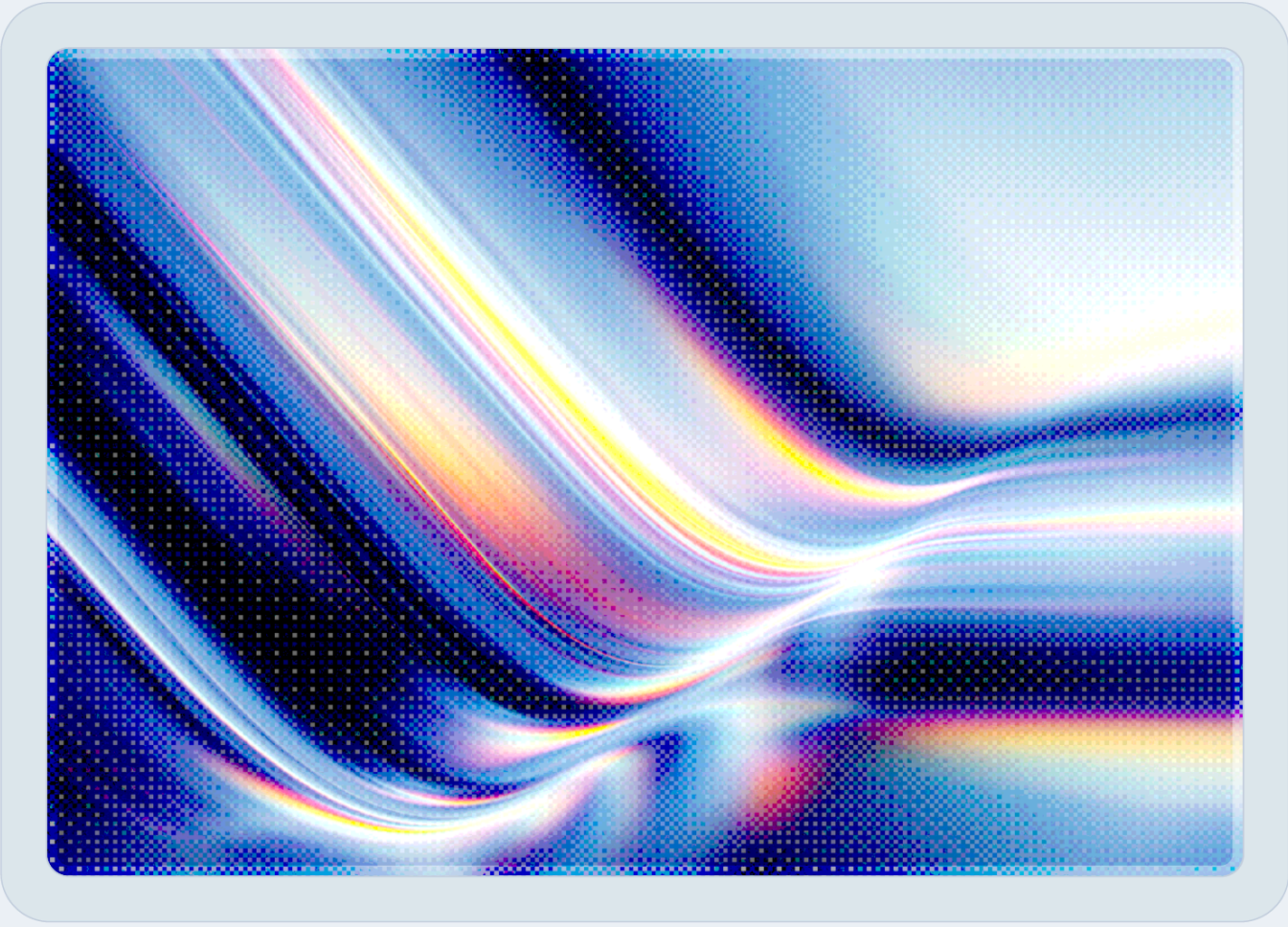
CSAI Foundation | Cloud Security Alliance

AI Coding Agent Sandbox Escapes: The Trust Handoff Flaw

How Cursor, Codex, Gemini CLI, and Antigravity Were Bypassed via Delayed-Trust File Writes

