NODE720

THREAT INTELLIGENCE + BUYER'S BRIEF

The Node.js Vulnerability Landscape 2025–2026

1,142 real CVEs in two years. 28% now hit AI & agentic platforms. Most ship a fix — yet the tools already in your pipeline can only tell you *after* the fact, and most can't block any of it. This brief shows where the risk is, **why your current stack misses it**, and what closes the gap.

1,142

REAL ADVISORIES ANALYSED · 2025–2026 · 0% INVENTED

296 / 846
2025 / 2026

28%
ON AI / AGENTIC PLATFORMS

80%
RUNTIME-BLOCKABLE

2
ENGINES · ONE DETECTOR CORE

Sources: OSV.dev · FIRST.org EPSS · CISA KEV

Your tools see the CVE. They don't stop it.

Read this page if you read nothing else. The data behind it follows — but the conclusion is what matters for a buyer.

Detection is solved. Prevention in the patch-lag window is not — and AI tooling just blew that window wide open.

1 **Volume tripled, and 28% is now AI/agentic.** 1,142 Node.js CVEs in two years; 315 of them on OpenClaw, n8n, Flowise, MCP servers and claude-code — untrusted model output reaching OS, code and network sinks.

2 **SCA tells you a version is vulnerable. A WAF guesses at the HTTP. Neither sees the sink.** The exploit is a data-flow decision inside your process — did attacker input change the *meaning* of an operation? Legacy tools can't answer that, so they alert (or miss) instead of block.

3 **89% ship a fix — but you're exposed until it deploys.** Across a real fleet that's days to months. **80% of the runtime-addressable set is blockable at the sink today**, before a redeploy.

4 **NODE720 is two engines on one detector core.** The same 16 detectors find source→sink flows pre-deploy (`node360-static`) and block them at the sink in production (`node360-rasp`) — by meaning, not signatures. What you find, you also block.

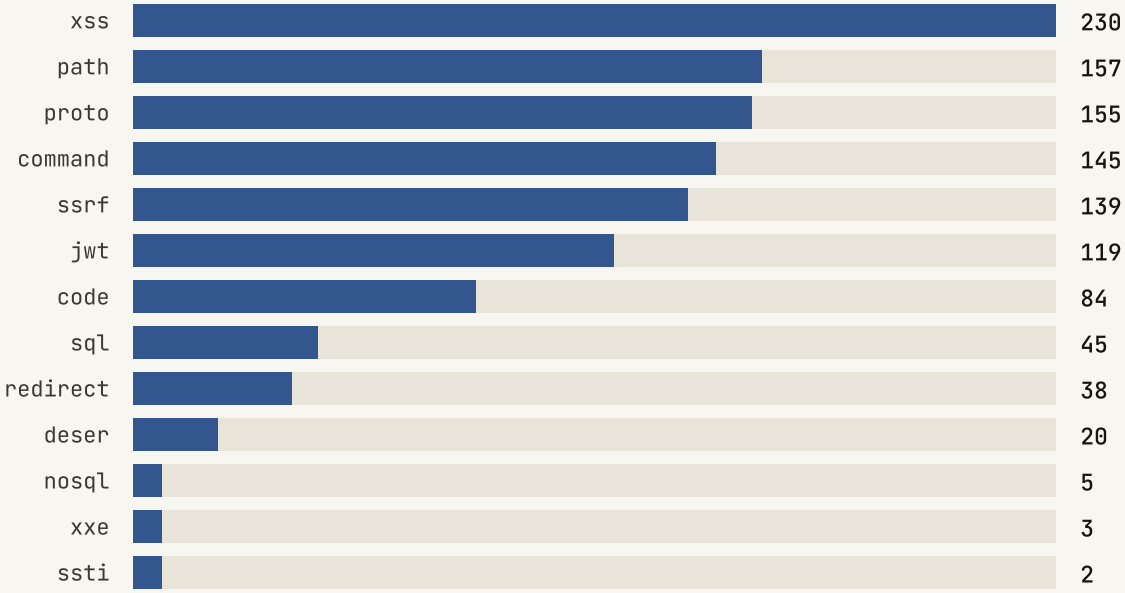
> BOTTOM LINE
>
> You don't have a detection problem — you have a **prevention-and-coverage** problem, and AI tooling made it urgent. The rest of this brief is the evidence.

