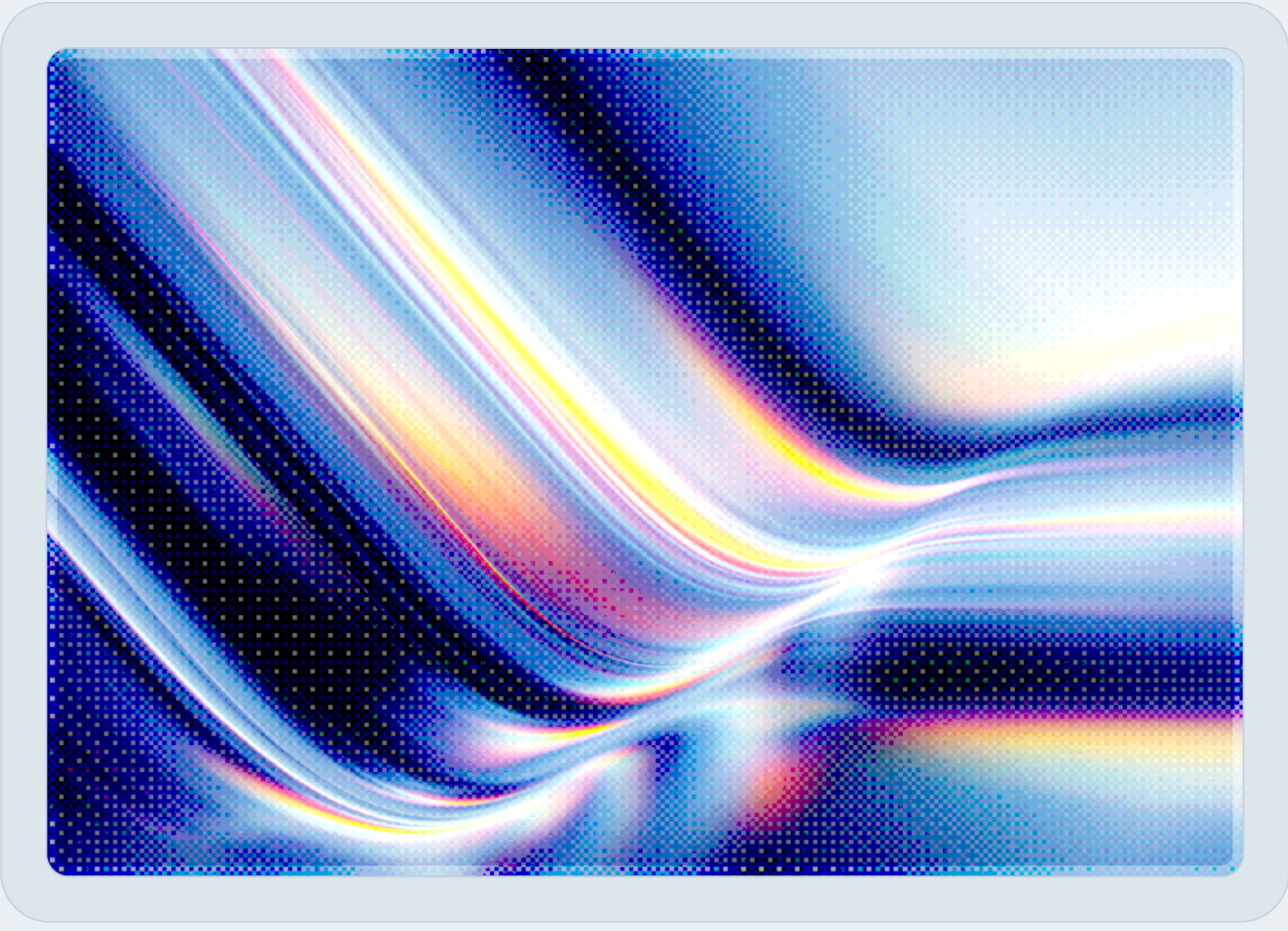
CSAI Foundation | Cloud Security Alliance