2026-07-22

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

Security researchers at Pillar Security spent a week publishing sandbox-escape findings against four widely used AI coding agents — Cursor, OpenAI's Codex CLI, Google's Gemini CLI, and Google's Antigravity — under the banner "The Week of Sandbox Escapes" [1][2]. None of the seven disclosed issues broke the sandbox itself; each instead exploited a gap between what the sandbox restricts and what a trusted component *outside* the sandbox later reads, runs, or scans. Pillar's researchers describe the pattern succinctly: the agent "stays inside the box and follows every rule," but writes a file — a hook configuration, a virtual environment interpreter, a Git config entry, a task definition — that some other, unsandboxed tool consumes after the agent's turn ends [2]. Cursor's workspace-controlled hook execution flaw, tracked as CVE-2026-48124 with a CVSS score of 8.5, was fixed in version 3.0.0 [3][4]. OpenAI patched a related Codex CLI allowlist bypass, nicknamed "GitPwned," in version 0.95.0 and paid a bug bounty for it [9]. Google classified its two Antigravity-specific findings as "other valid security vulnerabilities," downgraded their severity as difficult to exploit, and had not shipped fixes for either at the time of Pillar's disclosure; the company likewise declined to treat its own exposure in the shared Docker-socket finding as a vulnerability requiring a fix, a position OpenAI also took for Codex CLI's exposure to that same finding [1][2][6]. Taken together, the findings indicate that command-execution sandboxes built for AI coding agents are being defeated not by breaking isolation directly, but by manipulating what happens to an agent's output once it leaves the sandbox — a distinction this research suggests current vendor threat models may not yet consistently account for.

Background

AI coding agents such as Cursor, Codex CLI, Gemini CLI, and Antigravity are designed to operate autonomously inside a workspace: reading files, running shell commands, editing code, and interacting with developer tooling with only intermittent human approval. Because these agents routinely ingest untrusted content — a cloned repository, a dependency's README, a pull request diff, an issue title — vendors have converged on sandboxing as the primary defense against prompt injection turning into arbitrary code execution. The sandbox is meant to constrain what the agent itself can do: which files it can write, which commands it can run, which system calls it can make.

Pillar Security's research, conducted by Eilon Cohen, Dan Lisichkin, and Ariel Fogel and published as a series of daily posts starting July 20, 2026, tested that assumption against four widely used agents [1][2]. Rather than attempting to break out of the sandbox's process, container, or OS-level restrictions directly, the researchers looked at the boundary the sandbox does not police: the point where an agent-authored artifact is later consumed by something else. As the researchers put it, "the agent did not need to break the sandbox directly. It only had to write something that a trusted component outside the sandbox would later run" [2]. That "something else" varied by product — an IDE hook engine, a Python extension, a Git subsystem, a task runner, a Docker daemon — but the shape of the problem was the same across all four tools: sandboxes govern the agent's actions, not the downstream effects of the files the agent produces.

This is not an isolated pattern specific to Pillar's disclosures. Earlier in 2026, Cato Networks separately disclosed a different pair of Cursor sandbox flaws — CVE-2026-50548 and CVE-2026-50549, both rated 9.8 on CVSS 3.1 and collectively named "DuneSlide" — in which prompt-injected commands manipulated working-directory and symlink-validation logic to overwrite the sandbox helper itself and write outside the intended workspace boundary [12]. Cato reported those issues in February 2026; Cursor initially rejected them as out of scope for its threat model before reopening and fixing them in version 3.0, with CVE numbers assigned in June [12]. The DuneSlide flaws attacked the sandbox's own enforcement logic rather than exploiting a

downstream trust handoff, but the outcome — a single, unapproved prompt leading to code execution as the user — was the same. Read together, the two disclosures suggest that Cursor's sandbox in particular, and command-execution sandboxes for AI coding agents more broadly, have accumulated multiple independent classes of escape, discovered by unrelated research teams within months of each other; the evidentiary weight of both disclosures is concentrated heavily in a single vendor's product, since DuneSlide is Cursor-only and four of Pillar's seven findings are Cursor findings.

