INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-06 18:52 UTC

vm2 Sandbox Escape CVE-2026-26956: Public PoC Elevates Exploitation Risk Across npm Ecosystem

CVE VULNERABILITY | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CVE-2026-0134
Type	CVE Vulnerability
CVE ID	CVE-2026-26956, CVE-2026-22709, CVE-2023-30547, CVE-2023-29017, CVE-2022-36067
Severity	CRITICAL
CVSS Base Score	9.5
EPSS Score	0.0009 (25th percentile)
Affected Products	vm2 Node.js sandboxing library (versions up to 3.10.4); Node.js 25 (confirmed 25.6.1); all dependents with 1.3M+ weekly npm downloads
Published	2026-05-06T14:38:38
Discovery Source	Rss

Executive Summary

A critical vulnerability (CVE-2026-26956, CVSS 9.5) in the vm2 Node.js sandboxing library allows attackers to break out of JavaScript isolation and execute arbitrary code on the host system. With over 1.3 million weekly downloads and deep embedding in developer tooling, CI/CD pipelines, and cloud code execution environments, the exposure extends far beyond direct users to any organization with vm2 as a transitive dependency. A public proof-of-concept exploit is available, and vm2's history of recurring critical escapes (CVE-2026-22709, CVE-2023-30547, CVE-2023-29017, CVE-2022-36067) suggests a structural design problem; patch cycles alone may not be a sufficient long-term strategy, and organizations should evaluate migration to actively maintained alternatives.

Technical Analysis

CVE-2026-26956 (CVSS 9.5) is a sandbox escape in vm2 versions up to and including 3.10.4, confirmed affected on Node.js 25.6.1. The vulnerability abuses WebAssembly (Wasm) exception handling mechanisms to bypass JavaScript-level isolation, enabling arbitrary code execution on the underlying host. A related critical escape, CVE-2026-22709, requires separate review. A public proof-of-concept exploit is circulating, materially lowering the barrier to exploitation. Applicable CWEs: CWE-693 (Protection Mechanism Failure), CWE-913

(Improper Control of Dynamically-Managed Code Resources), CWE-1321 (Improperly Controlled Modification of Object Prototype Attributes). MITRE ATT&CK mappings: T1190 (Exploit Public-Facing Application), T1203 (Exploitation for Client Execution), T1195.001 (Compromise Software Supply Chain), T1059.007 (JavaScript). This is the fourth critical sandbox escape pattern identified in vm2 (prior: CVE-2023-30547, CVE-2023-29017, CVE-2022-36067), indicating a structural isolation weakness rather than isolated implementation bugs. Patch 3.10.5 addresses CVE-2026-26956; CVE-2026-22709 remediation should be confirmed separately against the NVD advisory at https://nvd.nist.gov/vuln/detail/CVE-2026-22709. EPSS score is currently low (0.092%, 25th percentile), though public PoC availability has historically been associated with increased exploitation attempts in the weeks following disclosure. NVD is the authoritative reference.

