contact@namiq.io · namiq.io

NamiQ

SECURITY ADVISORY

Brotli decoder overflow

CVE-2026-111497 · Google Brotli and Node.js

Assessment & Action Plan

An out-of-bounds write in the Brotli decoder, in CISA's exploited list since 27 October. Which Brotli and Node.js versions fix it, why finding where Brotli lives in your estate is the hard part, and what to do while you look.

PREPARED BY	Hung Le, NamiQ	DATE	2 November 2026
FACTS AS OF	28 October 2026	VERSION	1.1
SEVERITY	**Critical: CVSS 4.0 9.2; in CISA KEV**	STATUS	Fixed in Brotli 1.0.10, 1.1.1, 1.2.1; Node.js 26.12.1, 24.22.1, 22.23.4

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/brotli-decoder-cve-2026-111497

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.

Some of the most important libraries in your estate are ones nobody chose. Brotli is a compression format used to shrink web content, and its decoder is bundled into runtimes, browsers, proxies and language packages. On 21 October a flaw in that decoder was fixed. On 27 October CISA added it to its list of exploited vulnerabilities. This note is about how little you may know about where it lives.

It is for you if you run Node.js services, any software that decodes Brotli, or the build and image pipeline that decides what ships. Everything is as of 28 October 2026 and every claim links to its source at the end.

The short version

- CVE-2026-111497 is a buffer overflow in the Brotli decoder (CWE-787, out-of-bounds write; CWE-252, unchecked return value). A remote unauthenticated attacker who can get a crafted Brotli stream decoded can corrupt memory next to the decoder's state, disclose memory contents, crash the process or potentially execute code.
- CVSS 4.0 9.2, with attack requirements Present. The record credits OpenAI Codex Security (Khai Tran) and Sylvie Mayer.
- Affected: Google Brotli 1.0.3 through 1.2.0. Fixed in 1.0.10, 1.1.1 and 1.2.1. Brotli 1.2.1 was released on 21 October.
- Node.js: the 21 October security release fixes it in the 26.x, 24.x and 22.x lines; the release lists v26.12.1, v24.22.1 and v22.23.4. The release notes are the place to confirm the mapping for your line.
- CISA added it to the Known Exploited Vulnerabilities catalog on 27 October, due 10 November. Ransomware use is not recorded.

What the bug is

The Node.js advisory gives the clearest account. The decoder can be given a malformed stream with an invalid block switch. Instead of reporting that as a permanent format error, the decoder reported it as a temporary need for more input. The streaming decoder then kept appending input bytes to an internal buffer that holds eight bytes. It wrote past the end.

The mistake is a familiar one: an error condition treated as a request to try again. Everything after that is a buffer that was never meant to grow. The Brotli record's description is broader than Node's: it mentions corrupting adjacent decoder state, disclosing memory and potentially running code.

Why this one is hard to scope

Most flaws live in a product you can name. This one lives in a decoder that many products include. Where it hides:

- Node.js itself, through node:zlib. The advisory says an application that decompresses Brotli data an attacker supplies is exposed.
- System libraries on Linux, which many tools link to.
- Language packages for Python, Go and others. The record points to a Go implementation's release (andybalholm/brotli v1.2.7) as part of its references.
- Software that bundles its own copy: browsers, HTTP clients, proxies, application servers. A bundled copy does not update when your operating system does.
- Container images, where the library sits in a layer that was built months ago.

An HTTP response with a Brotli content-encoding is the obvious way compressed data reaches a decoder, but not the only one. Anything that decompresses Brotli files, API payloads or uploads is a path.

Things people will get wrong

- "We do not use Brotli." You may not choose it, but your proxy, your runtime or your client may decode it by default.
- "Our operating system is updated." A bundled or statically linked copy is not touched by that.
- "It is a Node.js issue." It is a Brotli issue that Node.js is one carrier of. The Node.js release is the one with the clearest advisory.
- "CVSS 4.0 9.2 and an attack requirement, so it is hard." Attack requirements Present means a condition must hold, not that the condition is rare. Decompressing data you do not control is a normal thing for a service to do.

The same Node.js release fixes more

If you upgrade Node.js for this, you pick up nine other fixes from the same release, including two TLS hostname-checking bypasses rated High (CVE-2026-105483 and CVE-2026-106069), a dns.resolve truncation issue (CVE-2026-105482), and several Medium and Low issues in the permission model,

UUID generation and zip handling. That is a reason to upgrade the whole runtime and not hunt for a patch to one library.

