NamiQ

contact@na

SECURITY ADVISORY

Jackson: fix incomplete

CVE-2026-110879 · jackson-databind

Assessment & Action Plan

A critical bypass of the Jackson validation added for CVE-2026-54512. Which of your services are exposed (it depends on polymorphism settings, not on the version alone), which releases fix it, and why 2.19.x has no patch.

PREPARED BY	Hung Le, NamiQ	DATE	2 November 2026
FACTS AS OF	28 October 2026	VERSION	1.1
SEVERITY	**Critical: CVSS 3.1 9.8**	STATUS	Fixed in 2.18.12, 2.21.8, 2.22.4, 3.1.8, 3.2.4

NamiQ © 2026 NamiQ. All rights reserved. This document is the property of NamiQ. It may be shared unaltered, with attribution.
Companion article: namiq.io/insights/jackson-databind-cve-2026-110879

Update, 2 November 2026: this note now ends with a section on checking your own servers and projects with lattice-scan, for IT admins, AI agent skills and CI/CD, and the analysis report is reissued with it. Everything else is as of 28 October 2026; check CISA's catalog for any change since.

In June, jackson-databind fixed a way to smuggle a forbidden class into a deserialiser by nesting it inside a generic type. Teams upgraded. On 20 October the project published a bypass of that fix, CVE-2026-110879, rated 9.8. If you upgraded in June and stopped, the versions you moved to are still affected.

This note is for you if you run Java or Kotlin services that use Jackson, which is most of them. Everything is as of 28 October 2026 and every claim links to its source at the end.

The short version

- CVE-2026-110879 (CWE-502, deserialisation of untrusted data) is an incomplete fix for CVE-2026-54512. Generic type arguments, and the element type of array type IDs, can still carry a class the validator should have refused.
- Patched: 2.18.12, 2.21.8, 2.22.4 on the 2.x line and 3.1.8 and 3.2.4 on the 3.x line (the 3.x artifact is tools.jackson.core). Affected: 2.10.0 to 2.18.11, 2.19.0 to 2.21.7, 2.22.0 to 2.22.3, 3.0.0 to 3.1.7 and 3.2.0 to 3.2.3.
- There is no 2.19.x fix. Services on 2.19.x need to move to 2.21.8 or later.
- The versions that fixed the earlier CVE (2.18.8, 2.21.4 and 3.1.4) are still affected.
- The 9.8 is the worst case. Whether you are in it depends on how your code and your frameworks use polymorphism, and on what classes are available on the classpath.

How the original bug worked, and how this one gets around it

Jackson can deserialise JSON into a type named inside the JSON itself, a feature called polymorphic typing. Because that lets whoever writes the JSON pick the class, Jackson puts a validator in front of it: a PolymorphicTypeValidator that allows or refuses class names.

In CVE-2026-54512 the validator checked the outer class of a generic type, the part before the angle bracket, and then returned the whole parsed type without checking the type arguments. An attacker could put a refused class inside an allowed container. The June fix added validation of the type parameters.

CVE-2026-110879 is the next gap. The advisory says the new check validated each argument against the outer base type rather than the argument's own declared binding. Under the default validator, an unrelated class inside a generic subtype is accepted even though the same class is refused as a direct type ID. Where a rule such as allowIfBaseType approves the outer type, Jackson swaps in a permissive validator and the embedded arguments are not restricted at all. The element type of a plain array type ID is likewise unchecked once the array's base type is approved.

Who is actually exposed

Not every use of Jackson. The advisory lists the configurations where the bypass applies:

- Class-name default typing, in Jackson 2.x and 3.x.
- Annotation-based class-name polymorphism, using the class-name or minimal-class-name type ID on a type, with a default Jackson 3.x JsonMapper.
- Allow rules that approve an outer base type, which then replace the validator with a permissive one.
- Array type IDs whose base type is approved, for example under a rule that allows Serializable.
- Spring Security's legacy Jackson 2 module, and Spring Data Redis 4.0.7's JSON serialiser and the legacy generic Jackson 2 serialiser, which the advisory names.