Action Checklist

1. Step 1: Containment. Inventory all direct and transitive dependencies for vm2 across your Node.js projects, CI/CD pipelines, and cloud code execution environments. Run 'npm ls vm2' or equivalent in each repository. Isolate or disable any internet-facing services that invoke vm2 for untrusted code execution until patching is confirmed. For internal-only vm2 usage, prioritize patching within 24 hours but containment can be deprioritized if network access controls prevent external exploitation.
2. Step 2: Detection. Search application logs and npm audit outputs for vm2 versions 3.10.4 and below. Run 'npm audit' across affected codebases and review output for vm2 findings. Monitor host-level process trees for unexpected child processes spawned from Node.js sandbox contexts. Check SIEM for anomalous command execution events originating from Node.js processes (e.g., unexpected shell invocations). No confirmed IOC hashes or network indicators are currently available from T1/T2 sources.
3. Step 3: Eradication. Upgrade vm2 to version 3.10.5 to address CVE-2026-26956. Confirm CVE-2026-22709 remediation status against the NVD advisory (https://nvd.nist.gov/vuln/detail/CVE-2026-22709). For environments where vm2 is a transitive dependency, update the dependency chain and force resolution to 3.10.5 or higher using package overrides (npm 'overrides' field in package.json). Evaluate migration to isolated-vm (https://github.com/laverdet/isolated-vm, actively maintained as of 2026-03-04) or similar alternatives for any use case involving execution of untrusted code. Consult your security team before migration to ensure the alternative meets your isolation and performance requirements.
4. Step 4: Recovery. After patching, re-run 'npm audit' to confirm no remaining vm2 findings. Validate that CI/CD pipeline builds reference the patched version. Monitor host-level process execution and Node.js application logs for 48-72 hours post-remediation for any signs of prior exploitation. Confirm runtime behavior of applications that depend on vm2 sandbox functionality has not changed unexpectedly.
5. Step 5: Post-Incident. Document vm2 dependency instances discovered during this exercise and add them to your software bill of materials (SBOM). Establish or review your third-party dependency review process to flag libraries with recurring critical CVE patterns (vm2 has four critical sandbox escapes since 2022). Evaluate adoption of a dependency management policy that triggers migration review when a library accumulates two or more critical escapes within a 24-month window. Map this gap to NIST SP 800-53 SA-15 (Development Process, Standards, and Tools) and SR-3 (Supply Chain Controls and Processes).

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and legal/compliance immediately if Sysmon Event ID 1 or auditd records confirm a shell process spawned as a child of a Node.js vm2 context (indicating successful sandbox escape), or if any affected service processes untrusted code submitted by external users and is subject to PCI-DSS, SOC 2, or HIPAA obligations, as exploitation would constitute a breach of the isolation boundary with potential data exposure requiring formal incident declaration.
Recovery Notes	After patching to vm2 3.10.5, verify the fix integrity by confirming the installed package checksum matches the npm registry value ('npm view vm2@3.10.5 dist.integrity') to rule out supply-chain substitution. Monitor host-level process trees and Node.js application logs for 72 hours post-remediation, specifically watching for unexpected child processes of node (shells, curl, wget, Python) that would indicate a persistence mechanism installed by an attacker who exploited CVE-2026-26956 before the patch was applied. Any service that executed untrusted JavaScript via vm2 and was internet-facing during the window between PoC publication and patch deployment should be treated as potentially compromised and subjected to full filesystem integrity verification using AIDE or Tripwire before being returned to full production status.
Forensic Artifacts	Sysmon Event ID 1 (Process Creation) logs: filter for events where ParentImage is node.exe or /usr/bin/node and Image is any shell or network utility (cmd.exe, bash, sh, curl, wget, python) — this is the direct host-level artifact of a successful vm2 CVE-2026-26956 sandbox escape achieving OS command execution (MITRE ATT&CK T1059) | Linux auditd execve syscall records (/var/log/audit/audit.log): filter by ppid matching the Node.js process PID — a sandbox escape from vm2's JavaScript isolation would surface here as an unexpected execve call originating from within the V8 runtime context, with uid/gid matching the Node.js service account | Node.js application access logs (Express.js stdout, PM2 logs at ~/.pm2/logs/, or systemd journal): search for HTTP POST requests to any endpoint that accepts JavaScript or code payloads in the timeframe surrounding the PoC publication date, including anomalous payload sizes or base64-encoded content in request bodies that could represent exploit delivery | npm-shrinkwrap.json and package-lock.json files from production deployments: these pin the exact vm2 version resolved at deploy time and serve as the authoritative record of which version was running in production — critical for determining exposure window between PoC availability and patch deployment | Filesystem modification timestamps under the application working directory and /tmp: a successful CVE-2026-26956 escape with follow-on persistence would likely write files outside the Node.js process's expected working directories — run 'find /var /tmp /home /opt -newer /etc/passwd -type f -ls' to identify files created or modified after the service started, which may indicate post-exploitation file drops (MITRE ATT&CK T1105 Ingress Tool Transfer)