Security Analysis

The Trust Handoff Pattern

Pillar's central finding is that a sandbox is only as strong as the assumptions made by everything downstream of it. Four recurring failure modes ran through the seven disclosed issues. First, denylist-based sandboxes — such as the macOS Seatbelt profile used by Antigravity — have so far failed to enumerate the full complexity of an operating system's attack surface, leaving specific primitives reachable that the denylist's authors did not anticipate [2][5]. Second, several agents treat project-local configuration files as trusted infrastructure rather than untrusted workspace content: a `.claude/settings.local.json` hook definition [2][3][4][10], a `.vscode` task file [2][11], and a Python virtual environment's interpreter binary [2][7] can all be written by the agent inside the sandbox and then executed, unsandboxed, by a separate tool that assumes anything already present in the workspace was placed there deliberately by the user. Third, command allowlists that key off a command's *name* rather than its actual arguments proved brittle; Codex CLI's allowlist trusted `git show` as inherently read-only, without accounting for flags like `--output` that let the command write arbitrary file content [9]. Fourth, agents that can reach a privileged local daemon — most notably the Docker socket — inherit whatever authority that daemon has, regardless of how tightly the agent's own process is contained, since launching a container through the daemon sidesteps the agent sandbox entirely [1][2][6].

Vulnerability Findings by Product

The table below summarizes the seven issues disclosed across the four affected products, drawing on Pillar Security's published write-ups and the corresponding vendor advisories.

Product	Vendor	Flaw	Mechanism	Advisory / CVE	Status
Cursor	Anysphere	Workspace-controlled hook execution	A `.claude/settings.local.json` "Stop" hook ran an unsandboxed local command automatically when an agent turn ended, without requiring user approval	CVE-2026-48124 / GHSA-pc9j-3qc2-95wv, CVSS 8.5 [3][4][10]	Fixed in Cursor 3.0.0
Cursor	Anysphere	Tampered Python	Agent, operating in Auto-Run Sandbox mode, replaced a virtualenv's Python executable; a separate, unsandboxed IDE	GHSA-p9g2-	Fixed in Cursor 3.1.2

Product	Vendor	Flaw	Mechanism	Advisory / CVE	Status
		virtual environment	extension later invoked that interpreter with the user's privileges	cr55-cw9c [2][7]	
Cursor	Anysphere	Git metadata indirection	Agent created an alternate Git metadata directory that did not use the literal name `.git` and set `core.fsmonitor` to an attacker-controlled script; Cursor's unsandboxed Git extension auto-detected the repository and executed the script	No CVE/GHSA identifier found in source material [2][8]	Patch status not specified in available source material
Cursor, Codex CLI, Gemini CLI	Anysphere, OpenAI, Google	Docker socket / VirtioFS exposure	A sandboxed agent could launch a privileged container through the host's Docker socket and mount the VirtioFS share, gaining read/write access to the user's home directory outside the sandbox boundary	GHSA-v4xv-rqh3-w9mc (Cursor-specific advisory) [2][6]	Fixed in Cursor 3.0.0 for Anysphere's exposure; OpenAI and Google did not fix their own exposure, classifying it respectively as informational and as an already-documented, out-of-scope risk
Codex CLI	OpenAI	"GitPwned" allowlist bypass	The safe-command allowlist trusted `git show` by name; `git show --output=./.git/config` wrote a malicious external diff-tool entry that executed with full user privileges the next time the victim ran `git diff`	CVSS 8.6; no confirmed CVE/GHSA identifier for this specific finding [9]	Fixed in v0.95.0; bug bounty paid
Antigravity	Google	macOS Seatbelt "allow-default" gap	Sandbox denylist profile left OS-level primitives reachable that a purpose-built allowlist would have excluded	No CVE assigned [1][2][5]	Classified by Google as an "other valid security

Product	Vendor	Flaw	Mechanism	Advisory / CVE	Status
					vulnerability"; not patched as of disclosure
Antigravity	Google	VS Code task-configuration execution	Agent wrote a `.vscode` task definition inside the sandbox; the host later executed that task outside the product's Secure Mode boundary	No CVE assigned [1][2][11]	Classified by Google as an "other valid security vulnerability"; not patched as of disclosure