A service that only maps JSON to plain, fixed types with no polymorphism is not in these cases. Exploitation also needs something useful to instantiate: the advisory's own demonstrations with a standard JDK showed file creation or truncation and outbound requests, and a path to running a process when Spring was on the classpath and the attacker could get the application to load attacker-controlled XML. The more libraries you ship, the more such classes exist.

Things people will get wrong

- "We patched CVE-2026-54512." You did, and you are still affected.
- "We are on 2.19." No 2.19.x fix exists; move to 2.21.8 or later.
- "Our services do not accept untrusted JSON." Data from a cache, a queue, a session store or another service counts if an attacker can write to it. The advisory's workarounds mention preventing attackers from writing serialised cache or session values for that reason.
- "Disable default typing and we are done." Annotation-based polymorphism on 3.x with the default mapper is also listed.

How we would triage it

This is our judgement. The facts come from the advisory.

- Find your Jackson versions first. It is quick, and in a large estate it will find services nobody thought used Jackson.
- Then find the polymorphism. A service with no class-name typing is a lower priority; one with it, taking input from outside your control, is the emergency.
- Upgrade everywhere anyway, because the fix is a version bump within a line and carrying a known critical in a core library is a bad default. But sequence by exposure.

Step 1: what does each service actually resolve

```
# Maven
mvn dependency:tree -Dincludes=com.fasterxml.jackson.core:jackson-databind
mvn dependency:tree -Dincludes=tools.jackson.core:jackson-databind

# Gradle
./gradlew dependencies --configuration runtimeClasspath | grep -iE 'jackson-databind'

# A built artifact
unzip -p app.jar META-INF/maven/com.fasterxml.jackson.core/jackson-databind/pom.properties
```

Resolve from the running build or the deployed artifact, not from the build file. A transitive dependency can pin an older version than the one you declared.

Step 2: find the polymorphism

```
grep -rnE
'activateDefaultTyping|enableDefaultTyping|Id\.CLASS|Id\.MINIMAL_CLASS|allowIfBaseType|GenericJackson2JsonRedisSerializer'
\
  --include=*.java --include=*.kt --include=*.xml --include=*.yml .
```

Check configuration and framework settings too, not just code. Anything that serialises objects into Redis, a message queue or a session store with Jackson deserves a look.

Step 3: upgrade

- 2.18.x to 2.18.12. 2.19.x to 2.21.8 or later. 2.21.x to 2.21.8. 2.22.x to 2.22.4.
- 3.0.x and 3.1.x to 3.1.8. 3.2.x to 3.2.4.
- Use your build tool's dependency management to force a single version across every module so that a transitive dependency cannot pull an older one back in.

If you cannot upgrade this week

These are the advisory's own suggestions. They reduce exposure; they do not close the flaw.

- Disable class-name polymorphism for untrusted data.
- Use narrow subtype rules rather than broad base-type approvals.
- Stop attackers writing serialised cache or session values.
- Remove dependencies that carry classes an attacker could use.

What to look for

- Deserialisation errors that name classes your application does not use, in logs for services that accept JSON from outside.
- Unexpected files being created or truncated by the application account, and unexpected outbound connections from services that normally make none.
- New child processes started by a Java service that does not normally start any.

Check your own servers and projects

The facts above are as of 28 October 2026. To compare your own estate with this CVE, the open-source lattice-scan command line checks the parts it can see. Naming the CVEs needs no key, only the CVE ids leave your machine, and a match is a candidate for a person to confirm, not a finding.

