INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-21 13:39 UTC

# Prompt Injection Vulnerabilities Across Six AI Developer Tool Platforms Enable Code Execution and Secret Exfiltration

CVE VULNERABILITY | CRITICAL | CVSS 9.5

| | |
|---|---|
| SCC Item ID | SCC-CVE-2026-0060 |
| Type | CVE Vulnerability |
| CVE ID | CVE-2026-21520 |
| Severity | CRITICAL |
| CVSS Base Score | 9.5 |
| EPSS Score | 0.0009 (26th percentile) |
| Affected Products | Google Antigravity IDE, Anthropic Claude Code, GitHub Copilot Agent, Google Gemini CLI, Microsoft Copilot Studio, Salesforce Agentforce, Cursor IDE, claude-code-action (GitHub Actions) |
| Published | 2026-04-21T06:22:00 |
| Discovery Source | Rss |

## Executive Summary

Researchers disclosed prompt injection and code execution vulnerabilities affecting eight AI-powered developer tools: GitHub Copilot Agent, Microsoft Copilot Studio, Google Gemini CLI, Anthropic Claude Code, Cursor IDE, Salesforce Agentforce, Google Antigravity IDE, and the claude-code-action GitHub Actions integration. Attackers can embed malicious instructions in content processed by these AI agents, causing them to execute code, access file systems, and exfiltrate secrets such as API keys and credentials without user awareness or approval. Organizations using these tools in development pipelines face direct risk of source code theft, credential compromise, and supply chain contamination.

## Technical Analysis

CVE-2026-21520 describes a class of prompt injection vulnerabilities affecting agentic AI developer tools that combine natural language processing with execution capabilities (shell access, file system read/write, secret retrieval). The root cause is a trust boundary failure: AI models with tool-use permissions process attacker-controlled content (malicious repository files, crafted code comments, poisoned documentation) as legitimate instructions, bypassing human-in-the-loop review. Affected products include Google Antigravity IDE, Anthropic Claude Code, GitHub Copilot Agent, Google Gemini CLI, Microsoft Copilot Studio, Salesforce

Agentforce, Cursor IDE, and the claude-code-action GitHub Actions integration. CVSS base score is 9.5 (Critical); this should be confirmed against the authoritative NVD record at nvd.nist.gov/vuln/detail/CVE-2026-21520 before operational decisions are made. EPSS score is 0.00091 (25th percentile) as of the configuration date, indicating low observed exploitation activity at time of publication. Applicable CWEs: CWE-78 (OS Command Injection), CWE-77 (Command Injection), CWE-20 (Improper Input Validation, closest recognized analog for prompt injection), CWE-116 (Improper Encoding/Escaping), CWE-94 (Code Injection). Relevant MITRE ATT&CK techniques include T1059 (Command and Scripting Interpreter), T1106 (Native API), T1552.001 (Credentials in Files), T1195.001 (Compromise Software Dependencies and Development Tools), T1190 (Exploit Public-Facing Application), and T1560 (Archive Collected Data). Patch status varies by vendor; confirm remediation status against individual vendor advisories. A Microsoft Security Advisory is available at msrc.microsoft.com/update-guide/vulnerability/CVE-2026-21520. Source quality score for this item is 0.632; NVD and CVE.org records are cited but their content has not been independently verified by this analysis.

