INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-07-02 07:14 UTC

DuneSlide: Zero-Click Prompt Injection Breaks Cursor's AI Sandbox, Exposes Developer Machines to Full Command Execution

CVE VULNERABILITY | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CVE-2026-0380
Type	CVE Vulnerability
CVE ID	CVE-2026-50548, CVE-2026-50549
Severity	CRITICAL
CVSS Base Score	9.5
EPSS Score	0.0064 (46th percentile)
Affected Products	Cursor AI code editor, all versions prior to 3.0
Published	2026-07-01T10:42:54
Discovery Source	Rss

Executive Summary

Two critical vulnerabilities in the Cursor AI code editor, tracked as 'DuneSlide' (CVE-2026-50548, CVE-2026-50549, CVSS 9.5), reportedly allow an attacker to execute arbitrary commands on a developer's machine without any user interaction, by injecting malicious instructions through AI-ingested content such as web search results or connected services. All Cursor versions prior to 3.0 are reported as affected. Organizations with developers using Cursor should treat this as an urgent patching priority, as successful exploitation could give an attacker direct access to source code, credentials, and internal systems reachable from the developer's workstation.

Technical Analysis

CVE-2026-50548 and CVE-2026-50549 (collectively 'DuneSlide') are reported to affect Cursor AI code editor versions prior to 3.0, with a CVSS base score of 9.5 per NVD records (The Hacker News source material references 9.8; the NVD figure is used as authoritative). The vulnerability chain enables zero-click prompt injection: malicious payloads embedded in AI-ingested content (MCP-connected services, web search results) reportedly escape the editor's terminal sandbox and achieve arbitrary command execution on the host OS without developer interaction. CWE contributors include CWE-22 (path traversal), CWE-59 (symlink following), CWE-693 (protection mechanism failure), and CWE-74 (injection). Relevant MITRE ATT&CK techniques

include T1059/T1059.004 (command and scripting interpreter), T1190 (exploit public-facing application), T1195 (supply chain compromise), T1552.001 (credentials in files), and T1574.010 (hijack execution flow). Source material originates from The Hacker News (T2) and NVD/CVE.org records (T1). A vendor patch advisory from Cursor has not been located in current source material; remediation guidance (upgrade to version 3.0) is sourced from the Hacker News report. Recommend obtaining official Cursor release notes before deployment. CVSS vector string is pending NVD publication.