02 · THE NUMBERS

Two years, 1,142 advisories.

1,142	TOTAL ADVISORIES
846	IN 2026 (VS 296 IN 2025)
28%	ON AI / AGENTIC PLATFORMS
13	DISTINCT VULN CLASSES
89%	SHIP A FIXED VERSION
6	AT EPSS ≥ 0.50
2	IN CISA KEV
80%	RUNTIME-BLOCKABLE

Advisory classes — the whole distribution



Injection dominates: XSS, path traversal, prototype pollution, command and SSRF make up the bulk. These are **sink-side** bugs — the attacker changes what an operation *does* — which is precisely the class a semantic runtime can reason about and a signature engine cannot.

Untrusted model output, real sinks.

The single biggest shift of the period. 315 advisories — 28% of the entire corpus — land on agentic and LLM-orchestration platforms. This surface barely existed in 2024.

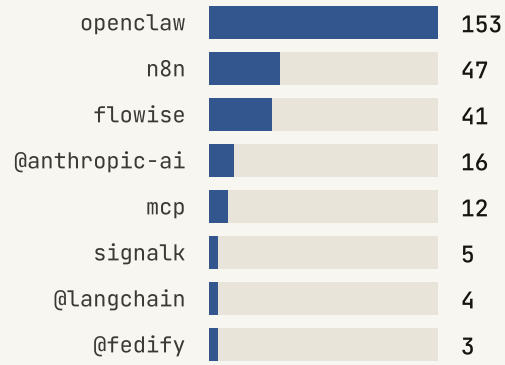
315
AI / AGENTIC CVES

28%
OF ALL ADVISORIES

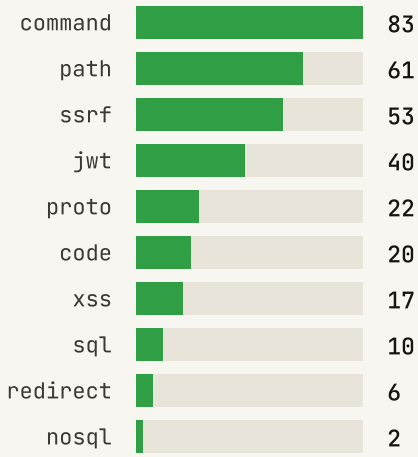
83
ARE COMMAND-INJECTION

114
REACH PATH / SSRF SINKS

Top AI / agentic platforms



Sink classes inside AI CVEs



WHY THIS IS DIFFERENT

In an agent, the **"user" is the model** and the **"input" is a prompt or a tool-call argument**. The dangerous call — spawn a shell, read a file, fetch a URL — looks like ordinary application code. So it sails past every tool that reasons about *packages* or *HTTP requests* instead of the *sink*.

Prompt → tool → sink. Nobody's watching the arrow.

Your SCA, WAF and SAST were built for a world where untrusted data arrives over HTTP and dangerous code is something you wrote. Agentic apps break both assumptions.

The flow legacy tools can't follow: a prompt or retrieved document carries an instruction → the LLM emits a tool-call → your code executes that tool with model-controlled arguments → the argument reaches `child_process`, `fs`, `fetch` or an eval. Every hop is "valid" app behaviour. There is no vulnerable package version to match, no malicious HTTP signature to flag, and no static source the SAST recognises as untrusted.

How NODE720 closes it

- → **Tool-dispatch allowlist** — `guardToolDispatch()` blocks non-allowlisted tools and **taints LLM-controlled arguments** so the sink detectors fire on them (OWASP LLM06 / Agentic-T7).
- → **RAG context is untrusted** — `taintRagContext()` marks retrieved/embedded text as a source, catching poisoned-document injection (LLM08).
- → **Agent memory is untrusted** — `taintMemory()` defends memory-poisoning across turns (Agentic-T1).
- → **Outbound guard** — secret redaction + prompt-leak protection on responses (LLM02 / LLM07), and a supply-chain runtime detector for malicious package behaviour (LLM03).

> THE POINT
>
> None of this is reachable by tools that operate on dependency manifests or HTTP signatures. Defending agents requires reasoning about **data provenance at the sink** — which is the same engine that handles classic injection.

Four tools. One blind spot.

Each tool in a typical pipeline answers a different question — and none of them answers the one the exploit actually turns on: did attacker-controlled input change the meaning of an operation at the sink?