## Action Checklist

1. Step 1: Containment, Immediately audit which AI developer tools from the affected list (GitHub Copilot Agent, Microsoft Copilot Studio, Google Gemini CLI, Anthropic Claude Code, Cursor IDE, Salesforce Agentforce, Google Antigravity IDE, claude-code-action) are active in your development environment. Restrict or suspend agent modes that have file system access, shell execution, or secret retrieval permissions until patches are confirmed applied. Do not allow these agents to process untrusted third-party repositories or external content without human review.
2. Step 2: Detection, Review CI/CD pipeline logs, IDE telemetry, and GitHub Actions run logs for anomalous outbound connections, unexpected file reads targeting secret stores (e.g., .env files, ~/.aws/credentials, ~/.ssh), or shell commands not initiated by a human user. Query SIEM for process execution events (T1059) spawned from AI agent processes. Flag any network connections from developer workstations to unexpected external endpoints following AI agent activity.
3. Step 3: Eradication, Apply vendor-issued patches as they become available. For Microsoft Copilot Studio, consult the MSRC advisory at msrc.microsoft.com/update-guide/vulnerability/CVE-2026-21520. For all other affected tools, monitor vendor security channels and update to patched versions immediately upon release. Disable tool-use or agentic features in configurations where patches are not yet available. Remove or rotate any secrets (API keys, tokens, credentials) that AI agents had access to, treating them as potentially compromised.
4. Step 4: Recovery, After patching, verify agent configurations enforce least-privilege: AI agents should have no more file system or execution access than explicitly required. Re-scan repositories and CI/CD pipelines processed by affected agents for unauthorized changes or exfiltrated data. Confirm outbound connections from developer environments return to baseline. Monitor credential stores and API key usage for anomalous activity for a minimum of 30 days post-remediation.
5. Step 5: Post-Incident, This vulnerability class exposes a systemic gap: AI agents with execution capabilities were granted implicit trust equivalent to authenticated human developers. Implement input sanitization and privilege separation controls for all agentic AI integrations. Establish a review process for any AI tool requesting file system, shell, or secret access before deployment. Map findings against NIST SP 800-53 SI-10 (Information Input Validation) and SA-11 (Developer Testing and Evaluation). Update your AI/ML tool procurement policy to require documented security assessments of agentic capabilities.

## IR / Forensic Enrichment

| | |
|---|---|
| **Triage Priority** | IMMEDIATE |
| **Escalation Criteria** | Escalate to CISO and legal/privacy counsel immediately if forensic review confirms any AI agent accessed and transmitted secrets (API keys, OAuth tokens, SSH private keys, .env credentials) to an external endpoint, or if affected CI/CD pipelines had access to production environment credentials, as this constitutes a credential compromise with potential regulatory notification obligations under applicable data protection frameworks. |
| **Recovery Notes** | After patching all affected AI developer tools and rotating all secrets accessible to those agents, re-enable agentic features only under a documented least-privilege configuration that restricts file system scope to project working directories and prohibits access to ~/.aws, ~/.ssh, ~/.config, and .env files. Monitor all newly issued API keys and tokens via cloud provider audit logs (AWS CloudTrail, GCP Audit Logs, Azure Monitor) for anomalous usage patterns — including access from unexpected IP ranges or service principals — for a minimum of 30 days, as compromised credentials may have been cached by an attacker prior to rotation. Conduct a full re-scan of all repositories processed by affected agents using gitleaks or truffleHog before re-enabling agent write access to any repository. |
| **Forensic Artifacts** | GitHub Actions run logs for all workflows invoking claude-code-action: examine step-level stdout/stderr for unexpected env variable dumps, curl/wget commands to non-GitHub endpoints, or base64-encoded payloads in step outputs — accessible via 'gh run view --log' or the Actions tab in the repository UI \| Cursor IDE and Electron application logs at %APPDATA%\Cursor\logs\ (Windows) or ~/Library/Logs/Cursor/ (macOS) recording tool-call invocations made by the AI agent, including filesystem read operations targeting .env files, credential directories, and SSH key paths \| Gemini CLI session logs and request history at ~/.gemini/logs/ or equivalent XDG_DATA_HOME path, capturing the full prompt context sent to the Gemini API including any injected instructions from malicious repository content processed during the session \| Network proxy or endpoint DNS/flow logs showing outbound HTTPS POST requests from developer workstation processes (node.exe, electron.exe, python.exe, cursor.exe) to AI vendor API endpoints (generativelanguage.googleapis.com, api.anthropic.com, api.openai.com) with anomalously large request payloads indicating potential secret exfiltration in the prompt or tool-call response body \| AWS CloudTrail, GCP Audit Logs, or Azure Activity Logs filtered on access key IDs and service account credentials that were present in developer environment variables or .env files during the AI agent exposure window, to determine whether exfiltrated credentials were subsequently used by an attacker from external IP addresses |