Per-Action IR Details

Step 1: Containment — Inventory all direct and transitive dependencies for vm2 across your Node.js projects, CI/CD pipelines, and cloud code execution environments. Run 'npm ls vm2' or equivalent in each repository. Isolate or disable any internet-facing services that invoke vm2 for untrusted code execution until patching is confirmed.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST IR-4 (Incident Handling), NIST CM-7 (Least Functionality), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: Run 'npm ls vm2 --all 2>/dev/null' and 'find . -name package-lock.json | xargs grep -l vm2' recursively across all repos from a CI runner or developer workstation. For cloud functions (AWS Lambda, GCP Cloud Functions), query deployment manifests: 'aws lambda list-functions | xargs -I{} aws lambda get-function-configuration --function-name {} | grep -i vm2'. Block inbound traffic to any HTTP endpoint that accepts and executes arbitrary JavaScript (e.g., code sandbox APIs, REPL services) using host-based firewall rules: 'ufw deny in on eth0 to any port 3000' (adjust port). Use osquery to enumerate Node.js processes currently running: 'SELECT pid, name, cmdline FROM processes WHERE name LIKE "%node%";'

Evidence: Before isolating services, capture: (1) running Node.js process list with full command lines ('ps auxf | grep node' on Linux; 'Get-WmiObject Win32_Process | Where-Object {$_.Name -like "*node*"} | Select-Object ProcessId, CommandLine' on Windows) to establish pre-containment baseline; (2) active network connections from Node.js PIDs ('ss -tulnp | grep node' or 'netstat -anp | grep node') to identify any established outbound connections that may indicate prior exploitation; (3) snapshot of all package-lock.json and npm-shrinkwrap.json files from production deployments before any changes are made — these are your pre-remediation dependency chain evidence.

Step 2: Detection — Search application logs and npm audit outputs for vm2 versions 3.10.4 and below. Run 'npm audit' across affected codebases and review output for vm2 findings. Monitor host-level process trees for unexpected child processes spawned from Node.js sandbox contexts. Check SIEM for anomalous command execution events originating from Node.js processes (e.g., unexpected shell invocations). No confirmed IOC hashes or network indicators are currently available from T1/T2 sources.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Deploy Sysmon with a configuration that captures process creation (Event ID 1) with ParentImage filtering: alert on any process where ParentImage contains 'node.exe' or '/bin/node' and Image contains 'cmd.exe', 'powershell.exe', 'sh', 'bash', 'python', or 'curl' — these are the shell escape patterns a CVE-2026-26956 PoC would trigger. Use this Sigma-compatible detection logic: 'ParentImage|endswith: node.exe AND Image|endswith|any: [cmd.exe, powershell.exe, sh, bash, wget, curl]'. For Linux hosts, enable auditd with: 'auditctl -a always,exit -F arch=b64 -S execve -F ppid=$(pgrep node) -k vm2_escape'. Run 'npm audit --json 2>/dev/null | python3 -c "import sys,json; [print(v[\"name\"],v[\"severity\"]) for v in json.load(sys.stdin)[\"vulnerabilities\"].values() if v[\"name\"]==\"vm2\"]"' to parse audit output programmatically.

Evidence: Capture before analysis: (1) Sysmon Event ID 1 (Process Creation) logs filtered for node.exe or node as parent process — specifically look for child processes that are shells, interpreters, or network tools spawned after the vm2 sandbox context was invoked, consistent with MITRE ATT&CK T1059 (Command and Scripting Interpreter) and T1055 (Process Injection); (2) application-level logs from the service invoking vm2 (e.g., Express.js access logs, typically at /var/log/app/ or stdout captured by PM2/systemd — look for HTTP POST requests submitting JavaScript payloads to code-execution endpoints in the hours before detection); (3) Linux auditd logs (/var/log/audit/audit.log) for execve syscalls with ppid matching the Node.js process — a sandbox escape for CVE-2026-26956 would manifest as an unexpected execve from within the vm2 V8 context; (4) npm audit JSON output ('npm audit --json > npm_audit_evidence_$(date +%Y%m%d).json') as documented vulnerability evidence.

