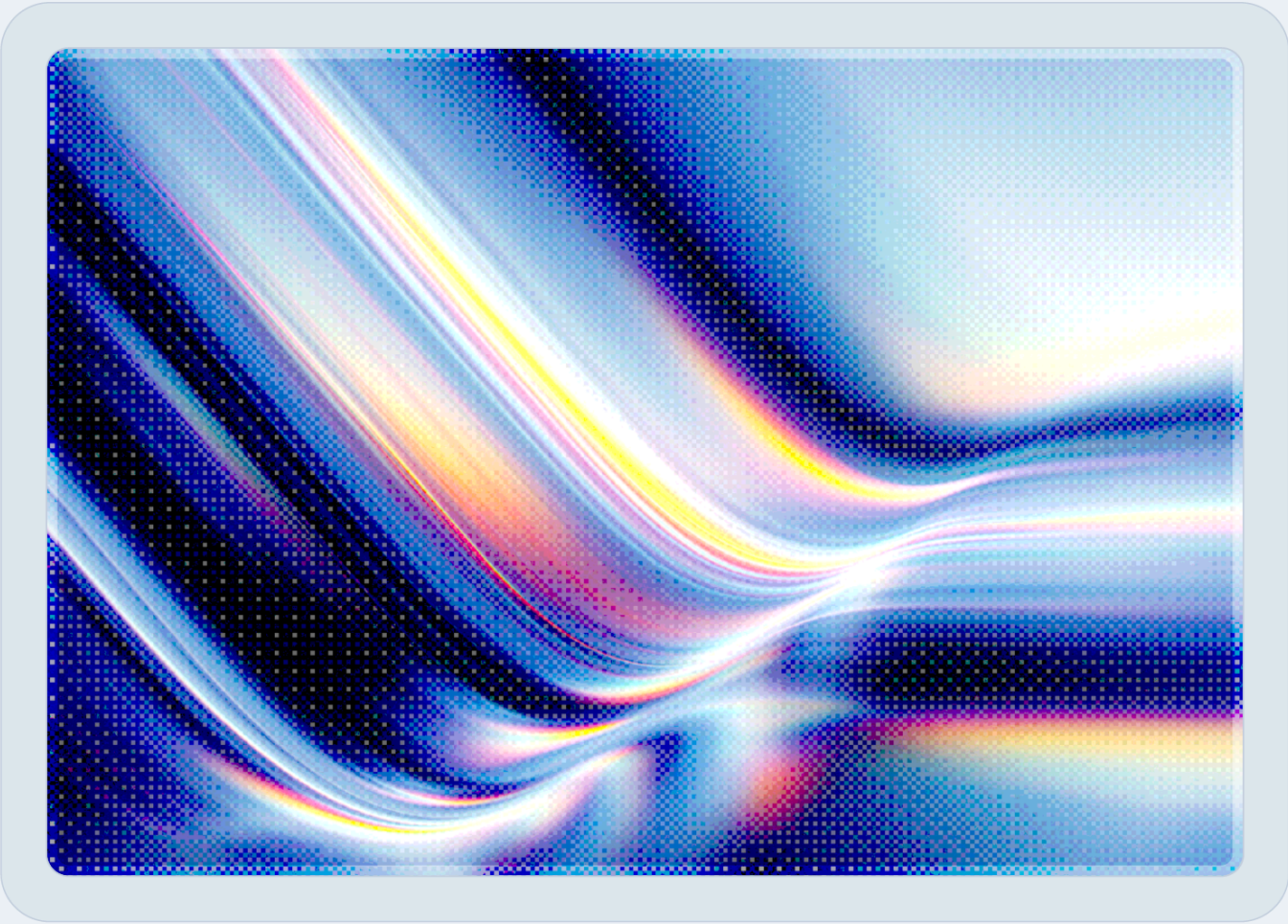
CSAI Foundation | Cloud Security Alliance

