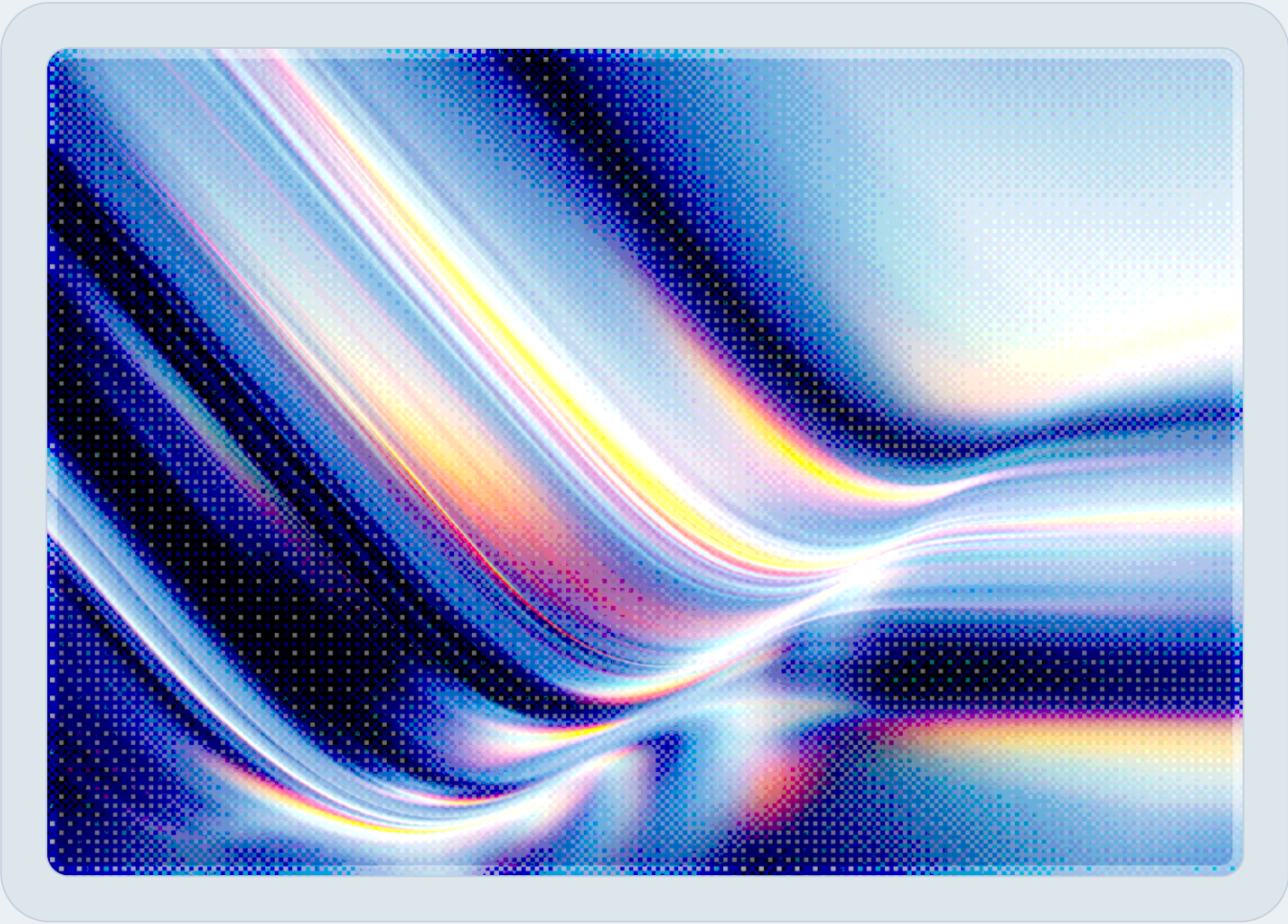
CSAI Foundation | Cloud Security Alliance