Sources: [1][2][3][4][5][6][7][8][9][10][11]

Of the seven issues, at least four received a confirmed vendor patch: Cursor's hook-execution and virtual-environment findings, Cursor's own exposure in the shared Docker-socket/VirtioFS finding, and OpenAI's fix for GitPwned in Codex CLI. The Git-metadata-indirection finding's patch status is not specified in the available source material. The remaining issues reflect vendor declination rather than remediation: Google did not fix either Antigravity-specific finding, classifying both as "other valid security vulnerabilities" it considered difficult to exploit, and neither OpenAI nor Google fixed their own exposure to the Docker-socket finding — OpenAI classified its exposure as informational, while Google described the underlying risk as already documented in its product materials and therefore out of scope [6]. That pattern, a single technical finding met with divergent responses depending on which vendor's product was affected, is itself a data point about how sandbox-escape severity gets triaged when the exploit path runs through a downstream trust handoff rather than a conventional memory-safety or privilege-escalation bug; a reviewer scoring exploitability primarily on the sandbox's own boundary may undercount risk that instead depends on what a second, unsandboxed component does with the sandbox's output.

Why This Class of Bug Persists

The consistent thread across both the Pillar disclosures and the earlier DuneSlide findings is that in each case the sandbox boundary itself held, while the downstream component consuming the sandbox's output was not designed to re-validate trust. A file written inside a sandbox is not, by itself, dangerous; it becomes dangerous only when something with more authority than the sandbox reads it and acts on its contents without re-validating trust. IDE extensions, Git subsystems, task runners, and container daemons were generally not designed with the assumption that their inputs might originate from an autonomous agent operating on attacker-controlled content, and so they extend to agent-written files the same implicit trust they would extend to a human developer's own edits. As Pillar's researchers frame the underlying threat model: "If an agent gets to write the future inputs of systems, it was never sandboxed in the first place" [2]. That framing reorients the security question away from "can the agent escape its sandbox" toward "what, outside the sandbox, will eventually trust something the agent wrote" — a question that spans IDE architecture, developer tooling, and CI/CD pipelines well beyond the agent runtime itself.

Recommendations

Immediate Actions

Organizations running Cursor, Codex CLI, Gemini CLI, or Antigravity in agentic or auto-run modes should update immediately to the patched versions identified above — Cursor 3.0.0 or later (3.1.2 for the virtual environment issue), and Codex CLI 0.95.0 or later — and should audit whether any workspace already contains a `.claude/settings.local.json` hook, a modified virtualenv interpreter, a non-standard Git metadata directory, or a `.vscode` task definition that was not authored by a known human contributor. Teams using Antigravity, or using Codex CLI or Gemini CLI with Docker Desktop reachable from the agent's execution environment, should treat those exposures as unresolved until the responsible vendor ships a fix, and should avoid running these agents against untrusted repositories or dependencies without additional host-level controls in the interim.

Short-Term Mitigations

Beyond patching, security teams should inventory every unsandboxed component that a coding agent's sandbox output can reach — Git hooks and config files, IDE extensions that read project-local interpreter paths, task runners, package managers, and any daemon socket (Docker or otherwise) mounted into or reachable from the agent's execution environment. Where possible, that reach should be reduced: mount Docker sockets only when strictly necessary and never alongside agent auto-run modes, configure IDE extensions to require explicit confirmation before executing project-local interpreters or task definitions that were modified during an agent session, and apply the same code-review scrutiny to agent-authored configuration files that a team would apply to agent-authored source code. Command allowlists that key on command name alone should be re-evaluated for commands, like `git show`, that support write-capable flags; the safer pattern is to model the full argument surface of a command before classifying it as read-only.