CAPABILITY	SCA / DEPENDABOT	WAF	CLASSIC SAST	NODE720
Catches unknown / zero-day (no version match)	✗	~	~	✓
Reasons about *meaning* at the sink (not regex/signature)	✗	✗	~	✓
Sees in-process, second-order & stored flows	✗	✗	~	✓
Covers non-HTTP vectors (queue, RAG, tool-calls)	✗	✗	✗	✓
Blocks at runtime — not just alerts	✗	~	✗	✓
Closes the patch-lag window (virtual patch)	✗	~	✗	✓
Understands prompt → tool → sink (AI / agentic)	✗	✗	✗	✓
Same detector pre-prod *and* in production	✗	✗	✗	✓

SCA / DEPENDABOT	Matches **known-vulnerable versions**. Blind to zero-days, your own code, and laundered flows; it can only **alert**, never block. The patch-lag window is entirely uncovered.
WAF	Regex/signatures on the **HTTP request**. It never sees the sink, in-process data flow, or non-HTTP vectors (queues, RAG, tool-calls). Encoding bypasses it; legitimate-looking payloads sail through.
CLASSIC SAST	No runtime context → high false positives and **no confirmation of exploitability**; misses dynamic/laundered flows and can't block anything in production.
NODE720	Decides **at the sink, with request provenance**, whether tainted input changed the operation's *meaning* — structural floor + taint — and **blocks before it executes**. Same detector core runs pre-prod and in production.

Two circles. One detector core.

Not another scanner that adds to the alert pile. A prevention platform that finds the flow before deploy and blocks it at the sink in production — from one source of truth.

INNER CIRCLE · PRE-PRODUCTION

node360-static

SAST / taint that reuses the runtime detectors byte-for-byte. Inter-procedural, cross-file, guard-aware; SARIF for CI. Find source→sink before you ship — no runtime trigger.

OUTER CIRCLE · PRODUCTION

node360-rasp

In-process RASP. Wraps the concrete dangerous APIs (child_process, fs, vm, sqlite) and blocks at the sink with request provenance — before the operation executes.

THE DECISION

Meaning, not signatures

A tokenizer differential / containment check asks: did the tainted span add an instruction, escape a value slot, leave the root? That's injection — caught structurally, taint-reduced for low FP.

PATCH-LAG COVER

Virtual patching

Hot-loadable, version-gated, fail-closed rules neutralise known CVEs (e.g. CVE-2025-29927) at runtime — closing the gap between disclosure and redeploy without code changes.

AI & AGENTIC

Provenance for model output

Taint tool-call args, RAG context and agent memory; allowlist tool dispatch; redact secrets on responses. The OWASP LLM Top-10 surface, defended at the sink.

ADOPTION

Zero code change

NODE_OPTIONS bootstrap, or two lines in an Express/Koa app. MIT, Node ≥ 22.15. Honest by design — structural detection is the dependable floor; native/FFI gaps are delegated, not hidden.

What kind, and where it's moving.

Share-of-corpus shift (percentage points) shows the surface relocating toward the AI pattern even as totals climb.

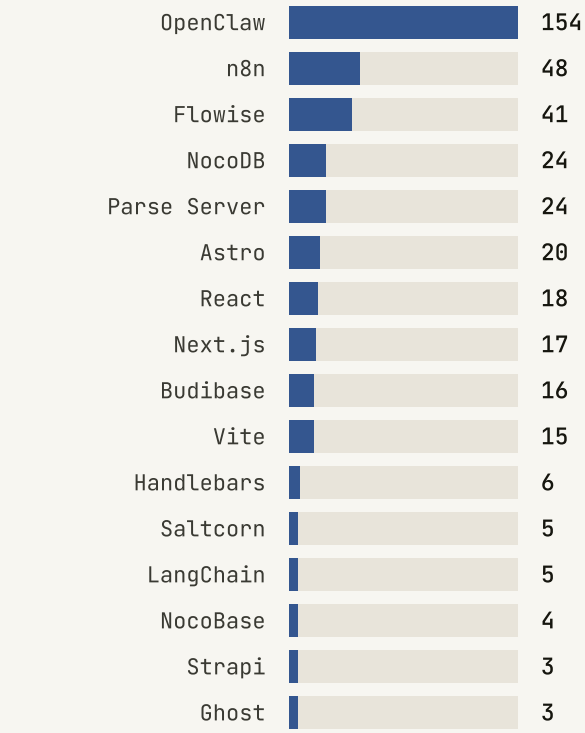
CLASS	2025	2026	2026 VOLUME	SHARE SHIFT
XSS	76	154		▼ 7.5pp
PATH	29	128		▲ 5.3pp
PROTO	55	100		▼ 6.8pp
COMMAND	39	106		▼ 0.6pp
SSRF	22	117		▲ 6.4pp
JWT	25	94		▲ 2.7pp
CODE	17	67		▲ 2.2pp
SQL	7	38		▲ 2.1pp
REDIRECT	12	26		▼ 1.0pp
DESER	12	8		▼ 3.1pp