Living Off the Model API: Stolen Claude Keys as Agent Implants

Security Implications and Guidance for Coding Agents Commanded Through Their Own Platform

2026-11-13

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

On November 12, 2026, Darktrace researchers published a lab demonstration showing that an attacker holding a developer's Claude Console credentials could turn an installed Claude Code agent into a remotely tasked implant [1]. Claude memory stores served as the command-and-control (C2) channel and the Claude Files API carried the exfiltrated data, so the attacker needed no infrastructure of their own after the first stage [1][2]. Every job, result, and upload went to `api.anthropic.com`, which is the endpoint the agent already uses for inference and which organizations that use the agent must already allow through egress controls.

This is not a platform vulnerability. Darktrace states that the technique does not rely on a flaw in the Claude platform, has not been observed by Darktrace in the wild, and requires a prior compromise of the developer endpoint [1]. The risk comes from features that work as designed being used by a party who holds valid credentials. The pattern also has precedent: Microsoft documented SesameOp, a backdoor that used the OpenAI Assistants API as a relay for commands and results [6], and MITRE catalogs the general behavior as technique AML.T0096, "AI Service API" [7]. The Darktrace work extends that pattern by using the agent, rather than custom malware, to carry out the tasking.

The practical lesson is that AI platform credentials should be treated as production credentials. In the demonstration, a standard Console API key could create memory stores and upload files as well as call models, while OAuth credentials could not reach memory stores [2]. Combined with an agent already installed on the host, that broader key scope is what turns credential theft into remote tasking. Short-lived, workspace-scoped, federated credentials narrow what a stolen token can do [2][10].

Finally, destination-based controls cannot separate this traffic from legitimate use, so detection must rely on endpoint behavior and provider-side records. Defenders need to correlate agent processes, child processes, command lines, and upload volumes against a baseline [1][2], and to review the memory and file resources that exist in their AI platform workspaces.

Background

AI coding agents such as Claude Code run on developer workstations and build hosts with the user's privileges. They can read the filesystem, run shell commands, and reach internal networks. To reach the model, the agent authenticates to the vendor's API. On Linux, Claude Code stores its login credentials in `~/.claude/.credentials.json` with owner-only file permissions. On macOS it uses the system Keychain, but falls back to the same file when the Keychain is unavailable [4]. A Console sign-in can also mint a per-user API key in a dedicated workspace [10]. Malware running as the developer can typically read the credentials file wherever it is used, much as infostealers already harvest browser cookies and cloud CLI tokens.

Theft of these credentials is already documented. In February 2026, Check Point Research disclosed Claude Code flaws that allowed remote code execution and API token exfiltration through malicious project files. One issue, tracked as CVE-2026-21852, let a repository redirect API traffic to an attacker-controlled endpoint before the trust prompt appeared [8]. In May 2026, Straiker described an ACRStealer campaign that impersonated Claude Code download pages and targeted credentials stored by AI coding assistants [9]. CSA's own coverage of IDE plugin campaigns in 2026 documented the same pattern of harvesting developer AI keys [13]. Much earlier reporting on key theft focused on what the stolen keys cost the victim in inference spend. The Darktrace demonstration asks a different question: what an attacker can control with a stolen key.

The answer depends on the platform features the credential unlocks. Anthropic documents memory stores as workspace-scoped collections of text documents that persist across sessions, are readable and writable through the REST API, and are designed to give Claude Managed Agents durable memory [3]. The Files API lets clients upload and retrieve files on Anthropic's infrastructure for later use [11]. Both are legitimate developer features. Together with a locally installed agent, they give an attacker a message queue, a file drop, and an executor, all reachable through one trusted hostname.

Security Analysis

The Darktrace demonstration is worth examining in detail because each stage relies on ordinary, documented behavior rather than an exploit. The sections below trace the attack chain as Darktrace reported it, explain why the use of an installed agent as the executor changes the defensive picture, and place the technique alongside related work. They then set out the preconditions that limit the technique and the forensic evidence the platform leaves behind.