Action Checklist

1. Step 1: Containment. Audit all developer workstations running Cursor versions prior to 3.0. According to the source report, disable or restrict Cursor's MCP-connected services and AI web search integration until the upgrade to version 3.0 is confirmed. Isolate developer machines with access to production repositories, cloud credential stores, or internal VPN segments as a precaution while patching is underway.
2. Step 2: Detection. Review terminal and shell execution logs on developer endpoints for unexpected process spawning from the Cursor editor process. Look for anomalous child processes, file system access patterns consistent with CWE-22 (path traversal) or CWE-59 (symlink following), and unusual outbound network connections originating from Cursor. Audit logs per NIST AU-2 (Event Logging) and AU-6 (Audit Record Review, Analysis, and Reporting) for command execution events tied to the editor process. Use endpoint detection tools to flag T1059/T1059.004 (scripting interpreter invocation) and T1552.001 (file-based credential access) behaviors sourced from the Cursor process tree.
3. Step 3: Eradication. Upgrade all Cursor AI code editor installations to version 3.0 or later, which is identified in the source report as the remediated release. Verify the upgrade against the vendor's official release notes before deployment. Until a vendor patch advisory URL can be independently confirmed, obtain the upgrade directly from the official Cursor website. Ensure patch deployment follows CIS Controls 7.3 (Automated OS Patch Management) and 7.4 (Automated Application Patch Management) to achieve coverage across all developer endpoints.
4. Step 4: Recovery. After upgrading, verify no unauthorized processes, scheduled tasks, or persistence mechanisms were established on affected developer machines. Check for signs of T1574.010 (Hijack Execution Flow) and T1055 (Process Injection) techniques, which are common post-exploitation persistence methods. Rotate credentials and API keys accessible from developer environments, consistent with D3-CRO (Credential Rotation). Re-enable MCP and web search integrations only after confirming the patched version is running. Continue monitoring endpoint logs per NIST AU-6 for 30 days post-remediation.
5. Step 5: Post-Incident. Review controls governing developer tooling procurement and approval (NIST AC-6, Least Privilege; CIS 2.1, Software Inventory; CIS 2.2, Ensure Authorized Software is Currently Supported). Evaluate whether AI-integrated development tools are in scope for your vulnerability management program under CIS 7.1. Assess whether MCP-connected service permissions follow least-privilege principles. Document findings and update developer endpoint security standards to explicitly address AI agent tool risks.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO and legal/privacy counsel immediately if forensic evidence confirms successful command execution (child processes, modified credential files, or unauthorized outbound connections from the Cursor process tree), or if any compromised developer machine had access to production infrastructure credentials, customer PII/PHI repositories, or signing keys — triggering breach notification assessment under applicable data protection regulations.
Recovery Notes	After confirming Cursor 3.0 is deployed across all developer endpoints, perform a targeted sweep of all repositories committed to during the exposure window for unauthorized code changes or secrets inadvertently committed, and revoke and reissue any cloud provider API keys, SSH keys, and service tokens that resided in the developer environment. Monitor Sysmon Event ID 1 process creation logs and outbound DNS/network activity from developer endpoints for 30 days post-remediation, specifically watching for delayed persistence mechanisms (scheduled tasks, cron jobs, or modified shell profiles) that a successful DuneSlide exploitation may have planted prior to detection. Re-enable Cursor MCP and web search integrations only after validating that the vendor's Cursor 3.0 release notes explicitly confirm remediation of CVE-2026-50548 and CVE-2026-50549.
Forensic Artifacts	Cursor session and conversation logs (~/.cursor/logs/ on Linux/macOS, %APPDATA%\Cursor\logs\ on Windows) — these may contain the raw AI response payload carrying the injected DuneSlide prompt instruction delivered via malicious web search result or MCP-connected service response | Sysmon Event ID 1 (Process Create) records showing cursor.exe or the Cursor Electron renderer as ParentImage for any shell interpreter (cmd.exe, powershell.exe, bash, sh, zsh, python) — the direct forensic signature of CVE-2026-50548/50549 arbitrary command execution | Cursor MCP server configuration file (~/.cursor/settings.json or %APPDATA%\Cursor\User\settings.json) — documents which external services were connected and their permission scopes at the time of potential exploitation, establishing the attack surface for the zero-click vector | Shell history files (~/.bash_history, ~/.zsh_history, PowerShell PSReadLine history at %APPDATA%\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt) for commands that neither the developer typed nor can account for — injected commands executed via the vulnerability would appear here without corresponding developer recall | File system change events (Sysmon Event ID 11 — FileCreate) and recently modified files in ~/.ssh/, ~/.aws/, ~/.config/, and any .env or secrets files within developer project directories opened in Cursor — identifying whether a successful DuneSlide payload performed credential harvesting or implanted persistence before containment

Per-Action IR Details

Step 1: Containment — Audit all developer workstations running Cursor versions prior to 3.0. According to the source report, disable or restrict Cursor's MCP-connected services and AI web search integration until the upgrade to version 3.0 is confirmed. Isolate developer machines with broad internal network access as a precaution while patching is underway.

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), NIST AC-17 (Remote Access), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: Use Windows Firewall (netsh advfirewall) or iptables/ufw to block all outbound connections from the Cursor process (cursor.exe or cursor binary) to external hosts: `netsh advfirewall firewall add rule name='Block Cursor Outbound' program='%LOCALAPPDATA%\Programs\cursor\cursor.exe' dir=out action=block`. On macOS/Linux, use `pf` or `ufw` with process-owner rules. Disable MCP server entries in Cursor's settings.json (~/.cursor/settings.json or %APPDATA%\Cursor\User\settings.json) by removing or nulling mcp.servers entries. Enumerate Cursor installs via:

`Get-WmiObject Win32_Product | Where-Object {$_.Name -like '*Cursor*'}` (Windows) or `find /Applications /home -name 'cursor' -type f 2>/dev/null` (macOS/Linux).