SharedRoot: A Sandbox Escape in Claude Cowork's VM

2026-07-24

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- Security researcher Oren Yomtov of Accomplish AI disclosed "SharedRoot," an attack chain that lets an unprivileged agent session inside Anthropic's Claude Cowork escape its Linux virtual machine and read or write arbitrary files on the host Mac — including SSH keys and cloud credentials — with no permission prompt to the user [1][2].
- The chain does not exploit a single bug in Cowork. It combines a permissive guest configuration (unprivileged user namespaces, a seccomp policy that allows netlink traffic-control calls) with a generic Linux kernel privilege-escalation bug, CVE-2026-46331 ("pedit COW"), to obtain root inside the guest, and then rides a writable, entire-host filesystem mount that Cowork exposes at `/mnt/.virtiofs-root` [2][3].
- Anthropic classified the disclosure as "Informative" and has not issued a fix for local execution; Cowork's current default now routes agent execution to the cloud rather than a local VM, which sidesteps this specific escape path for users who accept the default, but local execution — the mode the vulnerability targets — remains unpatched [1][2].
- SharedRoot is not an isolated incident. A separate researcher group reported a distinct Windows-side Cowork escape to Anthropic in March 2026 — publicly disclosed only in July 2026, around the same time as SharedRoot — involving DLL sideloading to defeat signature verification, a resume-flag bypass in the VM service, and an egress-allowlist override; Anthropic likewise declined to treat it as a qualifying vulnerability on the grounds that it requires pre-existing local code execution [4][5] — the same precondition that describes exactly what an agentic coding sandbox exists to contain.
- The pattern echoes a broader 2026 trend documented in CSA's own research: deployment-infrastructure boundaries that are supposed to survive an agent achieving root inside its sandbox are proving to be the recurring point of failure across multiple AI coding-agent products, not a Cowork-specific defect [8].

Background

Claude Cowork is Anthropic's agentic coding product, which gives a Claude-driven agent the ability to browse, edit, and execute code against folders a user connects to it. Because an agent processing untrusted repository content, web pages, or file contents is a plausible target for prompt injection and

other adversarial input, Anthropic isolates each Cowork session inside its own environment rather than running it directly on the user's machine. Anthropic has described this architecture publicly: claude.ai runs in ephemeral gVisor containers with no local execution at all, Claude Code runs a "human-in-the-loop" OS-level sandbox (Seatbelt on macOS, bubblewrap on Linux) directly on the developer's machine, and Cowork — the product with the broadest intended autonomy — runs each session inside a full virtual machine using the host's own hypervisor (Apple's Virtualization framework on macOS, Microsoft's Hyper-V-based HCS on Windows), complete with its own kernel, filesystem, and process table [6].

That VM boundary is the control the SharedRoot research targets. On July 23, 2026, Accomplish AI published a technical writeup describing how an agent running inside a fresh, unmodified Cowork session — triggered by nothing more than a single short message and a connected folder — could escalate from its ordinary session-user privileges to root inside the guest VM, and from there reach the entire host Mac filesystem [2]. The Hacker News covered the disclosure the same week, noting that Cowork's local-execution mode had reportedly reached roughly 500,000 macOS users before Anthropic shifted the product's default toward cloud-based execution [1]. The vulnerability's practical significance is straightforward: any Mac running a locally-executed Cowork session that processes attacker-influenced content — a malicious README, a poisoned pull request, a crafted web page the agent is asked to summarize — is a candidate for silent, credential-level compromise of the host, not just the connected project folder.

This is also the second publicly reported full sandbox escape against Cowork in 2026. A cybersecurity startup led by former Mandiant founder Kevin Mandia privately reported a distinct Windows-specific escape chain to Anthropic on March 20, 2026; Anthropic rejected it four days later, and the chain did not become public until SiliconANGLE and SC Media covered it on July 1 and July 2, 2026 — roughly three weeks before SharedRoot, not months apart [4][5]. That chain first sideloaded malicious code into the signed `claude.exe` process to defeat signature verification on the `CoworkVMService` named pipe, then exploited a resume-flag manipulation in that component to skip the creation of an unprivileged guest account, and separately overrode an approved-domain network allowlist with a wildcard on a per-command basis [4][5]. Anthropic disputed that report's severity on the grounds that exploiting it required an attacker to already have local code execution on the host; its stated rationale for closing SharedRoot as "Informative" was different — that the CVE fell within the bug-bounty program's 30-day publication window and that the proposed hardening measures amounted to defense-in-depth rather than standalone vulnerabilities [1][2]. The two dismissals rest on different stated justifications, but they share a common effect: Cowork's VM is meant to contain an agent that is, by definition, already executing arbitrary instructions inside the guest, so treating routine guest-configuration hardening as optional — for whichever stated reason — leaves the actual controlling boundary unaddressed once an agent has any code execution inside its sandbox.

