Hive Pro

Threat Level

Red

Hiveforce Labs

THREAT ADVISORY

VULNERABILITY REPORT

Node2Shell: Node.js Brotli Flaw Turns Compressed Data into Code Execution

Date of Publication	Last Updated Date	Admiralty Code	TA Number
October 23, 2026	October 27, 2026	A1	TA2026314

Summary

First Seen: October 21, 2026
Affected Products: Node.js, Bun, Google Brotli, Python Brotli, brotlicffi, Microsoft .NET
Impact: CVE-2026-111497, dubbed Node2Shell and also tracked as HTTP2Shell and Broccoli2Shell, is a publicly disclosed out-of-bounds write in Google's Brotli decoder, a library bundled with Node.js, Bun, the Python Brotli and brotlicffi packages, and Microsoft .NET. An unauthenticated remote attacker can trigger it with a crafted Brotli-compressed stream to corrupt memory, leak process data, crash the application, or execute arbitrary code. Exposure arises wherever untrusted Brotli data is decompressed, including Express 5 and body-parser 2.x request parsing, responses consumed through fetch(), Python HTTP clients with Brotli support installed, and .NET applications that enable Brotli request or response decompression. Proof-of-concept exploits are publicly available, and code execution has been separately demonstrated against an official Node.js build, which makes immediate patching critical.

CVE

CVE	NAME	AFFECTED PRODUCT	ZERO-DAY	CISA KEV	PATCH
CVE-2026-111497	Node2Shell/ HTTP2Shell/ Broccoli2Shell (Google Brotli Unchecked Return Value Vulnerability)	Node.js, Bun, Google Brotli, Python Brotli, brotlicffi, Microsoft .NET	✗	✓	✓

Vulnerability Details

#1

CVE-2026-111497 is an out-of-bounds write vulnerability (CWE-787) in Google's Brotli decompression library, copies of which are bundled into Node.js and other runtimes. The flaw is triggered by malformed compressed input that drives the decoder to a block-type transition the stream's own structure makes impossible. Instead of rejecting this condition, the vulnerable decoder interprets it as a request for more input and continues processing with an invalid internal state, which can cause it to write beyond an intended memory buffer. The corrected decoder returns a format error for the invalid transition and passes it to the caller, halting decompression before memory is corrupted. This fix had been committed to the upstream Brotli source in February 2026, months before it reached Node.js and other downstream software.

#2

The attack surface covers any path in which Node.js decompresses attacker-controlled Brotli data. Express 5 and body-parser 2.x inflate incoming request bodies carrying the Content-Encoding: br header through any of their json(), raw(), text() or urlencoded() parsers, which inflate Brotli by default, letting any client deliver a payload, while the built-in fetch() API automatically decompresses Brotli responses, exposing services that call attacker-controlled or compromised servers through webhooks or user-supplied URLs. Untrusted input passed directly to node:zlib Brotli functions is equally exposed, whereas a bare Node.js HTTP server does not decode incoming Brotli bodies by itself.

#3

The vulnerability affects Node.js 22.x before 22.23.4, 24.x before 24.22.1, and 26.x before 26.12.1, along with End-of-Life lines such as 25.x, 23.x, and 20.x that will receive no fix. The Bun runtime was also affected before version 1.4.4, and Node.js builds from Linux distribution packages that rely on the system Brotli library depend on the distribution's own Brotli update.

#4

The Node.js project rates the issue High severity and characterizes its impact as remote code execution. Exploit research progressed rapidly after disclosure, with code execution reproduced against the official, unmodified Node.js 24.22.0 Linux arm64 build through both consumed Brotli responses and configured request parsers, although success required multiple attempts. The flaw is publicly disclosed, and proof-of-concept exploits appeared on GitHub from October 22 but reach code execution only on deliberately weakened builds, while no in-the-wild exploitation has been reported.

#5

The flaw extends beyond Node.js because the vulnerable decoder ships in Google Brotli 1.0.3 through 1.2.0 and in software that bundles it. The Python Brotli package before 1.2.1 and brotlicffi before 1.2.1.0 compile the same decoder, exposing Python HTTP clients such as urllib3, requests, httpx and aiohttp whenever Brotli support is installed and untrusted Brotli data is decoded. Microsoft .NET 8 before 8.0.33, .NET 9 before 9.0.22, .NET 10 before 10.0.14 and .NET 11 release candidates before RC3 are affected through ASP.NET Core request decompression, HttpClient automatic response decompression and the System.IO.Compression Brotli APIs, although request and response decompression are disabled by default and .NET Framework is not affected.