Evidence: Before isolating any developer machine, capture: (1) full memory image using WinPmem or LiME to preserve any injected prompt payload or in-memory command artifacts from the Cursor AI process; (2) active network connections from the Cursor process — `Get-NetTCPConnection | Where-Object {$_.OwningProcess -eq (Get-Process cursor).Id}` or `netstat -ano | findstr `; (3) current Cursor process tree — `Get-WmiObject Win32_Process | Where-Object {$_.ParentProcessId -eq (Get-Process cursor).Id}` — to capture any live child processes spawned via CVE-2026-50548/50549 prompt injection before network isolation destroys C2 connectivity.

Step 2: Detection — Review terminal and shell execution logs on developer endpoints for unexpected process spawning from the Cursor editor process. Look for anomalous child processes, file system access patterns consistent with CWE-22 (path traversal) or CWE-59 (symlink following), and unusual outbound network connections originating from Cursor. Audit logs per NIST AU-2 (Event Logging) and AU-6 (Audit Record Review, Analysis, and Reporting) for command execution events tied to the editor process. Use endpoint detection tools to flag T1059/T1059.004 (scripting interpreter invocation) and T1552.001 (file-based credential access) behaviors sourced from the Cursor process tree.

NIST Phase: Detection Analysis

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

Controls: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: Deploy Sysmon (config with SwiftOnSecurity baseline) and enable Event ID 1 (Process Create) filtering on ParentImage matching the Cursor executable path — look for cmd.exe, powershell.exe, sh, bash, zsh, or python spawned as children of cursor.exe or the Cursor Electron renderer process. Query with: `Get-WinEvent -LogName 'Microsoft-Windows-Sysmon/Operational' | Where-Object {$_.Id -eq 1 -and $_.Message -like '*cursor*'}`. On Linux/macOS, enable auditd with `-a always,exit -F arch=b64 -S execve -F ppid=` or use osquery: `SELECT pid, ppid, name, cmdline FROM processes WHERE ppid IN (SELECT pid FROM processes WHERE name LIKE '%cursor%')`. Additionally, grep Cursor's extension host log (~/.cursor/logs/ or %APPDATA%\Cursor\logs\) for injected instruction patterns — look for base64-encoded strings or shell metacharacters in AI response payloads.

Evidence: This step is analytical and does not alter live state, but collect before any eradication action: (1) Cursor AI conversation/session logs at ~/.cursor/logs/ or %APPDATA%\Cursor\logs\ which may contain the injected prompt payload from malicious web search results or MCP-connected service responses; (2) Sysmon Event ID 1 logs showing process lineage from cursor.exe to any shell or interpreter; (3) file system audit events (Sysmon Event ID 11 — FileCreate) for newly created scripts, .ssh/authorized_keys modifications, or credential files accessed under the developer's home directory; (4) DNS query logs (Sysmon Event ID 22) from the Cursor process for anomalous external resolution patterns indicating C2 beacon or data exfiltration.

Step 3: Eradication — Upgrade all Cursor AI code editor installations to version 3.0 or later, which is identified in the source report as the remediated release. Verify the upgrade against the vendor's official release notes before deployment. Until a vendor patch advisory URL can be independently confirmed, obtain the upgrade directly from the official Cursor website. Apply CIS 7.3 (Perform Automated Operating System Patch Management) and CIS 7.4 (Perform Automated Application Patch Management) processes to ensure coverage across all developer endpoints.