The Demonstrated Attack Chain

Darktrace Signal Labs ran the scenario against a simulated Active Directory environment, using fictitious names and credentials [1]. A small payload, fetched from a staging server and run with `dash`, located the Claude Code installation on a Linux host and collected the user's Claude API key and Console OAuth credentials. It sent them to the staging server, which replied with identifiers for a memory store the attacker had created in a workspace the developer could access [1][12]. The store held three memories that acted as a heartbeat file, an input queue for jobs, and an output file for results. Although memory stores are designed for Claude Managed Agents, Darktrace notes that they can be added to a workspace and edited through the REST API regardless of whether Managed Agents are in use, which is how the hook and the agent reached them [1]. The payload then used `systemd-run` to create a transient service that repeatedly launched Claude Code in non-interactive mode, and the initial script removed itself [1].

On each launch, a hook wrote a timestamped check-in to the heartbeat memory. The agent, running Claude Sonnet 4.5, then read the next job from the input memory, carried it out with bash tool calls, and wrote the results to the output memory [1]. The operator gave instructions in plain English rather than as encoded commands. Help Net Security reports that the systemd service launched Claude Code with the `--dangerously-skip-permissions` flag, which turns off tool-use confirmation prompts, so no human approved individual actions [2]. Anthropic's documentation states that the `bypassPermissions` mode this flag enables is intended only for isolated containers and virtual machines [5], which makes its use on a developer host a departure from vendor guidance.

Over nine jobs, the agent carried out an operator-directed intrusion sequence [1]. It found SSH, SMB, and FTP services with `ip`, `ss`, `nc`, and a targeted `nmap` scan. It pulled employee details and a password from an anonymous FTP share, generated 190 candidate usernames, and sprayed the password against the domain controller over SMB until one account succeeded [1][2]. It then enumerated SMB shares, downloaded a 1 GB file, split it into three parts, and uploaded the parts through the Files API [1][2]. The operator retrieved the data from Anthropic's platform rather than from an attacker server, so the victim network never connected to infrastructure the attacker controlled after the first stage.

Why This Matters: The Executor Is Already Installed

The demonstration differs from earlier AI-API C2 cases mainly in who does the work. SesameOp used the OpenAI Assistants API only as a mailbox. Its .NET backdoor read encrypted commands from Assistant descriptions and ran them locally, and Microsoft reports that it did not use OpenAI's models at all [6]. In the Darktrace scenario, the installed agent is the backdoor, and the attacker does not need to deliver a purpose-built implant. In the demonstration, the operator still supplied fairly specific steps, such as which `nmap` command to run and which host to spray, but delivered them as plain-language jobs rather than

as implant code [1]. Coding agents can generally also write scripts and recover from errors on their own, which suggests that less skilled operators could give vaguer instructions; the demonstration did not test this. CSA's *Living Off the Agent* whitepaper describes this as a defining property of the LOTA attack class: the agent's legitimate capabilities become the attack infrastructure, and its behavior looks the same whether the instructions are legitimate or adversarial [14].

This has practical consequences for endpoint defenses. The binary that runs is a signed, approved developer tool. The process tree shows a coding agent spawning shells, which is behavior that is common for coding agents. The network destination is a sanctioned AI provider. Each layer that defenders usually rely on therefore reports normal activity. As Darktrace's Chief AI Officer, Tim Bazalgette, put it, "Approved is not the same as safe, and only behaviour tells you the difference" [2].

Comparison With Related Techniques

The demonstration is easiest to assess against the documented techniques that preceded it. The table below places it alongside SesameOp, earlier abuse of AI chatbots as C2 proxies, and the CVE-2026-21852 credential-leak flaw, comparing the channel the attacker uses, what executes on the host, and how data leaves. It shows that the AI-API C2 channel is established, while the agent-as-executor element is new. The CVE-2026-21852 row describes a credential-leak path rather than a C2 technique, so its channel column refers to where the attacker redirected traffic.