### Per-Action IR Details

**Step 1: Containment — Immediately audit which AI developer tools from the affected list (GitHub Copilot Agent, Microsoft Copilot Studio, Google Gemini CLI, Anthropic Claude Code, Cursor IDE, Salesforce Agentforce, Google Antigravity IDE, claude-code-action) are active in your development environment. Restrict or suspend agent modes that have file system access, shell execution, or secret retrieval permissions until patches are confirmed applied. Do not allow these agents to process untrusted third-party repositories or external content without human review.**

**NIST Phase:** Containment

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

**Controls:** NIST IR-4 (Incident Handling), NIST AC-6 (Least Privilege), NIST CM-7 (Least Functionality), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 2.3 (Address Unauthorized Software)

**Compensating:** Run 'gh extension list' and 'gh copilot --version' on developer workstations to enumerate GitHub Copilot Agent installations. For Gemini CLI, check 'pip show gemini-cli' or 'npm list -g @google/generative-ai'. Enumerate Cursor IDE installations via registry key HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall on Windows or 'ls /Applications/Cursor.app' on macOS. Disable agentic/agent-mode features in VS Code settings.json by setting 'github.copilot.editor.enableAutoCompletions' and removing agent-mode scopes, or for Cursor, disable via Settings > Features > Agent. For claude-code-action, disable the GitHub Actions workflow file by commenting out the 'on:' trigger block and committing the change immediately.

**Evidence:** Before restricting agents, capture the current permission scope granted to each AI tool: export GitHub Copilot OAuth token scopes via 'gh auth status -t'; capture Gemini CLI config at ~/.config/gemini/ and any GOOGLE_API_KEY environment variables; snapshot Cursor IDE settings at %APPDATA%\Cursor\User\settings.json (Windows) or ~/Library/Application Support/Cursor/User/settings.json (macOS); capture claude-code-action workflow YAML files under .github/workflows/ in all repositories; document Salesforce Agentforce connected app permissions from Setup > Connected Apps > OAuth Usage; preserve Microsoft Copilot Studio agent configuration exports showing which connectors and actions were enabled.

**Step 2: Detection — Review CI/CD pipeline logs, IDE telemetry, and GitHub Actions run logs for anomalous outbound connections, unexpected file reads targeting secret stores (e.g., .env files, ~/.aws/credentials, ~/.ssh), and shell commands not initiated by a human user. Query SIEM for process execution events (T1059) spawned from AI agent processes. Flag any network connections from developer workstations to unexpected external endpoints following AI agent activity.**

**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 logs Event ID 1 (Process Create) and Event ID 3 (Network Connection) with parent process filtering for node.exe, python.exe, electron.exe, cursor.exe, and 'gh' CLI — these are the runtime processes for the affected AI tools. Use this PowerShell to query for suspicious child processes: Get-WinEvent -LogName Security | Where-Object {$_.Id -eq 4688 -and $_.Message -match 'cmd.exe|powershell.exe|bash'} | Where-Object {$_.Message -match 'node.exe|cursor.exe|electron.exe'}. For GitHub Actions, use 'gh run list --json conclusion,workflowName,createdAt' and 'gh run view --log' to retrieve step-level execution logs and inspect for unexpected curl, wget, or env variable dumps. For network detection without SIEM, run Wireshark or tcpdump filtered on developer workstations: 'tcpdump -i any -w capture.pcap host and not (net 192.168.0.0/16 or net 10.0.0.0/8)' to capture unexpected external connections from AI agent processes. Write a Sigma rule matching ParentImage containing 'node.exe' or 'cursor.exe' with CommandLine containing '.env' or '.aws' or '.ssh'.