Step 3: Eradication — Upgrade vm2 to version 3.10.5 to address CVE-2026-26956. Confirm CVE-2026-22709 remediation status against the NVD advisory (https://nvd.nist.gov/vuln/detail/CVE-2026-22709). For environments where vm2 is a transitive dependency, update the dependency chain and force resolution to 3.10.5 or higher using package overrides (npm 'overrides' field in package.json). Evaluate migration to isolated-vm or a comparable actively maintained alternative for any use case involving execution of untrusted code.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-7 (Least Functionality), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: For transitive dependency resolution without an enterprise dependency management platform, add the following to package.json: '{"overrides": {"vm2": ">=3.10.5"}}' then run 'npm install' and verify with 'npm ls vm2' — no version below 3.10.5 should appear. For Yarn workspaces use 'resolutions' field equivalently. Verify the installed vm2 package integrity after upgrade: 'cat node_modules/vm2/package.json | grep version' and cross-reference against the published npm registry checksum ('npm view vm2@3.10.5 dist.integrity'). For environments evaluating isolated-vm as a replacement, note it uses V8 Isolates via native bindings rather than vm2's pure-JS sandbox, providing a fundamentally different (and more robust) isolation boundary — test in staging before production cutover.

Evidence: Before executing upgrades, preserve: (1) a full copy of the current node_modules/vm2/ directory ('tar czf vm2_preupgrade_$(date +%Y%m%d).tar.gz node_modules/vm2/') as forensic baseline in case post-incident analysis requires examination of the exploitable code path in vm2's lib/main.js sandbox escape surface; (2) git diff or snapshot of package.json and package-lock.json before and after the override addition, timestamped, for change management evidence; (3) for any system where prior exploitation is suspected, take a memory dump or at minimum capture /proc/[node-pid]/maps and /proc/[node-pid]/environ before terminating the process, as a successful CVE-2026-26956 escape may leave evidence of injected code in the Node.js heap.

Step 4: Recovery — After patching, re-run 'npm audit' to confirm no remaining vm2 findings. Validate that CI/CD pipeline builds reference the patched version. Monitor host-level process execution and Node.js application logs for 48-72 hours post-remediation for any signs of prior exploitation. Confirm runtime behavior of applications that depend on vm2 sandbox functionality has not changed unexpectedly.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST IR-5 (Incident Monitoring), NIST SI-2 (Flaw Remediation), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Automate post-patch validation in CI/CD by adding a pipeline gate: 'npm audit --audit-level=critical && npm ls vm2 | grep -v "3.10.[5-9]\|3.1[1-9]\|[4-9]\.[0-9]" && echo FAIL || echo PASS'. For 48-72 hour post-remediation monitoring without SIEM, configure Sysmon to log to a dedicated channel and set up a cron job that queries process creation events every 15 minutes: 'Get-WinEvent -LogName "Microsoft-Windows-Sysmon/Operational" -FilterXPath "*[EventData[Data[@Name=\"ParentImage\"] and contains(.,\"node\")]]" | Where-Object {$_.TimeCreated -gt (Get-Date).AddMinutes(-15)}'. On Linux, tail auditd logs with filtering: 'ausearch -k vm2_escape --start recent | aureport -x --summary'.

Evidence: Capture during recovery validation: (1) 'npm audit --json' output post-patch as a timestamped clean-bill artifact for audit trail purposes (NIST AU-11 Audit Record Retention); (2) CI/CD pipeline build logs showing the vm2 3.10.5 dependency resolved correctly across all affected pipeline stages — screenshot or export from GitHub Actions/Jenkins/GitLab CI as applicable; (3) continued collection of Sysmon Event ID 1 and auditd execve records for the 48-72 hour window to detect any persistence mechanism (e.g., cron job, systemd timer, or Node.js startup script) installed by an attacker who exploited the sandbox escape prior to patching, consistent with MITRE ATT&CK T1053 (Scheduled Task/Job).

Step 5: Post-Incident — Document vm2 dependency instances discovered during this exercise and add them to your software bill of materials (SBOM). Establish or review your third-party dependency review process to flag libraries with recurring critical CVE patterns (vm2 has four critical sandbox escapes since 2022). Evaluate adoption of a dependency management policy that triggers migration review when a library accumulates two or more critical escapes within a 24-month window. Map this gap to NIST SP 800-53 SA-15 (Development Process, Standards, and Tools) and SR-3 (Supply Chain Controls and Processes).