Security Analysis

The SharedRoot chain begins from the position any Cowork agent session normally occupies: an unprivileged Linux user inside the guest VM, constrained by a seccomp filter that Accomplish AI characterizes as a "default-allow denylist" rather than a strict default-deny policy [2]. From that position, the session user calls `unshare` to create a new Linux user namespace — a facility the guest kernel permits unprivileged users to use by design — and within that namespace the calling process appears as root and is granted `CAP_NET_ADMIN` [2][3]. Because the seccomp policy permits netlink-socket operations, the now-capable process can configure the kernel's traffic-control (`tc`) subsystem, and doing so triggers the guest kernel to autoload a module it would not otherwise have loaded: `act_pedit`, the packet-editing action handler [2].

Loading `act_pedit` puts the exploitable code on the table. CVE-2026-46331, nicknamed "pedit COW" and disclosed as a generic Linux local-privilege-escalation bug in mid-June 2026, arises because the kernel's `tcf_pedit_act()` function calculates the writable copy-on-write range for a packet edit before the edit's final runtime offset is known; when that offset resolves to a location outside the range the kernel privately copied, the resulting write lands in the shared page cache instead of a private copy [7]. In practice this lets an unprivileged local user corrupt the cached, on-disk image of a root-owned binary without ever writing to disk, planting a malicious payload that persists only in memory [7]. SharedRoot uses this to poison the page cache of a root-owned helper binary that Cowork's privileged host daemon, `coworkd`, later re-executes without enforcing `NoNewPrivs` — at which point the poisoned binary runs with root privileges inside the guest, and the session user has gone from unprivileged to guest-root [2][3].

Guest-root is where the escape becomes a host compromise, because of a separate architectural choice: Cowork mounts the entire host filesystem, read-write, at `/mnt/.virtiofs-root` inside the guest via VirtioFS, intending that mount to be reachable only by root within the guest [2][3]. That intention is exactly what guest-root privileges defeat. Once the agent process holds guest-root, it can read and write any file the logged-in Mac user could access — SSH keys, AWS and other cloud CLI credentials, browser data, source repositories, and arbitrary application configuration — none of it limited to the specific folder the user connected to the Cowork session in the first place [2][3]. Anthropic's own description of Cowork's containment model states that two of its six isolation mechanisms are "enforced outside the guest kernel" specifically so they would survive an agent achieving root inside the VM, and that filesystem access to user-connected folders is scoped through distinct read-only, read-write, and read-write-no-delete modes with symlink resolution performed before path validation [6]. SharedRoot's target, the blanket `/mnt/.virtiofs-root` mount, sits outside that more carefully scoped per-folder model — it is a broader, less-restricted surface that guest-root access reaches directly.

It is worth being precise about what is and is not a Cowork-specific defect here. CVE-2026-46331 is a general Linux kernel bug affecting a wide range of kernel versions (from v5.18 up to a v7.1 release candidate, fixed in v7.1-rc7), and it has nothing to do with Anthropic's product; any Linux host or guest that allows unprivileged user namespaces and has not yet patched the kernel is exploitable in the same way, independent of Cowork [7]. What is specific to Cowork's design is the combination of choices that turns a guest-local privilege escalation into a full host breach: permitting unprivileged namespace creation and netlink/`tc` configuration by default, allowing autoload of an unused kernel module, re-executing a root-owned binary without `NoNewPrivs`, and exposing the complete host filesystem as a single writable mount rather than scoping guest access to only the folders a user actually connected. Accomplish AI's own framing is instructive: any one of four independent hardening changes — disabling unprivileged namespaces, tightening the seccomp policy, blocking module autoload, or scoping the host mount — would have broken the chain on its own [2][3]. That a fully generic kernel bug could be converted into credential-level host compromise through a chain of otherwise-ordinary guest configuration choices is the central lesson for anyone evaluating VM-based agent sandboxes, not merely a story about one unpatched CVE.

Recommendations

Immediate Actions

Organizations that have deployed Claude Cowork should determine whether affected endpoints are running agent sessions in local VM execution mode rather than the newer cloud-execution default, since the local mode is the one SharedRoot targets and no patch for it has been issued as of this writing [1][2]. Where local execution is in use, security teams should avoid connecting folders that contain SSH keys, cloud CLI credential directories, or other secrets to a Cowork session until Anthropic ships a fix, and should treat any Mac that has run a local Cowork session against untrusted repository or web content as a candidate for credential rotation. Teams running any self-managed Linux VM or container infrastructure that permits unprivileged user namespaces should independently confirm their kernels are patched against CVE-2026-46331, since the underlying bug is unrelated to Cowork and affects a broad range of kernel versions [7].