**Evidence:** Collect GitHub Actions run logs for all workflows invoking claude-code-action, specifically examining step outputs for base64-encoded blobs, curl commands to non-GitHub endpoints, or 'cat ~/.aws/credentials' patterns. Pull Gemini CLI request/response logs from ~/.gemini/logs/ or equivalent. For Cursor IDE, examine ~/.cursor/logs/ and the Electron app logs at %APPDATA%\Cursor\logs\ for tool-call invocations showing filesystem reads. Capture network flow logs or proxy logs showing outbound HTTPS POST requests from developer workstations to non-corporate endpoints occurring within the session window of AI agent usage. On Windows, query Sysmon Event ID 11 (FileCreate) and Event ID 23 (FileDelete) for access to %USERPROFILE%\.aws\credentials, %USERPROFILE%\.ssh\id_rsa, and any .env files in project directories.

**Step 3: Eradication — Apply vendor-issued patches as they become available. For Microsoft Copilot Studio, consult the MSRC advisory at msrc.microsoft.com/update-guide/vulnerability/CVE-2026-21520. For all other affected tools, monitor vendor security channels and update to patched versions immediately upon release. Disable tool-use or agentic features in configurations where patches are not yet available. Remove or rotate any secrets (API keys, tokens, credentials) that AI agents had access to, treating them as potentially compromised.**

**NIST Phase:** Eradication

**Reference:** NIST 800-61r3 §3.4 — Eradication

**Controls:** NIST SI-2 (Flaw Remediation), NIST SI-5 (Security Alerts, Advisories, and Directives), NIST IA-5 (Authenticator Management), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 5.2 (Use Unique Passwords)

**Compensating:** Until vendor patches are released, disable agentic features at the configuration level: for GitHub Copilot Agent, remove the 'agents' scope from the OAuth app via GitHub Settings > Applications > Authorized OAuth Apps; for Gemini CLI, unset GOOGLE_API_KEY and GEMINI_API_KEY environment variables system-wide via /etc/environment or Windows System Environment Variables; for claude-code-action, set the workflow to 'if: false' as an emergency disable without deleting the file. For secret rotation, use AWS CLI 'aws iam delete-access-key --access-key-id ' and immediately create replacement keys; for GitHub PATs, revoke via Settings > Developer Settings > Personal Access Tokens and audit all tokens with 'repo' scope that were accessible to any affected agent. Use truffleHog ('trufflehog git file://.') on all repositories processed by affected agents to identify secrets that may have been read and are now in git history.

**Evidence:** Before rotating secrets, document the full list of credentials accessible to each affected agent: for GitHub Actions workflows using claude-code-action, enumerate all repository secrets and environment secrets via 'gh secret list --repo '; for Gemini CLI and Anthropic Claude Code, audit shell environment variables in developer .bashrc/.zshrc/.profile for API keys; capture the Microsoft Copilot Studio connector authentication configurations showing which credentials were bound to the agent. Preserve a point-in-time export of AWS CloudTrail, GCP Audit Logs, or Azure Activity Logs for the 30-day window prior to discovery, as these will show whether exfiltrated API keys were already used by an attacker before rotation.

**Step 4: Recovery — After patching, verify agent configurations enforce least-privilege: AI agents should have no more file system or execution access than explicitly required. Re-scan repositories and CI/CD pipelines processed by affected agents for unauthorized changes or exfiltrated data. Confirm outbound connections from developer environments return to baseline. Monitor credential stores and API key usage for anomalous activity for a minimum of 30 days post-remediation.**

**NIST Phase:** Recovery

**Reference:** NIST 800-61r3 §3.5 — Recovery

**Controls:** NIST IR-4 (Incident Handling), NIST AC-6 (Least Privilege), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 6.3 (Require MFA for Externally-Exposed Applications)

