INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-22 13:51 UTC

vitejs vite - vitejs vite Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

CVE VULNERABILITY | HIGH | CVSS 7.5 | CISA KEV

SCC Item ID	SCC-CVE-2026-0211
Type	CVE Vulnerability
CVE ID	CVE-2026-39365
Severity	HIGH
CVSS Base Score	7.5
EPSS Score	0.0169 (82th percentile)
KEV Status	Yes — CISA Known Exploited Vulnerability
Affected Products	vitejs/vite 6.0.0-<6.4.2, 7.x-<7.3.2, 8.x-<8.0.5
Published	2026-05-22T00:00:00Z
Discovery Source	Vulncheck Kev

Executive Summary

A path traversal vulnerability in Vite's development server allows attackers to read arbitrary source map files outside the project root by embedding directory-traversal sequences in URLs. Organizations using Vite versions 6.0.0-6.4.1, 7.x-7.3.1, or 8.x-8.0.4 in any environment reachable by untrusted parties are directly at risk. CISA has confirmed active exploitation; unpatched systems expose internal source code structure and logic to reconnaissance, accelerating follow-on attacks.

Technical Analysis

CVE-2026-39365 (CWE-22: Path Traversal) affects Vite's development server across three version branches: 6.0.0-<6.4.2, 7.x-<7.3.2, and 8.x-<8.0.5. The server resolves requests for .map files associated with optimized dependencies and passes the resolved path directly to readFile without sanitizing or rejecting '../' sequences embedded in the URL. This bypass circumvents the server.fs.strict allow list, permitting unauthenticated read access to arbitrary .map (source map JSON) files anywhere on the host filesystem the process can reach. MITRE ATT&CK maps to T1083 (File and Directory Discovery) and T1552.001 (Credentials in Files). CVSS base score: 7.5 (High). EPSS: 0.01694 (82nd percentile). CISA KEV confirmed, indicating active exploitation in the wild. Fixed releases: 6.4.2, 7.3.2, 8.0.5.

Action Checklist

1. Step 1: Containment. Immediately identify all environments running vitejs/vite 6.0.0-6.4.1, 7.x-7.3.1, or 8.x-8.0.4. Block external network access to Vite dev server ports (default 5173) at the perimeter firewall or WAF if a patch cannot be applied immediately. Vite dev servers should never be exposed to untrusted networks; enforce immediately as an emergency control per NIST AC-4 (Information Flow Enforcement) and CIS 4.4 (Implement and Manage a Firewall on Servers).
2. Step 2: Detection. Query web server and application logs for URL patterns containing encoded or literal '../' sequences targeting paths that include '/@fs/', '/node_modules/.vite/', or '.map' extensions. Look for HTTP 200 responses to requests with path traversal patterns. Review SIEM for T1083 (File and Directory Discovery) alerts correlated with Vite process activity. Check for access to sensitive files (e.g., .env, private keys, config files) via source map requests. Apply NIST AU-6 (Audit Record Review, Analysis, and Reporting) and AU-3 (Content of Audit Records) to ensure log completeness.
3. Step 3: Eradication. Upgrade vitejs/vite to 6.4.2, 7.3.2, or 8.0.5 per the patched release for your version branch. Verify the installed version via 'npm list vite' or 'yarn list vite'. If upgrade is not immediately possible, restrict the Vite dev server to localhost (server.host: 'localhost') and enforce server.fs.strict: true in vite.config as a temporary mitigation; note this does not fully remediate the vulnerability. Reference CIS 7.3 (Perform Automated Operating System Patch Management) and CIS 7.4 (Perform Automated Application Patch Management).
4. Step 4: Recovery. After patching, verify the installed Vite version matches a fixed release. Re-enable any temporarily blocked services. Rotate secrets or credentials present in any .env files, configuration files, or source map files that were accessible to the vulnerable server, treating them as potentially compromised. Monitor for anomalous authentication attempts or access patterns using rotated credentials per NIST AC-2 (Account Management) and D3-CRO (Credential Rotation).
5. Step 5: Post-Incident. Review your development pipeline for any Vite dev server instances exposed outside localhost, and enforce a policy prohibiting dev server exposure to untrusted networks. Assess whether source map files in affected environments exposed proprietary logic, hardcoded secrets, or internal path structures. Map this gap to NIST SI-4 (System Monitoring) and CIS 8.2 (Collect Audit Logs) to improve detection of path traversal attempts going forward. Add a check for Vite version compliance to your software inventory process per CIS 2.1 (Establish and Maintain a Software Inventory) and CIS 2.2 (Ensure Authorized Software is Currently Supported).