Carbonato: Telegram-Controlled AI Agent Hijacks Docker Hosts

2026-09-28

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

Carbonato is a self-propagating botnet that compromises Docker hosts whose daemon API is exposed to the network without authentication, then uses the resulting privileged container access to install an unmodified open-source AI agent framework, Hermes Agent, and repoint it toward attacker goals by swapping out its persona configuration file [1][2][3]. Operators issue instructions through Telegram, and the agent — running under a persona named "GH0ST" — interprets those instructions, writes and executes terminal commands, and reports results back through the same channel, producing an interactive command loop that requires no manual typing of individual commands by the operator [3]. The malicious persona explicitly instructs the agent to prioritize harvesting AI provider API keys ahead of SSH credentials, access tokens, and database contents, naming fourteen large language model providers as collection targets [3]. Researchers at ThreatDown identified the operation through an unauthenticated Docker image registry that had been publicly reachable since May 2026, and recovered operational evidence spanning October 2024 through August 2026, indicating the campaign has been active for nearly two years before public disclosure [2][3]. This case extends a pattern CSA has tracked across several 2026 incidents [5][6]: attackers are not merely using AI tools to write malware faster, they are embedding AI agents directly inside the malware itself, converting each compromised host into an autonomous, instructable asset rather than a static implant.

Background

Docker's daemon exposes a REST API that, by default configuration choices made by many operators, can be bound to a TCP socket on port 2375 without any authentication requirement. Any client that can reach that port can direct the daemon to pull images, create containers, and — critically — launch a container with the host filesystem mounted and elevated privileges, which effectively grants arbitrary code execution on the underlying host. This misconfiguration is well understood and has long been abused by cryptomining and DDoS botnets, but Carbonato adds a distinguishing second stage once it obtains that initial foothold. After using the exposed API to launch a privileged container, it opens a reverse SSH tunnel to a relay server, installs an SSH server seeded with an operator-controlled key, and reports the new deployment to a Telegram chat, giving the operators durable remote access alongside the automated tooling [3][4].

Where earlier Docker-targeting botnets have historically tended to deploy a fixed payload — a miner, a DDoS agent, a generic backdoor — Carbonato instead deploys Hermes Agent, an MIT-licensed, publicly available AI agent framework, and leaves its core code untouched. The only modification is to overwrite Hermes Agent's persona configuration file, SOUL.md, with a thirty-nine-line prompt that assigns the agent a "senior hacker" identity named GH0ST and instructs it to execute any task an operator submits without ethical constraint, maintain persistence on the host, and prioritize the theft of AI API keys [3]. Because the framework itself is not altered, the malicious behavior lives entirely in a configuration file rather than in compiled code, which narrows what static malware signatures alone can reliably catch. Once installed, the agent connects to an LLM gateway operated by the attackers, which was found to advertise 12 of its 27 served models on a free tier, with the remainder presumably dependent on the operators' own or stolen API capacity — a detail that plausibly explains the priority placed on stealing legitimate API keys to subsidize or expand that capacity [3].

Discovery of the campaign traces back to an unauthenticated Docker image registry that had been reachable on the open internet since at least May 2026. ThreatDown researchers used that exposure to recover fifty-nine repositories, two hundred thirty-four image tags, and six hundred five SHA-256-verified image blobs, totaling roughly 4.3 gigabytes of image data and spanning nearly 945,000 indexed files, which allowed reconstruction of the toolkit's evolution over time rather than a single snapshot [3]. As of early September 2026, six of the seven registries the researchers had identified remained operational, suggesting the operators had not yet reacted to the exposure at the time of disclosure [3]. Coverage from The Hacker News and BleepingComputer corroborates the core technical chain — unauthenticated Docker API, privileged container escape, AI-agent-driven post-exploitation over Telegram — and both outlets note that ThreatDown's evidence trail runs from October 2024 through August 2026, indicating sustained, low-visibility operation well before the public writeup [1][2].

Security Analysis

Carbonato's propagation mechanism follows a conventional worm pattern layered on top of the Docker exposure: every five minutes, infected hosts scan neighboring /24 network ranges for additional Docker daemons listening without authentication, and any newly found host is compromised using the same privileged-container technique [1][2][3]. Persistence is redundant, combining cron jobs, systemd timers, rc.local entries, and OpenRC hooks, with some artifacts set with immutable file attributes to resist casual removal, and a watchdog process that re-pulls the implant if its files are deleted [3]. The malware also invests in blending into normal host activity — the core process masquerades as a kernel worker thread (`[kworker/u2:0]`), and containers are named to resemble legitimate services such as `systemd-resolved` , which raises the bar for an administrator doing a casual process listing to notice anything wrong [3].