On a server: run it on the server itself, not on the laptop or CI runner that deploys to it. It reads installed packages (dpkg, rpm or apk), snaps, the names of running container images and the kernel version, compares them on the machine, and keeps the inventory there. A distribution can fix a CVE under an unchanged version number, so on Debian-family hosts read the changelog line the report shows.

```
lattice-scan --cve CVE-2026-110879
```

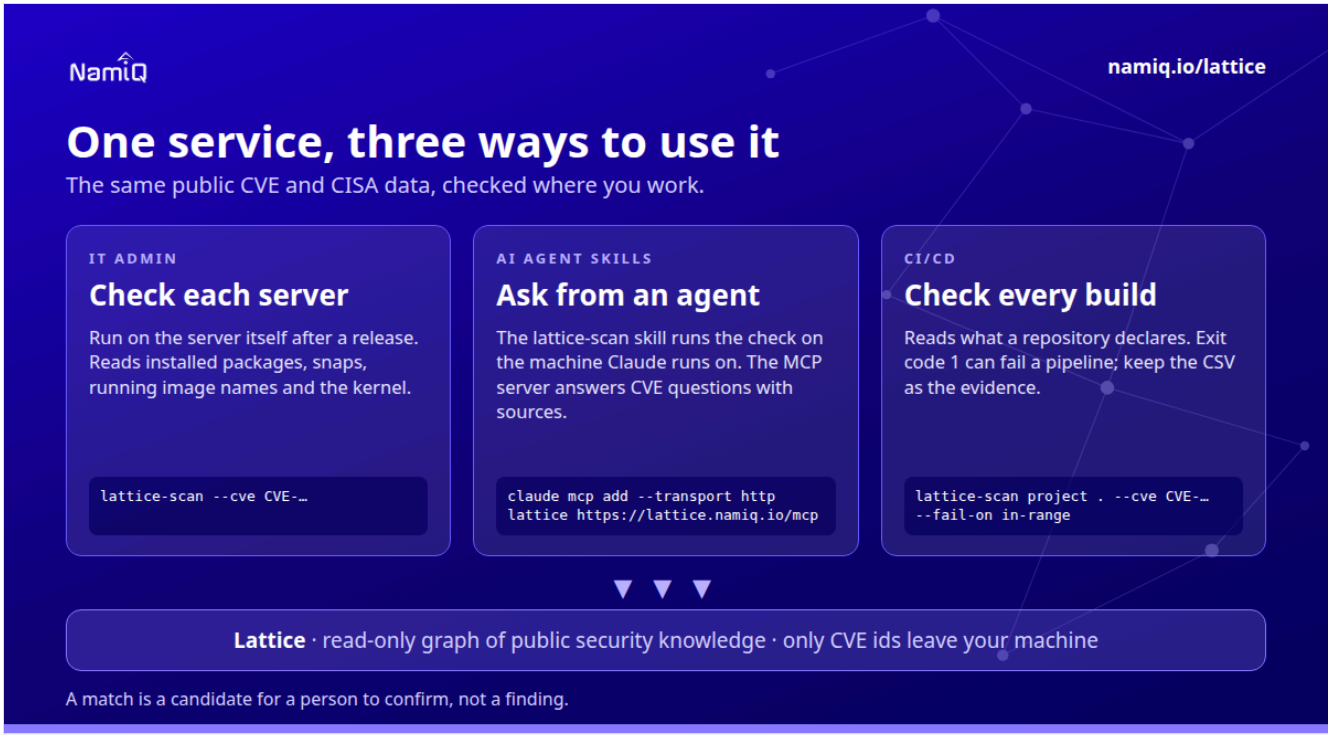
On a repository: the project check reads what it declares, in Python, JavaScript, Java, Rust, C and C++, Make, and Docker and compose files. It installs and runs nothing. A declared range or a tag without a patch number cannot show whether a fix is in, so such a row is undetermined and needs a person.

```
lattice-scan project . --cve CVE-2026-110879
```

A Jackson bundled inside an application's fat JAR, or inside a vendor product, shows in neither check; scan the build that produced it or ask the vendor. The host check sees Jackson only where a distribution package installed it.