How we would triage it

This is our judgement from the sources.

- Services that decode Brotli from outside your control come first. If you do not know whether a service does, find out; the answer decides the priority.
- Node.js is the cleanest fix because there is a dated release. Upgrade it everywhere you can.
- For bundled copies, the work is inventory. You cannot patch what you cannot list. Treat this week as the exercise that produces an SBOM you can query.

Step 1: find where Brotli lives

```
# Node.js versions in use
node --version

# Shared library on a host, and what links to it
ldconfig -p | grep -i brotli
find / -name 'libbrotli*' -not -path '/proc/*' 2>/dev/null

# Python and Go dependencies
pip list 2>/dev/null | grep -i brotli
grep -rn 'andybalholm/brotli' --include=go.mod --include=go.sum .

# Container images: look inside, not at the Dockerfile
docker run --rm --entrypoint sh IMAGE -c 'node --version; find / -name "libbrotli*"
2>/dev/null'
```

Add anything your software bill of materials lists that mentions Brotli. If you have none, this is the week to start one.

Step 2: upgrade

- Node.js: 26.12.1, 24.22.1 or 22.23.4 or later for your line. Rebuild images and redeploy.
- Brotli libraries: 1.0.10, 1.1.1 or 1.2.1 or later.
- Bundled copies: check the vendor's release notes and update the product. Where there is no fix, record the owner and the date you will ask again.

If you cannot upgrade this week

- Do not decompress untrusted Brotli data with unpatched code. The Node.js advisory lists no other mitigation, and this is our suggestion, not Node's.

- Limit which services can reach untrusted compressed input, and run them with the least privilege you can.
- Where a proxy or gateway decodes Brotli on your behalf, find out and treat it as in scope.

What to look for

- Crashes in services that decode compressed data, especially clusters of them, shortly after requests with a Brotli content-encoding.
- Unexpected child processes or outbound connections from services that decompress external data.
- Brotli-encoded request bodies arriving at endpoints that do not normally receive them.