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: For teams without a centralized patch management platform, script the upgrade using Cursor's built-in update mechanism or direct binary replacement: on Windows, run the Cursor 3.0 installer silently with `cursor-installer.exe /S`; on macOS, replace the app bundle via `cp -R Cursor.app /Applications/`; on Linux, replace the AppImage or package. Post-upgrade, verify the installed version by checking the About dialog or parsing

`~/.cursor/package.json` (Linux/macOS) or `%LOCALAPPDATA%\Programs\cursor\resources\app\package.json` (Windows) for `"version": "3.0.x"`. Maintain a simple CSV asset inventory of all developer endpoints, Cursor version, and upgrade confirmation timestamp — auditable with: `Get-ItemProperty HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\* | Where-Object {$_.DisplayName -like '*Cursor*'} | Select DisplayName, DisplayVersion`.

Evidence: Before applying the Cursor 3.0 patch on any potentially compromised machine, capture: (1) a forensic disk image or at minimum a targeted collection of ~/.cursor/ (Linux/macOS) or %APPDATA%\Cursor\ (Windows) including settings.json, mcp server configs, and session/conversation logs that may contain the injected DuneSlide payload; (2) running process list and open file handles for the Cursor process (`handle.exe -p cursor` on Windows, `lsof -p ` on Linux/macOS) before the installer terminates the process; (3) any persistence artifacts already placed by a successful exploitation — scheduled tasks (`schtasks /query /fo LIST /v`), cron jobs (`crontab -l`), startup folder contents, or ~/.bashrc / ~/.zshrc modifications that would survive the application upgrade.

Step 4: Recovery — After upgrading, verify no unauthorized processes, scheduled tasks, or persistence mechanisms were established on affected developer machines, referencing MITRE ATT&CK T1574.010 and T1055 (process injection) as indicators. Rotate credentials and API keys accessible from developer environments, consistent with D3-CRO (Credential Rotation). Re-enable MCP and web search integrations only after confirming the patched version is running. Continue monitoring endpoint logs per NIST AU-6 for 30 days post-remediation.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AC-2 (Account Management), NIST AC-12 (Session Termination), NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Enumerate and rotate all credentials that were accessible from compromised developer environments: check ~/.ssh/ for private keys, ~/.aws/credentials, ~/.config/gcloud/, %APPDATA%\npm\etc\npmrc, and any .env files in recently opened project directories (recoverable from Cursor's recent workspace list at ~/.cursor/storage.json). Use `git log --all --oneline` on any repositories the developer worked in to check for unauthorized commits. For persistence sweep: `schtasks /query /fo LIST /v | findstr /i 'cursor\|ai\|mcp'` (Windows), `crontab -l && ls /etc/cron.*` (Linux), `launchctl list | grep -i cursor` (macOS). Verify Cursor 3.0 is running before re-enabling MCP by checking settings.json and confirming no legacy mcp.servers entries point to untrusted endpoints.

Evidence: Before rotating credentials (which constitutes an irreversible live-state change for active sessions), capture: (1) currently active SSH agent identities (`ssh-add -l`) and any forwarded agent sockets that a DuneSlide exploitation could have leveraged for lateral movement to internal systems; (2) browser/IDE stored credential stores and token caches accessible from the Cursor process context — specifically VS Code / Cursor token storage at %APPDATA%\Cursor\User\globalStorage\; (3) a snapshot of recently accessed files (Windows Prefetch, macOS .bash_history / .zsh_history, Linux ~/.local/share/recently-used.xbel) to establish what sensitive files the injected command may have read or exfiltrated before containment.

Step 5: Post-Incident — Review controls governing developer tooling procurement and approval (NIST AC-6, Least Privilege; CIS 2.1, Software Inventory; CIS 2.2, Ensure Authorized Software is Currently Supported). Evaluate whether AI-integrated development tools are in scope for your vulnerability management program under CIS 7.1. Assess whether MCP-connected service permissions follow least-privilege principles. Document findings and update developer endpoint security standards to explicitly address AI agent tool risks.