Who can use it, and how

For	What to do	Command
IT admin	Run the host check on each server after a release, and again when a new entry lands.	lattice-scan --cve CVE-2026-110879
AI agent skills	The lattice-scan skill runs the check on the machine Claude runs on. The Lattice MCP server answers questions about the CVE with its sources.	claude mcp add --transport http lattice https://lattice.namiq.io/mcp
CI/CD	Check the repository on every pull request. Exit code 1 fails the pipeline; keep the CSV as the evidence.	lattice-scan project . --cve CVE-2026-110879 --fail-on in-range



Where the same check is used. The three ways read the same public data; only CVE ids leave your machine.

Guide: namiq.io/lattice/guides/choose-your-channel. MCP server (io.namiq/lattice in the MCP registry): namiq.io/lattice. Package: pypi.org/project/lattice-scan, or namiq.io/apt.

What we do not know yet

- Whether anyone is exploiting it. The advisory does not say, and we have not read any exploit code.
- Which of your dependencies bundle their own copy of Jackson. Shaded copies do not appear in a dependency tree.
- Whether a further bypass follows. The bug has been fixed twice in five months.

The advisory credits the researchers who found and reported the bypass, and the project maintainer who wrote the fix. If you take one thing from this: a validator is an allow-list you are trusting to be complete. Where you can, do not deserialise attacker-chosen types at all.

Appendix 1. Severity decoded

CVE-2026-110879: score 9.8, `CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H` From the advisory. The earlier CVE-2026-54512 was scored 8.1 with attack complexity High.

Metric	Value	What it means here
Attack vector	Network	Reachable over a network, with no need to be on the same segment.
Attack complexity	Low	No special conditions beyond the flaw itself.
Privileges required	None	No account is needed.
User interaction	None	No one has to do anything.
Scope	Unchanged	Damage stays inside the vulnerable component.
Confidentiality	High	Total loss of confidentiality.
Integrity	High	Total loss of integrity.
Availability	High	Total loss of availability.

Appendix 2. Timeline

Date	Event
16 June 2026	GHSA-j3rv-43j4-c7qm, CVE-2026-54512, published: generic type arguments were not validated. Fixed in 2.18.8, 2.21.4 and 3.1.4.
20 October 2026	GHSA-m8xf-2cr7-77ph, CVE-2026-110879, published: a bypass of that validation.
28 October 2026	The CVE record appears in the public CVE feed.

Appendix 3. Exposure by situation

These priorities are our judgement from the attacker prerequisites in the sources. They are not vendor ratings.

Your situation	Priority	Timeframe
Untrusted JSON is deserialised with class-name polymorphism or default typing	High	Upgrade within days
Spring Security legacy Jackson 2 module, or Spring Data Redis JSON serialisers in use	Check at once	Upgrade within days
Jackson is present but polymorphism is off or limited to a closed list of known types	Lower	Next release cycle
Already on a fixed version	None	Confirm the resolved version

Appendix 4. Definition of done

- Every service's resolved jackson-databind version is known from its dependency tree, not from its build file.
- Each is on a fixed version for its line, with no 2.19.x left.
- Code and configuration that use class-name polymorphism or default typing are listed, and each has a reason to exist.
- Serialised data from untrusted writers is no longer deserialised with class-name polymorphism.
- A dependency check fails builds that resolve a vulnerable version.

Sources

- jackson-databind advisory GHSA-m8xf-2cr7-77ph, CVE-2026-110879: https://github.com/FasterXML/jackson-databind/security/advisories/GHSA-m8xf-2cr7-77ph
- jackson-databind advisory GHSA-j3rv-43j4-c7qm, CVE-2026-54512: https://github.com/FasterXML/jackson-databind/security/advisories/GHSA-j3rv-43j4-c7qm
- CVE-2026-110879 record (HeroDevs CNA): https://www.cve.org/CVERecord?id=CVE-2026-110879