The command-and-control architecture marks a departure from prior Docker-targeting botnets. Rather than a bespoke C2 protocol, operators rely on Telegram as the interface: newly compromised hosts report in to a specific Telegram chat with host details, and operators subsequently send free-form task instructions through that same channel [3]. Those instructions are handed to the Hermes Agent instance running on the victim host together with the malicious SOUL.md persona, and an LLM interprets the objective, generates the appropriate terminal commands, executes them locally, and returns the output to Telegram — an interactive loop that lets a single operator direct many compromised hosts through natural-language instructions instead of hand-written scripts for each target [3]. This is functionally consistent with the broader 2026 trend of "agentic botnets," in which the traditional bot-herder's fixed command menu is replaced by a general-purpose reasoning agent capable of improvising against whatever it finds on a given host, a pattern also documented in CSA's prior analysis of hallucination-driven malware delivery techniques that similarly convert individual compromises into scalable, low-effort operations for the attacker [5].

Credential and data theft is explicitly prioritized in the malicious persona toward AI provider API keys over other sensitive material, including SSH credentials, access tokens, and database contents [3]. This targeting choice reflects a broader shift already flagged in CSA's analysis of the NadMesh botnet, which found that credential-harvesting campaigns increasingly treat exposed AI and development infrastructure as a gateway into cloud and API-key sprawl rather than an end target in itself [6]. Attribution signals collected by ThreatDown — use of voseo Spanish (a regional dialect common in Central and South America), a Telegram operator handle referencing Costa Rica's country calling code, and infrastructure timestamps consistent with the America/Costa_Rica timezone in a minority of examined configurations (14 of 162, per ThreatDown) — point toward Costa Rica-based operators, though the timezone signal alone is weak given how few configurations showed it, and ThreatDown stops short of linking the campaign to any previously catalogued threat group [3]. Separately, ThreatDown's writeup notes that the same operational infrastructure also supports a trojanized cryptocurrency wallet distribution operation, suggesting Carbonato functions as one product line within a broader, financially motivated toolkit rather than a single-purpose botnet [3].

Recommendations

Immediate Actions

Organizations running Docker should immediately verify whether the daemon's REST API is reachable over the network on TCP port 2375, or any other port, without client certificate authentication, since this single misconfiguration is the entire initial access vector Carbonato relies on. Any host found in this state

should be treated as potentially already compromised, not merely at risk, given the volume of historical exposure ThreatDown recovered from public registries. Incident responders should hunt for the specific indicators published by ThreatDown, including container or image names such as `gh0st/`, `fsociety/`, `netd-svc`, and `system/resolved`; a process disguised as `[kworker/u2:0]`; the environment variables `GH0ST_C2` and `CARBONATO_API_KEY`; and outbound connections to the reported infrastructure, which includes command-and-control addresses 45.79.183.61, 91.99.195.164, and 213.136.79.115, an LLM gateway at 213.136.83.197, and a reverse-tunnel relay at 190.211.124.187 [3]. Any environment where these indicators are found should trigger immediate rotation of all locally stored AI provider API keys, SSH keys, and access tokens, since credential theft — not merely persistence — is the operators' stated priority.

Short-Term Mitigations

Docker daemon APIs should never be exposed to the network without TLS client-certificate authentication; where remote management is required, it should be placed behind a VPN or bastion host rather than a bare TCP listener. Security teams should add detection rules for privileged container creation events that mount the host filesystem, since this single API call is the pivot point from "exposed service" to "full host compromise" in this campaign. Persistence hunting should extend beyond typical cron and systemd checks to include rc.local and OpenRC hooks with immutable file attributes, and any watchdog-style scripts that re-fetch remote payloads on a schedule. Container image registries should likewise require authentication for both push and pull operations; the scale of data ThreatDown was able to recover — nearly 4.3 gigabytes across fifty-nine repositories — underscores how much operational detail an unauthenticated registry can leak to defenders and attackers alike.