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST SA-15 (Development Process, Standards, and Tools), NIST SR-3 (Supply Chain Controls and Processes), NIST IR-8 (Incident Response Plan), NIST SI-2 (Flaw Remediation), CIS 2.1 (Establish and Maintain a

Software Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Generate an SBOM using free tooling: 'npx @cyclonedx/cyclonedx-npm --output-format json --output-file sbom_$(date +%Y%m%d).json' for CycloneDX-format SBOM covering the entire Node.js dependency tree. Store in version control alongside the application. To automate future vm2-pattern detection (libraries with recurrent critical CVEs), add a GitHub Actions workflow using 'npm audit --audit-level=critical' as a required check on all PRs. For the migration review trigger policy, create a lightweight decision record template (e.g., a Markdown ADR) that fires when OSV.dev or NVD shows two or more CVSS ≥9.0 findings against the same package within 24 months — vm2's history (CVE-2022-36067, CVE-2023-29017, CVE-2023-30547, CVE-2026-22709, CVE-2026-26956) is the reference case for calibrating this threshold.

Evidence: Preserve as post-incident documentation: (1) the complete output of 'npm ls vm2 --all' from each affected repository at discovery time, retained as evidence of blast radius scope for lessons-learned reporting; (2) the SBOM generated post-remediation as the authoritative baseline for future supply chain comparisons; (3) a timeline log correlating vm2 CVE disclosure dates (CVE-2022-36067 through CVE-2026-26956) against your organization's dependency update history — this establishes whether the library's recurrent critical escape pattern should have triggered earlier migration review and supports process improvement under NIST 800-61r3 §4 (lessons learned).

Detection Guidance

Run 'npm ls vm2' and 'npm audit' in all Node.js project directories to surface direct and transitive vm2 dependencies at version 3.10.4 or below. In SIEM or EDR telemetry, query for child processes spawned by node.exe or node processes with unusual parent-child relationships (e.g., node spawning cmd.exe, /bin/sh, or powershell.exe). Alert on process creation events where the initiating process is a Node.js runtime and the child process is a system shell or interpreter not expected in normal application operation. In cloud environments running hosted code execution (e.g., serverless functions, code sandboxes), review execution logs for anomalous system calls or unexpected network outbound connections from sandbox runtimes. No confirmed public IOCs (IPs, domains, file hashes) have been identified from T1 or T2 sources as of this item's configuration date (2026-03-04). Monitor NVD (https://nvd.nist.gov/vuln/detail/CVE-2026-26956) and CISA for updates.

Framework Mappings

MITRE-ATTACK

- **T1190** — Exploit Public-Facing Application
- **T1203** — Exploitation for Client Execution
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1059.007** — JavaScript

NIST-800-53R5

- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring

- **SR-2** — Supply Chain Risk Management Plan

CIS-V8

- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management
- **15.1** — Establish and Maintain an Inventory of Service Providers

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain
- **A.5.23** — Information security for use of cloud services

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program

SOC2-TSC

- **CC9.2** — Manages risks associated with vendors and business partners

HIPAA-SECURITY

- **164.312(e)(1)** — Transmission Security

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1190	Exploit Public-Facing Application	Initial-Access
T1203	Exploitation for Client Execution	Execution
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1059.007	JavaScript	Execution

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/critical-vm2-sandbox...	T3
CVE-2026-22709 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-22709	T1
Critical vm2 Node.js Flaw Allows Sandbox Escape and Arbitrary ...	https://thehackernews.com/2026/01/critical-vm2-nodejs-flaw-allows-s...	T3

Source	URL	Tier
CVE-2026-26956 - Exploits & Severity - Feedly	https://feedly.com/cve/CVE-2026-26956	T3
Critical Sandbox Escape in vm2 Enables RCE | Blog - Endor Labs	https://www.endorlabs.com/learn/cve-2026-22709-critical-sandbox-esc...	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-26956, CVE-2026-22709, CV...	T1

Generated 2026-05-06 18:52 UTC by TJS Security Command Center