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to incident commander and legal/compliance if forensic log review reveals HTTP 200 responses to path traversal requests returning .env files, private keys, or database credentials — this confirms data exfiltration and may trigger breach notification obligations depending on the sensitivity of exposed secrets and applicable regulatory framework (e.g., GDPR, state breach notification laws); also escalate immediately if CISA's confirmed active exploitation status intersects with evidence of inbound traversal attempts from external IPs in your access logs.

Recovery Notes	After applying the Vite patch (6.4.2, 7.3.2, or 8.0.5), verify remediation by running a controlled traversal test from an isolated host: 'curl -v "http://:5173/@fs/../../../etc/passwd"' — a 403 or connection refusal confirms the fix is active. All secrets accessible to the vulnerable Vite instance must be treated as compromised and rotated before the system returns to service, regardless of whether traversal was confirmed in logs, because the exposure window may predate log retention. Monitor authentication and API usage logs for rotated credentials for a minimum of 30 days to detect delayed use of any exfiltrated secrets.
Forensic Artifacts	Vite dev server stdout/stderr logs or reverse proxy (nginx/caddy) access logs: primary artifact for confirming exploitation — look for HTTP 200 responses to requests containing '/@fs/', '%2F..%2F', '%252F', or '.map' with source IPs outside RFC1918 ranges and non-trivial response Content-Length values indicating file content was returned. | vite.config.js or vite.config.ts from each affected project: documents whether server.host was set to '0.0.0.0' or a routable address and whether server.fs.strict was false (the default), establishing the exploitable configuration and exposure scope specific to this CVE-2026-39365 vulnerability mechanism. | Project .env and .env.* files combined with source map files (.js.map, .ts.map) in the .vite/deps cache and dist directories: these are the direct targets of the path traversal attack — their contents reveal what secrets, API keys, internal hostnames, and code logic were exposed to an attacker who successfully exploited the /@fs/ traversal endpoint. | package.json and package-lock.json (or yarn.lock) from all Node.js projects on affected hosts: establishes which Vite version was installed including transitive dependencies, defines the precise exposure window by correlating install timestamps with log activity, and serves as evidentiary documentation of the vulnerable version for post-incident reporting. | Network flow records or firewall logs for TCP port 5173 (and 5174–5180) covering the full exposure window: identifies source IPs that connected to the Vite dev server, frequency and volume of requests, and whether access patterns suggest automated scanning (consistent with MITRE T1595 Active Scanning) or targeted traversal attempts characteristic of reconnaissance preceding a follow-on attack.

Per-Action IR Details

Step 1: Containment — Immediately identify all environments running vitejs/vite 6.0.0–6.4.1, 7.x–7.3.1, or 8.x–8.0.4. Block external network access to Vite dev server ports (default 5173) at the perimeter firewall or WAF if a patch cannot be applied immediately. Vite dev servers should never be exposed to untrusted networks; enforce this now as an emergency control per NIST AC-4 (Information Flow Enforcement) and CIS 4.4 (Implement and Manage a Firewall on Servers).

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), NIST CM-7 (Least Functionality), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory)

Compensating: Run 'npm list vite --depth=0' or 'grep -r "\"vite\"" /path/to/project/package.json' across all project directories to enumerate affected versions. Block TCP port 5173 immediately with: 'iptables -A INPUT -p tcp --dport 5173 -j DROP' (Linux) or 'netsh advfirewall firewall add rule name="Block Vite 5173" protocol=TCP dir=in localport=5173 action=block' (Windows). Use 'netstat -tlnp | grep 5173' or 'ss -tlnp sport = :5173' to confirm no active listeners remain externally reachable. Note Vite may also bind to ports 5174–5180 if 5173 is occupied — block the full range.

Evidence: Before blocking, capture a snapshot of active network connections from the Vite process: 'ss -tlnp | grep node' and 'lsof -i :5173' to document which process was listening and its PID. Record the Vite process start time from 'ps -eo pid,lstart,cmd | grep vite' to establish the exposure window. Preserve the current vite.config.js (or vite.config.ts) to document whether server.fs.strict was false (the default) and server.host was set to '0.0.0.0' or a non-localhost value, both of which confirm exploitable exposure.

Step 2: Detection — Query web server and application logs for URL patterns containing encoded or literal '../' sequences targeting paths that include '/@fs/', '/node_modules/.vite/', or '.map' extensions. Look for HTTP 200 responses to requests with path traversal patterns. Review SIEM for T1083 (File and Directory Discovery) alerts correlated with Vite process activity. Check for access to sensitive files (e.g., .env, private keys, config files) via source map requests. Apply NIST AU-6 (Audit Record Review, Analysis, and Reporting) and AU-3 (Content of Audit Records) to ensure log completeness.