**Compensating:** Run 'git log --all --oneline --author="github-actions[bot]"' and 'git log --all --oneline --author="copilot"' across all affected repositories to identify commits made by AI agent identities during the exposure window; diff suspicious commits against known-good baselines. Use truffleHog or gitleaks ('gitleaks detect --source . -v') on all repositories to detect secrets inadvertently committed by a prompt-injected agent. For API key usage monitoring without a SIEM, configure AWS CloudWatch alarms on IAM credential usage anomalies, enable GCP API key restriction audit logs, and set GitHub webhook alerts for unexpected repository pushes from bot accounts. Deploy osquery with a query polling outbound network connections every 60 seconds: SELECT pid, name, remote_address, remote_port FROM process_open_sockets WHERE remote_address NOT IN ();

**Evidence:** Capture the post-patch agent configuration state as a verified baseline: export Cursor IDE settings.json, GitHub Copilot policy settings from the GitHub organization security page, and Salesforce Agentforce connected app OAuth scope lists. Document the network baseline by capturing 48 hours of outbound connection logs from developer workstations after agents are re-enabled under patched versions, establishing the new normal for comparison during the 30-day monitoring window. Preserve AWS CloudTrail 'LookupEvents' output filtered on the rotated access key IDs to confirm no post-rotation usage of compromised credentials.

**Step 5: Post-Incident — This vulnerability class exposes a systemic gap: AI agents with execution capabilities were granted implicit trust equivalent to authenticated human developers. Implement input sanitization and privilege separation controls for all agentic AI integrations. Establish a review process for any AI tool requesting file system, shell, or secret access before deployment. Map findings against NIST SP 800-53 SI-10 (Information Input Validation) and SA-11 (Developer Testing and Evaluation). Update your AI/ML tool**

**procurement policy to require documented security assessments of agentic capabilities.**

**NIST Phase:** Post Incident

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

**Controls:** NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-10 (Information Input Validation), NIST SA-11 (Developer Testing and Evaluation), NIST RA-3 (Risk Assessment), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

**Compensating:** Document the lessons-learned findings specific to this incident: which AI agent had the broadest permission scope, which secret stores were accessible, and which repository contexts were processed without human review. Write a one-page AI Agent Security Standard requiring that any tool mapping to MITRE ATT&CK T1059 (Command and Scripting Interpreter) or T1552 (Unsecured Credentials) execution paths must pass a documented privilege review before production deployment. For input sanitization without commercial tooling, implement a pre-commit hook using 'detect-secrets' (pip install detect-secrets) that blocks commits containing patterns matching prompt injection payloads (e.g., 'ignore previous instructions', 'system prompt:', 'SYSTEM:') in files processed by AI agents. Add a GitHub branch protection rule requiring human code review for any PR where the committer is a bot identity (github-actions[bot], copilot-swe-agent).

**Evidence:** Compile the complete forensic timeline for the lessons-learned report: first AI agent activation date per tool, permission scope at time of vulnerability disclosure, date each agent processed untrusted external content (pull requests, third-party repos, user-submitted issues), and the window during which prompt injection exploitation was plausible. Map each affected tool to its MITRE ATT&CK technique: T1059 (Command and Scripting Interpreter) for shell execution by Gemini CLI and Anthropic Claude Code, T1552.001 (Credentials In Files) for .env and ~/.aws/credentials reads, T1567 (Exfiltration Over Web Service) for secret exfiltration via the AI tool's outbound API channel. Retain all GitHub Actions run logs, Sysmon captures, and network flow data for a minimum of 12 months to support any downstream regulatory or legal review.