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-2026-111497
```

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.

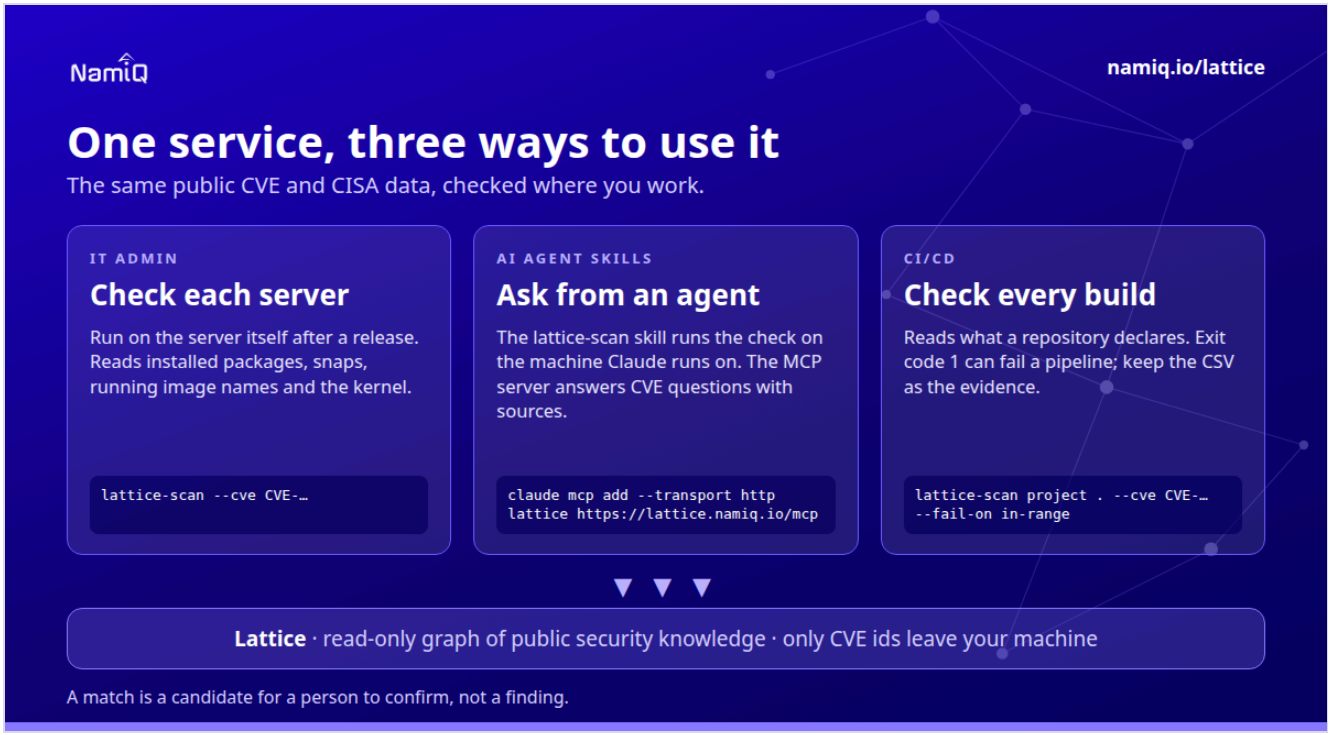
```
lattice-scan project . --cve CVE-2026-111497
```

Software that bundles its own Brotli inside a binary, or inside a vendor product, shows in neither check; ask the vendor. The host check reads the running Node.js version and the Brotli packages a distribution installed.

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-2026-111497

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-2026-111497 --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

- How it is being exploited, or by whom. The listing does not say.
- Whether public exploit code appears. We have not read any.
- Which products bundle an affected Brotli and when each ships a fix.

If you take one thing from this: a library does not need to be on your architecture diagram to be in your attack surface. The question to answer this week is not "are we patched" but "where is it".

Appendix 1. Severity decoded

CVE-2026-111497: score 9.2, `CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N` From the CVE record (CVSS 4.0). Attack requirements are Present: a condition outside the attacker's control must hold.

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.
Attack requirements	Present	Needs a deployment condition the attacker does not control.
Privileges required	None	No account is needed.
User interaction	None	No one has to do anything.
Vulnerable system: confidentiality	High	Total loss of confidentiality.
Vulnerable system: integrity	High	Total loss of integrity.
Vulnerable system: availability	High	Total loss of availability.
Subsequent systems: confidentiality	None	No data exposed.
Subsequent systems: integrity	None	Nothing can be altered.
Subsequent systems: availability	None	No effect on availability.

Appendix 2. Timeline

Date	Event
21 October 2026	Node.js ships security releases that fix the decoder issue; Brotli 1.2.1 is released.
22 October 2026	CVE record published.
27 October 2026	CISA adds CVE-2026-111497 to the Known Exploited Vulnerabilities catalog, due 10 November.

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
Services that decompress Brotli data from outside your control (uploads, proxied responses, third-party APIs)	High	Upgrade within days
Node.js services using node:zlib on untrusted Brotli input	High	Upgrade within days
Browsers, clients and tools that fetch and decode Brotli content from the internet	Medium	Update in the next cycle
Systems that only ever decompress data you produced	Lower	Next cycle
Already on a fixed version	None	Confirm the version

Appendix 4. Definition of done

- Every place that links or bundles Brotli is listed with its version.
- Node.js is on 26.12.1, 24.22.1, 22.23.4 or later on every host and in every image.
- Brotli libraries are 1.0.10, 1.1.1 or 1.2.1 or later wherever you control them.
- Software that bundles its own Brotli has a fixed release from its vendor, or an owner and a date.
- Untrusted Brotli input is not decompressed by unpatched code.
- An SBOM or dependency check fails builds that carry an affected Brotli.

Sources

- Node.js October 2026 security releases: https://nodejs.org/en/blog/vulnerability/october-2026-security-releases
- CVE-2026-111497 record (Google CNA): https://www.cve.org/CVERecord?id=CVE-2026-111497
- Google Brotli release v1.2.1: https://github.com/google/brotli/releases/tag/v1.2.1
- andybalholm/brotli release v1.2.7: https://github.com/andybalholm/brotli/releases/tag/v1.2.7
- CISA Known Exploited Vulnerabilities catalog: https://www.cisa.gov/known-exploited-vulnerabilities-catalog