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-3 (Content of Audit Records), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: Without a SIEM, run the following grep against Vite's stdout/stderr logs or any reverse proxy access logs (nginx/caddy): 'grep -E "(\.\.%2F|\.\.%252F|\.\./|/@fs/|/node_modules/\.vite/|\.map)" /var/log/nginx/access.log | grep " 200 "'. For a Sigma rule equivalent, use 'zgrep' across rotated logs with the same pattern. If Vite is fronted by nothing and logs to stdout, capture via: 'journalctl -u your-vite-service --since="YYYY-MM-DD" | grep -E "(/@fs/|\.map|\.\./)"'. Cross-reference source IP addresses from any HTTP 200 hits against known internal CIDR ranges to identify external vs. internal origin.

Evidence: Preserve raw Vite dev server stdout/stderr logs or reverse proxy access logs (nginx access.log, caddy access logs) covering the full exposure window — these are the primary artifact for this exploit. Extract all HTTP 200 responses to requests containing '/@fs/', percent-encoded traversal sequences ('%2F..%2F', '%252F'), or '.map' file requests that resolve outside the project root. Capture the source IP, timestamp, full request URI, and response size for each hit — response body size is critical because a successful traversal of a .env or private key file will return a non-trivial content-length. Collect the project's source map files (.js.map, .ts.map) from the dist or .vite cache directory to assess what internal code structure and variable names were exposed.

Step 3: Eradication — Upgrade vitejs/vite to 6.4.2, 7.3.2, or 8.0.5 per the patched release for your version branch. Verify the installed version via 'npm list vite' or 'yarn list vite'. If upgrade is not immediately possible, restrict the Vite dev server to localhost (server.host: 'localhost') and enforce server.fs.strict: true in vite.config as a temporary mitigation — note this does not fully remediate the vulnerability. Reference CIS 7.3 (Perform Automated Operating System Patch Management) and CIS 7.4 (Perform Automated Application Patch Management).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), 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: Upgrade using: 'npm install vite@latest' (or pin to '6.4.2', '7.3.2', or '8.0.5' explicitly) then run 'npm list vite' to confirm the resolved version — watch for nested dependencies that may pin an older version via package-lock.json. For monorepos, run 'npm list vite --workspaces' or 'yarn workspaces info | grep vite'. If immediate upgrade is blocked by dependency conflicts, apply the dual mitigation in vite.config: set 'server: { host: "localhost", fs: { strict: true } }' AND document this as a temporary workaround with a deadline — the Vite security advisory explicitly notes these settings reduce but do not eliminate the traversal surface. Verify vite.config changes took effect by attempting to access a known file outside the project root via curl from localhost: 'curl "http://localhost:5173/@fs/etc/passwd"' — a 403 indicates the control is active.

Evidence: Before upgrading, preserve a copy of the existing package.json and package-lock.json (or yarn.lock) to document the vulnerable version in the evidentiary record. Capture 'npm list vite --all' output to identify if the vulnerable version exists as a transitive dependency that would survive a top-level upgrade. After patching, run the same command again and diff the outputs to confirm the vulnerable version is no longer present anywhere in the dependency tree. Document the patch timestamp for audit trail purposes per NIST SI-2 (Flaw Remediation) requirements.

Step 4: Recovery — After patching, verify the installed Vite version matches a fixed release. Re-enable any temporarily blocked services. Rotate secrets or credentials present in any .env files, configuration files, or source map files that were accessible to the vulnerable server, treating them as potentially compromised. Monitor for anomalous authentication attempts or access patterns using rotated credentials per NIST AC-2 (Account Management) and D3-CRO (Credential Rotation).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AC-2 (Account Management), NIST IA-5 (Authenticator Management), NIST CP-10 (System Recovery and Reconstitution), CIS 5.2 (Use Unique Passwords)

Compensating: Enumerate all secrets potentially exposed via source map traversal: run 'grep -rE "(API_KEY|SECRET|PASSWORD|TOKEN|PRIVATE_KEY|DATABASE_URL)" .env .env.* vite.config.* *.config.js' to build a rotation checklist. For each rotated credential, set the old value as a canary in your logging pipeline — if it appears in any subsequent auth attempt, it confirms exfiltration and active abuse. Use 'git log --all --full-history -- .env' to determine if .env files were ever committed, as source maps may expose values that were only present at build time. Monitor authentication logs for the rotated service accounts for 14 days minimum, using 'grep ' against application logs if no SIEM is available.