## Detection Guidance

Focus detection on behavioral anomalies from AI agent processes rather than signature-based IOCs, as no public IOCs are confirmed for this CVE. Key signals: (1) Shell or script interpreter processes (bash, python, powershell) spawned as child processes of IDE or AI agent executables, map to T1059. (2) File reads targeting credential paths: ~/.aws/credentials, ~/.ssh/id_rsa, .env, *.pem, *_key files, OS keychain access, map to T1552.001. (3) Unexpected outbound HTTP/HTTPS connections from developer workstations or CI/CD runners to domains not in your approved list, particularly following repository clone or code review actions, map to T1071. (4) GitHub Actions workflow runs with anomalous step execution, particularly steps not matching the committed workflow YAML. (5) Exfiltration indicators: large base64-encoded blobs in process arguments or network payloads, map to T1027, T1560. Query EDR telemetry for parent-child process chains where the parent is a known AI agent binary. No confirmed IOCs (IPs, domains, hashes) are available from the source data for this CVE.

## Framework Mappings

### MITRE-ATTACK

- **T1106** — Native API
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1059** — Command and Scripting Interpreter
- **T1134** — Access Token Manipulation
- **T1565** — Data Manipulation

- **T1552.001** — Credentials In Files
- **T1574** — Hijack Execution Flow
- **T1027** — Obfuscated Files or Information
- **T1195** — Supply Chain Compromise
- **T1560** — Archive Collected Data
- **T1546** — Event Triggered Execution
- **T1552** — Unsecured Credentials
- **T1059.004** — Unix Shell
- **T1190** — Exploit Public-Facing Application
- **T1071** — Application Layer Protocol
- **T1059.006** — Python

### NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **SA-9** — External System Services
- **SR-2** — Supply Chain Risk Management Plan
- **SR-3** — Supply Chain Controls and Processes
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **CA-7** — Continuous Monitoring
- **SI-10** — Information Input Validation

### OWASP-TOP10-2021

- **A03:2021** — Injection

### CIS-V8

- **2.5** — Allowlist Authorized Software
- **16.10** — Apply Secure Design Principles in Application Architectures

### ISO-27001-2022

- **A.8.26** — Application security requirements

## MITRE ATT&CK Mapping

| Technique ID | Technique Name | Tactic |
|---|---|---|
| T1106 | Native API | Execution |
| T1195.001 | Compromise Software Dependencies and Development Tools | Initial-Access |
| T1059 | Command and Scripting Interpreter | Execution |
| T1134 | Access Token Manipulation | Defense-Evasion |
| T1565 | Data Manipulation | Impact |
| T1552.001 | Credentials In Files | Credential-Access |
| T1574 | Hijack Execution Flow | Persistence |
| T1027 | Obfuscated Files or Information | Defense-Evasion |
| T1195 | Supply Chain Compromise | Initial-Access |
| T1560 | Archive Collected Data | Collection |
| T1546 | Event Triggered Execution | Privilege-Escalation |
| T1552 | Unsecured Credentials | Credential-Access |
| T1059.004 | Unix Shell | Execution |
| T1190 | Exploit Public-Facing Application | Initial-Access |
| T1071 | Application Layer Protocol | Command-And-Control |
| T1059.006 | Python | Execution |

## Sources

| Source | URL | Tier |
|---|---|---|
| **Security News** | https://thehackernews.com/2026/04/google-patches-antigravity-ide-fl... | T3 |
| **CVE-2026-21520 Detail - NVD** | https://nvd.nist.gov/vuln/detail/CVE-2026-21520 | T1 |
| **CVE-2026-21520: Copilot Studio Information Disclosure - SentinelOne** | https://www.sentinelone.com/vulnerability-database/cve-2026-21520/ | T3 |
| **Vulnerability Details : CVE-2026-21520 - Copilot Studio** | https://www.cvedetails.com/cve/CVE-2026-21520/ | T3 |

| Source | URL | Tier |
|---|---|---|
| **CVE-2026-21520 - CVE Record** | https://www.cve.org/CVERecord?id=CVE-2026-21520 | T3 |
| **Microsoft Security Advisory** | https://msrc.microsoft.com/update-guide/vulnerability/CVE-2026-21520 | T1 |



Generated 2026-04-21 13:39 UTC by TJS Security Command Center