS layer itself is *not* the difference — `solr:10.0.0` and `apache/solr:10.1.0-SNAPSHOT` are both Ubuntu 24.04, and `docker scout` flags the exact same 9 OS (`deb`) packages as vulnerable in both. The gap is entirely in the Java/application layer, for two concrete, verified reasons:

1. **A `gosu` binary that has since been removed.** All three released images (`solr:10.0.0`, `solr:9.10.1`, `solr:9.8.1`) contain `/usr/sbin/gosu`, a statically-linked Go binary used to drop from root to the `solr` user at container start. It was removed from the Docker build entirely by SOLR-17353 ("Remove gosu binary from the Docker image"), merged 2026-04-16 and cherry-picked to `branch_9x` — **after** all three of these releases were cut (10.0.0: 2026-02-12, 9.10.1: 2026-01-14, 9.8.1: 2025-03-05). A statically-linked Go binary bundles its own copy of the Go standard library, and `docker scout` flags it as `pkg:golang/stdlib@1.22.2` — accounting for **62 of the 136 CVEs** in `solr:10.0.0` alone (4 critical, 24 high, 30 medium, 2 low, 2 unspecified). None of the from-source builds have `gosu` at all (confirmed by diffing `/usr/local/bin` and `which gosu` in each container), since they were built from the current branch tips, after the fix.
2. **Ordinary dependency-version drift since each release was cut.** Released artifacts are frozen at whatever dependency versions were resolved at release time; the live branches have had version bumps land since (some specifically to fix CVEs). E.g. for 10.0.0 vs. today's branch_10x tip: Jetty 12.0.27 → 12.0.34, Jackson 2.20.0 → 2.18.3/2.22.0, Log4j 2.25.3 → 2.26.0, and Netty 4.2.6.Final (multiple CVEs) isn't flagged at all in the branch build. `commons-beanutils`, `kafka-clients`, `opennlp-tools`,

and `zookeeper` only show up as CVE-affected in the released image — either bumped past the vulnerable range, or not resolved into the same version on the branch tip.

So this isn't "released Solr has 5x more vulnerabilities than trunk" — it's "a several-months-old release, frozen in time, is missing since-merged fixes (including one that deletes a whole embedded Go runtime)." A newer patch release cut from current branch tips would look much closer to the SNAPSHOT numbers.

Notes

- **main** and **branch_10x** were built with Gradle 9.6.0 under JDK 25 (the machine's default JDK). **branch_9x** uses Gradle 8.10, which does not support JDK 25, so it was built with `JAVA_HOME` pointed at the installed Temurin 21 JDK instead.
- VEX suppression only knocks out 1-5 CVEs per image here, far fewer than the ~5-exploitable/many-suppressed split described on the security-dependency-cves.html page. That page's `solr.openvex.json` statements are keyed to specific dependency versions (e.g. particular Jetty/xerces/Calcite jar versions); since these are SNAPSHOT builds off active branches, several dependency versions have already moved past what's covered by the currently-published VEX statements, so fewer matches apply.
- branch_9x has the only CRITICAL-severity finding across the three images, plus by far the largest raw CVE count — expected, since it carries the oldest base image (`eclipse-temurin:17-jre-jammy` vs. `eclipse-temurin:25-jre-noble` for main/10x) and dependency set.
- Raw scan logs are saved under `/tmp/solr-cve-scan/` (`build-*.log` for the docker builds, `scout-*-novex.log` / `scout-*-vex.log` for the scans) if you want to drill into individual CVE IDs.