Strategic Considerations

At a program level, this research reinforces that sandbox isolation and least-privilege design are necessary but not sufficient controls for agentic coding tools: the trust boundary that matters is not just around the agent's process, but around every downstream consumer of the agent's output. Security architects evaluating or deploying coding agents should require vendors to document not only their sandbox's isolation technology, but also the full set of unsandboxed components — extensions, daemons, hooks, task runners — that the agent's sandbox output eventually reaches, and should ask how each of those components validates trust before acting on agent-authored content. This research also illustrates how the same underlying finding can receive materially different severity treatment depending on which vendor triages it; architects should press vendors on their reasoning when a bug that routes through a second, unsandboxed component is scored lower than a bug that breaks the sandbox boundary directly, since the disclosures in this note show that routing can be the entire exploit rather than a mitigating factor.

CSA Resource Alignment

CSA's AI Controls Matrix (AICM) v1.1 provides the control vocabulary most directly applicable to this pattern of findings [13]. Its Infrastructure Security domain speaks to exactly the isolation-tier question these disclosures raise — whether an agent's sandbox is matched to the adversarial assumptions of the content it processes — while its Application and Interface Security domain covers the input-validation and command-handling failures that let a Docker socket, a Git allowlist, or a hook engine be

manipulated into acting on attacker-controlled content. Evaluating a coding agent deployment against AICM's Identity & Access and Logging & Monitoring domains also surfaces a likely audit gap in this pattern of findings: none of the vendor advisories or Pillar Security write-ups reviewed for this note describe an alert being generated when an unsandboxed component consumed an agent-written file, suggesting that these products did not yet treat that handoff as a trust boundary worth monitoring.

The MAESTRO agentic AI threat modeling framework is the complementary tool for mapping where these failures actually occur within an agent's architecture [14]. MAESTRO's layered model separates an agent's foundation model and orchestration logic from its deployment and infrastructure layer and from the broader agent ecosystem it interacts with; applied to this research, every one of Pillar Security's seven findings sits at the boundary between the deployment/infrastructure layer (the sandbox) and the ecosystem layer (the IDE extension, Git subsystem, task runner, or Docker daemon that ultimately trusted the agent's output). That framing reinforces this note's central point: securing the sandbox layer alone, without threat-modeling everything the sandbox's output subsequently reaches, leaves exactly the gap these disclosures exploited.

References

[1] Bill Toulas. "Cursor, Codex, Gemini CLI, Antigravity hit by sandbox escapes." BleepingComputer, July 21, 2026.

[2] Eilon Cohen, Dan Lisichkin, and Ariel Fogel. "The Week of Sandbox Escapes." Pillar Security, July 20, 2026.

[3] SentinelOne. "CVE-2026-48124: Cursor Code Editor RCE Vulnerability." SentinelOne Vulnerability Database, 2026.

[4] Cursor (Anysphere). "Cursor Desktop sandbox escape via Claude hook configuration (GHSA-pc9j-3qc2-95wv)." GitHub Security Advisories, 2026.

[5] Pillar Security. "The Week of Sandbox Escapes Day 1: Escaping Antigravity's allow-default seatbelt." Pillar Security Blog, July 2026.

[6] Pillar Security. "The Week of Sandbox Escapes Day 2: One Docker socket to rule them all." Pillar Security Blog, July 2026.

[7] Pillar Security. "The Week of Sandbox Escapes Day 3: The sandbox let me edit a venv, and something else ran it." Pillar Security Blog, July 2026.

[8] Pillar Security. "The Week of Sandbox Escapes Day 4: Git directories do not have to be called .git." Pillar Security Blog, July 2026.

[9] Pillar Security. "The Week of Sandbox Escapes Day 5: GitPwned: Allowlist to RCE." Pillar Security Blog, July 2026.

[10] Pillar Security. "The Week of Sandbox Escapes Day 6: The hook was already in the workspace." Pillar Security Blog, July 2026.

[11] Pillar Security. "The Week of Sandbox Escapes Day 7: A time bomb in .vscode: bypassing Antigravity's Secure Mode." Pillar Security Blog, July 2026.

[12] The Hacker News. "Critical Cursor Flaws Could Let Prompt Injection Escape Sandbox and Run Commands." The Hacker News, July 2026.

[13] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, June 22, 2026.

[14] Ken Huang / Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." Cloud Security Alliance, February 6, 2025.