Vulnerability

CVE ID	AFFECTED PRODUCTS	AFFECTED CPE	CWE ID
CVE-2026-111497	Node.js (22.x before 22.23.4, 24.x before 24.22.1, 26.x before 26.12.1, and End-of-Life versions), Bun (before 1.4.4), Google Brotli (1.0.3 through 1.2.0), Python Brotli (before 1.2.1), brotlicffi (before 1.2.1.0), Microsoft .NET 8 (before 8.0.33), .NET 9 (before 9.0.22), .NET 10 (before 10.0.14), .NET 11 (before RC3)	cpe:2.3:a:nodejs:node.js:*:*:*:*:*:*:*:* cpe:2.3:a:google:brotli:*:*:*:*:*:*:*:* cpe:2.3:a:microsoft:.net:*:*:*:*:*:*:*:*	CWE-787 CWE-252

Recommendations

Upgrade Node.js Runtimes Immediately: Update all Node.js deployments to 22.23.4, 24.22.1, or 26.12.1, or to a newer patch release within the same line. Prioritize services that accept Brotli-compressed request bodies through Express 5 or body-parser 2.x and those that fetch content from external or user-supplied URLs, as these offer the most direct routes to the vulnerable decoder. The same releases also fix other high-severity issues, including TLS hostname verification bypasses, so upgrading reduces risk beyond this CVE.

Block Untrusted Brotli Request Bodies: Node.js has published no workaround, so where an immediate runtime upgrade is not feasible, reject inbound requests carrying Content-Encoding: br at the reverse proxy, load balancer, WAF, or application middleware. This reduces inbound exposure only and does not protect against Brotli responses that applications fetch from external servers, so treat it strictly as a bridge to the runtime upgrade.

Map Brotli Decompression Paths: Identify every point where untrusted Brotli data enters applications by auditing dependencies for Express 5 and body-parser 2.x, searching code and dependencies for brotliDecompress and createBrotliDecompress calls from node:zlib, and cataloguing outbound fetch() calls to servers outside organizational control, including webhook deliveries and user-supplied URLs.

Update Bundled and System Brotli Components: Upgrade Bun to version 1.4.4 or later, and upgrade the Python Brotli package to 1.2.1 and brotlicffi to 1.2.1.0 in every virtual environment, container image and lock file. For Node.js or Python installed from Linux distribution packages that use the system Brotli library, apply the distribution's Brotli security update in addition to checking the runtime version. Where Python bindings cannot be upgraded immediately, uninstall the Brotli packages so HTTP clients stop advertising and decoding Brotli. Review other software that bundles Brotli, as the same fix shipped in Brotli 1.2.1, 1.1.1, and 1.0.10.

Monitor for Decompression Anomalies: Alert on unexpected Node.js process exits and decoder errors during decompression, and correlate them with the deployed runtime version, application logs, and the origin of the compressed data. These signals help prioritize investigation but are not reliable standalone indicators of exploitation.

Patch .NET and Rebuild Container Images: Update .NET 8, 9 and 10 to 8.0.33, 9.0.22 or 10.0.14 and move .NET 11 previews to Release Candidate 3. Rebuild and republish self-contained .NET applications, which carry their own runtime and are not fixed by a host update. Verify the runtime inside every container image, CI runner and host using node --version or dotnet --list-runtimes, since a package inventory can miss a bundled binary, and move pins such as node:24.22.0 to a fixed release before redeploying. Where .NET patching is delayed, remove the br provider from ASP.NET Core request decompression, exclude DecompressionMethods.Brotli from HttpClient automatic decompression, and stop passing untrusted data to the System.IO.Compression Brotli APIs.

Potential MITRE ATT&CK TTPs

Tactic	Technique	Sub-technique
Initial Access	T1190: Exploit Public-Facing Application	
Execution*	T1059: Command and Scripting Interpreter	
	T1203: Exploitation for Client Execution	
Collection*	T1005: Data from Local System	
Impact	T1499: Endpoint Denial of Service	T1499.004: Application or System Exploitation

*Inferred

Patch Links

https://nodejs.org/en/blog/release/v22.23.4/

https://nodejs.org/en/blog/release/v24.22.1/

https://nodejs.org/en/blog/release/v26.12.1/

https://bun.com/blog/bun-v1.4.4

https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-111497

https://github.com/google/brotli/releases

https://pypi.org/project/Brotli/1.2.1/

https://pypi.org/project/brotlicffi/1.2.1.0/

References

https://www.imperva.com/blog/node2shell-from-brotli-data-to-remote-code-execution/

https://www.aikido.dev/blog/node2shell-nodejs-brotli-rce-cve-2026-111497

https://nodejs.org/en/blog/vulnerability/october-2026-security-releases#nodejs-brotli-decoder-out-of-bounds-write-permits-remote-code-execution-cve-2026-111497---high

https://github.com/oven-sh/bun/pull/46392

https://devblogs.microsoft.com/dotnet/dotnet-october-2026-oob-security-updates/

What Next?

At **Hive Pro**, it is our mission to detect the most likely threats to your organization and to help you prevent them from happening.

Book a Demo of **HivePro**.

REPORT GENERATED ON

October 23, 2026 • 06:35 PM

© 2026 All Rights are Reserved by Hive Pro

More at www.hivepro.com