READ IT

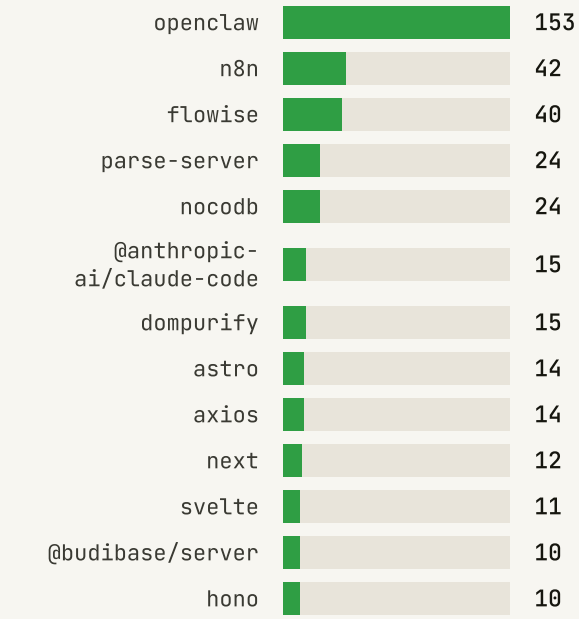
Path traversal and SSRF rise fastest — exactly what you'd expect when agents fetch URLs and read files for untrusted prompts. The composition is bending toward the agentic threat model.

Where the bugs live.

Product / framework families



Top npm packages



THE SIGNAL

The leaderboard is agentic tooling — OpenClaw, n8n, Flowise, claude-code, LangChain. These are exactly the packages an SCA marks "no known vulnerability" between disclosure and patch, and exactly the runtime flows a WAF can't see. NODE720 covers them at the sink the day they're found.

The sharp head of the tail.

Ranked by EPSS with CISA Known-Exploited flags — where remediation and runtime mitigation pay off first.

CVE	PACKAGE	CLASS	EPSS	KEV	SUMMARY
CVE-2025-55182	react-server-dom-webpack	DESER	99.6%	KEV	React Server Components are Vulnerable to RCE
CVE-2025-59528	flowise	CODE	90.2%		Flowise has Remote Code Execution vulnerability
CVE-2025-8943	flowise	COMMAND	70.9%		Flowise OS command remote code execution
CVE-2025-55184	react-server-dom-parcel	DESER	65.6%		Denial of Service Vulnerability in React Server Components
CVE-2025-55183	react-server-dom-parcel	DESER	62.4%		Source Code Exposure Vulnerability in React Server Components
CVE-2025-11953	@react-native-community/cli	COMMAND	61.9%	KEV	@react-native-community/cli has arbitrary OS command injection
CVE-2025-6514	mcp-remote	COMMAND	47.2%		mcp-remote exposed to OS command injection via untrusted MCP server connections
CVE-2025-54782	@nestjs/devtools-integration	COMMAND	46.2%		@nestjs/devtools-integration: CSRF to Sandbox Escape Allows for RCE against JS Developers
CVE-2025-59536	@anthropic-ai/claude-code	CODE	29.3%		Claude Code can execute commands prior to the startup trust dialog
CVE-2024-57041	nodebb	XSS	25.1%		NodeBB Cross-site scripting (XSS) vulnerability
CVE-2025-53107	@cyanheads/git-mcp-server	COMMAND	19.3%		@cyanheads/git-mcp-server vulnerable to command injection in several tools
CVE-2025-67779	react-server-dom-parcel	DESER	18.9%		Denial of Service Vulnerability in React Server Components

Four bugs other tools wave through — blocked.