NIST Phase: Post Incident

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

Controls: NIST AC-6 (Least Privilege), NIST AU-11 (Audit Record Retention), 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), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Conduct a lessons-learned session within 5 business days documenting: how Cursor reached developer machines without triggering vulnerability management coverage, which MCP-connected services had permissions exceeding developer need-to-know, and whether AI tool ingestion of external content (web search, third-party service responses) is treated as an untrusted input surface in your threat model. Create a software inventory entry for Cursor and all AI-integrated IDE tooling in your asset register with mandatory version tracking. Draft a one-page policy addendum requiring that AI agent tools with external data ingestion capabilities undergo security review before approval, explicitly referencing the DuneSlide zero-click prompt injection vector as the documented rationale.

Evidence: Retain all forensic artifacts collected during this incident for a minimum period consistent with your audit record retention policy (NIST AU-11): Cursor session/conversation logs containing any recovered injected prompt payloads, Sysmon process creation logs from affected developer endpoints, network flow records showing Cursor outbound connections during the exposure window, and any credential rotation records. These artifacts constitute the evidentiary record for a post-incident review and may be required for regulatory notification assessment if developer machines had access to PII, source code repositories containing customer data, or production infrastructure credentials.

Detection Guidance

Monitor endpoint process trees on developer workstations for child processes spawned by the Cursor editor binary, particularly shell interpreters (bash, sh, zsh, PowerShell, cmd.exe) that were not explicitly invoked by the developer. Flag unexpected file system access in sensitive directories (home directories, .ssh, .aws, .env files) traceable to the Cursor process, consistent with CWE-22 path traversal and T1552.001 credential harvesting. Collect and review audit logs per NIST AU-2 and AU-12 for command execution events. Apply D3-SFA (System File Analysis) to monitor for modification of configuration files or authentication stores accessible from the editor's working directory. Look for outbound network connections from the Cursor process to unrecognized external hosts, which may indicate exfiltration following code execution. Because the attack vector is indirect (malicious content ingested by the AI agent), there may be no user-visible trigger event; rely on behavioral detection rather than user-reported anomalies. No confirmed IOC signatures (hashes, IPs, domains) are available from current source material.

Framework Mappings

MITRE-ATTACK

- **T1574.010** — Services File Permissions Weakness
- **T1190** — Exploit Public-Facing Application
- **T1566** — Phishing
- **T1059** — Command and Scripting Interpreter
- **T1059.004** — Unix Shell
- **T1557** — Adversary-in-the-Middle
- **T1106** — Native API
- **T1055** — Process Injection
- **T1552.001** — Credentials In Files
- **T1195** — Supply Chain Compromise

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
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-8** — Spam Protection
- **CM-7** — Least Functionality
- **AC-6** — Least Privilege
- **SA-9** — External System Services
- **SR-2** — Supply Chain Risk Management Plan
- **SR-3** — Supply Chain Controls and Processes
- **AC-3** — Access Enforcement
- **SI-10** — Information Input Validation

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control
- **A03:2021** — Injection

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **16.12** — Implement Code-Level Security Checks
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain

SOC2-TSC

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1574.010	Services File Permissions Weakness	Persistence
T1190	Exploit Public-Facing Application	Initial-Access

Technique ID	Technique Name	Tactic
T1566	Phishing	Initial-Access
T1059	Command and Scripting Interpreter	Execution
T1059.004	Unix Shell	Execution
T1557	Adversary-in-the-Middle	Credential-Access
T1106	Native API	Execution
T1055	Process Injection	Defense-Evasion
T1552.001	Credentials In Files	Credential-Access
T1195	Supply Chain Compromise	Initial-Access

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/07/critical-cursor-flaws-could-let-p...	T2
CVE-2026-50548 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-50548	T1
Preventing CVE-2026-50549: Secure Your Server Now - BitNinja	https://bitninja.com/blog/preventing-cve-2026-50549-secure-your-ser...	T3
CVE-2026-50549 - Vulnerability Details - OpenCVE	https://app.opencve.io/cve/CVE-2026-50549	T3
CVE-2026-50549 - CVE Record	https://www.cve.org/CVERecord?id=CVE-2026-50549	T1

Generated 2026-07-02 07:14 UTC by TJS Security Command Center