Evidence: Before re-enabling services, document which .env files, configuration files, and source map artifacts were within the traversal reach of the vulnerable Vite server — specifically files accessible under the paths that server.fs.allow would have permitted. Collect a directory listing of '.vite/deps' and any dist/.map files to confirm what source code structure was cached and potentially readable. Preserve authentication logs from all services whose credentials appeared in reachable config files, covering the full exposure window, as these will be needed to determine if stolen credentials were used prior to rotation.

Step 5: Post-Incident — Review your development pipeline for any Vite dev server instances exposed outside localhost, and enforce a policy prohibiting dev server exposure to untrusted networks. Assess whether source map files in affected environments exposed proprietary logic, hardcoded secrets, or internal path structures. Map this gap to NIST SI-4 (System Monitoring) and CIS 8.2 (Collect Audit Logs) to improve detection of path traversal attempts going forward. Add a check for Vite version compliance to your software inventory process per CIS 2.1 (Establish and Maintain a Software Inventory) and CIS 2.2 (Ensure Authorized Software is Currently Supported).

NIST Phase: Post Incident

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

Controls: NIST SI-4 (System Monitoring), NIST CA-7 (Continuous Monitoring), CIS 8.2 (Collect Audit Logs), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Add a Vite version compliance check to your CI/CD pipeline by inserting a pre-build script: 'node -e "const v=require(\"./node_modules/vite/package.json\").version; const semver=require(\"semver\"); if(!semver.satisfies(v,\">=6.4.2 =7.3.2 =8.0.5\")) process.exit(1);"' — this fails the build if a vulnerable version is detected. Write a YARA rule targeting source map file patterns in web server directories to flag accidental exposure. Create a persistent Sigma rule for your log pipeline detecting HTTP requests matching '/@fs/' or percent-encoded traversal sequences against port 5173/5174 from non-RFC1918 source addresses. Conduct a one-time audit of all development environments using 'find / -name "vite.config.*" 2>/dev/null' to enumerate any forgotten or shadow dev server deployments.

Evidence: Compile the final forensic package: the raw access logs with traversal attempts highlighted, the directory listing of files that were accessible via the traversal path, the before/after package.json diff confirming remediation, and a list of all rotated credentials with rotation timestamps. If HTTP 200 responses with non-trivial content-length were observed for .env or config file requests, treat this as a confirmed data exposure and document it in the incident report with specificity about which files were returned — this determination affects breach notification obligations. Retain all evidence per your organization's retention policy and NIST AU-11 (Audit Record Retention) requirements.

Detection Guidance

Search web access logs for HTTP requests containing '../', '%2e%2e%2f', '%2e%2e/', or '..' patterns in URL paths that also include '.map', '/@fs/', or '/node_modules/.vite/deps/'. Filter for HTTP 200 responses to these requests; a successful traversal will return content rather than a 404 or 403. In SIEM, correlate Vite process file read events (if host-level logging is enabled) with file paths outside the configured project root. Behavioral indicators include requests for files at unusual depths relative to the project root (e.g., '/../../../../etc/' patterns) and unexpectedly large .map file responses. MITRE T1083 (File and Directory Discovery) and T1552.001 (Credentials in Files) are the relevant technique signatures. Apply NIST AU-2 (Event Logging) to confirm .map request logging is enabled across all Vite-hosting systems. No public IOCs (IPs, domains, hashes) have been confirmed for this vulnerability at this time.

Framework Mappings

MITRE-ATTACK

- **T1083** — File and Directory Discovery
- **T1552.001** — Credentials In Files

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control

NIST-800-53R5

- **AC-3** — Access Enforcement
- **SI-10** — Information Input Validation
- **IR-5** — Incident Monitoring

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **16.12** — Implement Code-Level Security Checks

NIST-CSF-2

- **DE.AE-08** — Incidents are declared when adverse events meet the defined incident criteria

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1083	File and Directory Discovery	Discovery
T1552.001	Credentials In Files	Credential-Access

Sources

Source	URL	Tier
vulncheck_kev	https://nvd.nist.gov/vuln/detail/CVE-2026-39365	T1
CVE-2026-39365: Vite Path Traversal Vulnerability - SentinelOne	https://www.sentinelone.com/vulnerability-database/cve-2026-39365/	T3
CVE-2026-39365 - TuxCare Vulnerabilities	https://tuxcare.com/cve-tracker/cve/details/cve-2026-39365/?product...	T3
CVE-2026-39365: Affected Packages & Status Updates | Advisories	https://images.chainguard.dev/security/CVE-2026-39365	T3
CVE-2026-39365 | Tenable®	https://www.tenable.com/cve/CVE-2026-39365	T3
CISA KEV	https://www.cisa.gov/known-exploited-vulnerabilities-catalog	T1

Generated 2026-05-22 13:51 UTC by TJS Security Command Center