Representative advisories, with NODE720's honest verdict on each: where it blocks at the sink, and where it's partial and says so.

CVE-2025-55182 DESER EPSS 99.6% · KEV

React Server Components remote code execution

RSC deserialise a crafted Flight payload through the webpack module loader, reviving attacker modules. A CISA-KEV entry.

> node360: Partial — the deserialize detector flags function-reviving markers; framework-internal protocol deser is a documented shared blind spot, delegated to the Permission Model.

CVE-2026-41640 SQL EPSS 1.9%

@nocobase/database SQL injection

An attacker-controlled association name is string-concatenated into a JOIN deep inside recursive eager-loading — several calls from the request.

> node360: BLOCKED where SCA/WAF can't reach: the SQL tokenizer differential sees the tainted span emit keyword/operator tokens at db.query — a laundered flow a regex or version-match never sees.

CVE-2026-1470 CODE EPSS 18.7%

n8n AlaSQL workflow RCE

n8n's Merge node evaluates AlaSQL in SQL mode, so an untrusted workflow expression executes code in the automation engine.

> node360: BLOCKED — code/SQL detectors cover the AlaSQL eval sink; the runtime wrap fires on the tainted expression even though it's 'valid' app code to a WAF.

CVE-2025-11953 COMMAND EPSS 61.9% · KEV

@react-native-community/cli OS command injection

The dev CLI passes untrusted input into a shell command → arbitrary OS execution. CISA-KEV.

> node360: BLOCKED — the command detector analyses shell-argv metacharacters / argv[0] taint at the child_process sink.

What we block — and what we don't.

Measured over the 1,115-CVE matrix. We publish the gaps because an honest floor is the only defensible one.

CLASS	TOTAL	BLOCKS	GAP	% ADDRESSABLE
XSS	227	53	27	66%
PATH	150	94	34	73%
PROTO	149	116	14	89%
COMMAND	143	93	16	85%
SSRF	137	108	11	91%
JWT	114	8	8	50%
CODE	82	28	15	65%
SQL	45	36	9	80%
REDIRECT	38	29	1	97%
DESER	20	6	3	67%
NOSQL	5	3	2	60%
XML	3	1	1	50%
SSTI	2	0	1	0%
TOTAL	**1,115**	**575**	**142**	**80%**

THE HONEST PART

NODE720 is semantic sink-side enforcement + provenance + virtual patching, with structural detection as the dependable floor — **not** a sound dataflow engine. 398 CVEs (missing-auth, business-logic, framework-internal deser) are out of in-process scope by design and delegated to the Node Permission Model / seccomp. We exclude them from the addressable denominator rather than inflate the number.

12 · THE BOTTOM LINE

Find it pre-prod. Block it in prod.

AI tooling turned Node.js security into a runtime-prevention problem your current stack wasn't built for. The fix is not another alert feed.

→ **Scan source pre-deploy** with node360-static — catch the source→sink flow before it ships.

→ **Enforce at the sink in production** with node360-rasp — block by meaning, the same detectors, before execution.

→ **Close the patch-lag window** with virtual patches, and **defend agents** by tainting model output and allowlisting tool dispatch.

720° of Node.js security.

Two circles, one detector core: every line proven clean before deploy, every attack stopped at the sink in production. Zero blind spots — and we tell you exactly where the floor is.

`Get the platform · node720.com · MIT · Node ≥ 22.15`

How this was built.

Fully reproducible, deterministic joins — zero LLM-recalled CVEs.

FEED	SOURCE	USE
OSV.dev	`npm/all.zip (221,085 advisories)`	vuln list, affected package + ranges, CWE, dates
FIRST.org EPSS	`epss_scores-current · v2026.06`	exploitation-probability score
CISA KEV	`known_exploited_vulnerabilities`	confirmed known-exploited flag

SCOPE NOTE

EPSS is a 30-day exploitation estimate, not proof. "Runtime-blockable / addressable" counts advisories whose sink class has a wired detector — not a per-CVE exploit guarantee. The 398-CVE fundamental bucket is excluded from the addressable denominator by design.

Dataset: cve-2025-full-traceable.json · cve-2026-full-traceable.json · coverage-360.json
1,142 advisories (296 in 2025, 846 in 2026) · generated 2026-06-23.

NODE720