Strategic Considerations

Carbonato is best understood as an early instance — technically unremarkable in its AI-agent integration, which consists of an untouched open-source framework plus a swapped configuration file, even though the surrounding persistence and evasion tradecraft is already mature — of a pattern CSA expects to recur with increasing polish: attackers embedding general-purpose AI agents inside malware so that a single natural-language instruction, rather than a purpose-built exploit chain, drives post-compromise behavior. Security teams should begin treating any AI agent framework deployed inside their own environment — whether for legitimate automation or, as here, hijacked for malicious use — as a governed asset class with its own identity, credential scope, and audit trail, consistent with the least-privilege principles CSA's AI Controls Matrix applies to non-human identities [7]. Because Carbonato's malicious behavior lives in a configuration file rather than compiled code, defenders should not expect conventional signature-based detection to reliably catch variants of this technique; monitoring should

instead focus on behavioral indicators such as unexpected outbound Telegram API traffic from server workloads, anomalous privileged-container creation, and unusual patterns of credential enumeration. Finally, organizations that self-host or expose AI development infrastructure of any kind should treat network-exposure review as a recurring, not one-time, control, given that Carbonato's own initial foothold sat unaddressed on the public internet for roughly two years before independent researchers discovered it.

CSA Resource Alignment

Carbonato sits at the intersection of threads CSA's AI Safety Initiative has already been tracking, and this document draws directly on that prior work rather than restating it. The closest analog is CSA's research note on NadMesh: A Botnet Built to Hunt Exposed AI Infrastructure for Cloud Keys, which documented a separate Go-based botnet chaining more than twenty remote-code-execution vectors — including the same unauthenticated Docker container API abuse, at roughly 30 percent of observed attempts — to harvest cloud and AI provider credentials from exposed infrastructure [6]. The recommendations in that note, particularly around default-deny network posture and rapid credential rotation after exposure, apply directly to Carbonato and reinforce that exposed Docker APIs have become a recurring, high-value entry point across multiple independent botnet operations rather than an isolated incident.

Because Carbonato's payload is a weaponized, instructable AI agent rather than a fixed binary, CSA's broader body of agentic AI sandboxing guidance is directly relevant, both as a mirror of the attacker's own tooling choices and as a defensive reference point: a recurring theme in CSA's agentic AI analysis is that ambient authority granted to agent runtimes — not model reasoning failures — tends to be the exploitable weakness, which is consistent with the mechanism Carbonato exploits on the victim side, where a privileged container hands the agent unrestricted host access. Organizations deploying legitimate AI agent frameworks internally should apply least-privilege isolation and identity-gateway controls to prevent their own agents from becoming a Carbonato-style liability if compromised through an unrelated vector.

CSA's HalluSquatting: AI Hallucinations Weaponized for Botnet Delivery documented the broader emergence of "agentic botnets," in which compromised hosts are enrolled as instructable AI-driven nodes rather than static implants, and Carbonato is a concrete, independently sourced example of that same operating model realized through a different initial access technique [5]. Read together, these two notes and CSA's AI Controls Matrix (AICM) v1.1 — whose Identity and Access Management, Threat and

Vulnerability Management, and Application and Infrastructure Security domains cover both the network-exposure and agent-privilege failures at play here — provide the control baseline organizations should use to assess exposure to this and structurally similar campaigns.

References

[1] The Hacker News. "Carbonato Botnet Compromises Docker Hosts to Deploy Telegram-Controlled Hermes AI Agent." The Hacker News, September 2026.

[2] BleepingComputer. "New Carbonato malware uses AI agents to hijack exposed Docker hosts." BleepingComputer, September 2026.

[3] ThreatDown. "CARBONATO: a botnet built around an AI agent." ThreatDown (Malwarebytes), September 2026.

[4] SC Media. "New Carbonato botnet uses AI framework to target insecure Docker daemons." SC Media, September 2026.

[5] Cloud Security Alliance. "HalluSquatting: AI Hallucinations Weaponized for Botnet Delivery." Cloud Security Alliance, July 2026.

[6] Cloud Security Alliance. "NadMesh: A Botnet Built to Hunt Exposed AI Infrastructure for Cloud Keys." Cloud Security Alliance, July 2026.

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