Short-Term Mitigations

Vendors building VM- or container-based sandboxes for coding agents — and enterprises operating self-hosted equivalents — should apply the hardening measures the SharedRoot research identifies as sufficient to break this specific chain: disabling unprivileged user namespaces where agent workloads do

not require them, moving seccomp policy from a denylist to a default-deny model that blocks `unshare`, `setns`, `clone3`, and `AF_NETLINK` for session users, preventing autoload of kernel modules such as `act_pedit` that ordinary agent workloads never need, and enforcing `NoNewPrivs` on any privileged host process that re-executes a binary on behalf of a lower-privileged caller [2][3]. More broadly, no VM-based agent sandbox should expose the complete host filesystem as a single writable mount; guest access should be scoped strictly to the folders a user explicitly connects, using the least-privileged mount mode the workflow allows, consistent with the read-only/read-write/read-write-no-delete distinctions Anthropic already documents for connected folders elsewhere in Cowork's architecture [6].

Strategic Considerations

The recurrence of full sandbox-escape disclosures against the same product within a matter of weeks — a Windows-specific escape and a macOS-specific escape, both publicly disclosed within about three weeks of each other in mid-to-late 2026, closed on different stated grounds but neither resulting in a shipped fix for local execution — suggests enterprises should not treat a vendor's classification of a report as "Informative" as evidence that a given VM boundary is sound, particularly when the underlying precondition in both cases is simply that the agent already has code execution inside its own sandbox. Security teams evaluating agentic coding tools for use with sensitive repositories or credentials should require vendors to document their isolation architecture in enough technical detail to assess independently — mount scope and write permissions, the guest privilege model, and which specific layers of the containment stack are designed to survive an agent reaching root inside the guest — rather than relying on marketing language about "sandboxing" alone. Procurement and security-review processes for agentic AI tools should treat guest-root escalation as a critical-severity finding by default, regardless of whether the reporting vendor's own bug-bounty triage agrees, given that the entire value proposition of an agent sandbox is containing code the agent itself is expected to run.

CSA Resource Alignment

SharedRoot fits a pattern CSA has already documented independently. CSA's research note "Claw Chain: Four CVEs Enable Full AI Agent Compromise" [8] analyzed a different AI coding agent's sandbox and found time-of-check/time-of-use race conditions that let an attacker with sandbox code execution redirect filesystem writes and substitute symbolic links to reach paths outside the intended mount root — a different mechanism than SharedRoot's kernel-privilege-escalation chain, but the same failure category. Both vulnerabilities map to what CSA's MAESTRO framework designates Layer 4, Deployment and Infrastructure, the layer agentic AI architectures depend on to enforce execution boundaries once a

model or its tooling has been compromised or manipulated [8][9]. CSA's "Agentic AI Threat Modeling Framework: MAESTRO" addresses container- and orchestration-level compromise at Layer 4 and treats privilege escalation as a threat that can chain across layers; SharedRoot is a concrete instance of exactly that chaining, from an agent's in-sandbox execution up through a deployment-infrastructure boundary, converting an in-sandbox compromise into a full host breach [9]. Finally, the incident bears directly on the Infrastructure Security and Application and Interface Security domains of CSA's "AI Controls Matrix (AICM) v1.1," whose control objectives call for auditing hypervisor and mount configurations, least-privilege guest access scoping, and sandboxing of custom code execution — precisely the controls (namespace restriction, mount scoping, seccomp hardening) whose absence made the SharedRoot chain possible [10]. Enterprises assessing Cowork or comparable agentic coding tools against AICM should treat this incident as a concrete test case for those control objectives rather than a hypothetical.

References

[1] The Hacker News. "Claude Cowork Flaw Could Let AI Agent Escape Its VM and Access Mac Files." July 2026.

[2] Yomtov, Oren / Accomplish AI. "SharedRoot: Escaping the Claude Cowork Sandbox." Accomplish AI Blog, July 23, 2026.

[3] GBHackers. "Claude Cowork Sandbox Escape Flaw Lets Attackers Access SSH Keys and Cloud Credentials." July 2026.

[4] SC Media. "Researchers Detail Attack Chain Escaping Anthropic's Claude Cowork Sandbox." July 2026.

[5] SiliconANGLE. "Armadin Details Full Sandbox Escape in Claude Cowork but Anthropic Disputes Risk." July 1, 2026.

[6] Anthropic. "How We Contain Claude Across Products." Anthropic Engineering Blog, 2026.

[7] The Hacker News. "New Linux pedit COW Exploit Enables Root Access by Poisoning Cached Binaries." June 2026.

[8] Cloud Security Alliance. "Claw Chain: Four CVEs Enable Full AI Agent Compromise." CSA Labs, May 17, 2026.

[9] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 6, 2025.

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