Technique	Attacker Channel	Executor on Host	Exfiltration Path	Status
SesameOp backdoor [6]	OpenAI Assistants API (Assistant description fields)	Custom .NET backdoor	Results posted back via Assistants API	Observed in the wild (Microsoft DART, 2025)
AI chatbot C2 proxy abuse [15]	Public AI assistant web or API services	Custom malware	Varies by family	Documented by CSA, March 2026

Technique	Attacker Channel	Executor on Host	Exfiltration Path	Status
Darktrace memory-store C2 [1]	Claude memory stores (heartbeat, input, output)	Legitimate Claude Code agent	Claude Files API	Lab proof of concept, not observed by Darktrace in the wild
Malicious repo redirect, CVE-2026-21852 [8]	Attacker endpoint via `ANTHROPIC_BASE_URL`	Claude Code	API key leaked to attacker	Disclosed and patched (Claude Code 2.0.65) [18]

Preconditions and Limits

The technique depends on several conditions, and each one is a point where defenders can intervene. The attacker must already be able to run code on the developer host, must obtain a Console API key with permission to create or write memory stores and upload files, and must be able to run the agent persistently with permission checks disabled [1][2]. Darktrace tested one platform and notes that it has not reproduced the technique elsewhere [2]. Help Net Security reports that Darktrace shared the findings with Anthropic as a courtesy rather than through formal disclosure, because it did not consider the behavior a vulnerability [2]. No Anthropic statement was available as of this writing. Readers should also note that the analysis rests on a single research team's lab work published one day before this note; the other scenario sources [2][12] report on that same work.

The approach is still likely to transfer to other platforms. Any AI service that pairs inference with persistent text storage or file hosting under the same credential and hostname offers a similar channel, and Darktrace suggests that these trade-offs are worth examining in any agent platform an organization deploys [2]. Based on this analysis, CSA expects key-stealing malware families to start adding agent-tasking modules, but there is no public evidence of this yet.

Available Forensic Signals

The design of the platform gives defenders some evidence to work with. Anthropic's documentation states that every change to a memory creates an immutable memory version, attributed to the writing session, which provides an audit trail. Past versions may be deleted after 30 days unless exported [3]. Workspaces can hold their own members, API keys, and resource limits, and API keys can be scoped to a single workspace [10]. In principle, an organization that reviews unexpected memory stores, version histories showing frequent small writes to a single memory, and unusual Files API upload volumes could detect activity like this in the provider-side record. That depends on administrators knowing these resources exist and checking them.

Recommendations

The recommendations below follow the preconditions identified above. Immediate actions focus on finding exposed credentials and checking for misuse, short-term mitigations reduce the value of any credential that is stolen, and strategic measures bring AI platform credentials and agent deployments under the same governance as other high-privilege infrastructure.

Immediate Actions

Organizations that run Claude Code or similar agents with Console or API credentials should first find out where those credentials are stored. That means inventorying `~/.claude/.credentials.json` files, `ANTHROPIC_API_KEY` values in environment variables and dotfiles, and keys in shell history across developer machines and CI runners [4]. Any key found on a host with signs of compromise, or one that infostealer campaigns such as ACRStealer may have reached, should be rotated [9]. Administrators should list memory stores and recent Files API uploads in every workspace and investigate stores that no sanctioned application created [3][11].

Security teams should also block unattended, unsupervised agent execution where it is not needed. Claude Code supports a managed setting, `permissions.disableBypassPermissionsMode`, that prevents users from entering `bypassPermissions` mode [5]. Drawing on the process activity described in the demonstration [12], CSA recommends that endpoint teams add detections for agent binaries launched by `systemd-run`, cron, or other persistence mechanisms outside approved automation, and for agent processes that spawn `nmap`, `nc`, `smbclient`, or similar reconnaissance tools.

Short-Term Mitigations

Over the following weeks, organizations should reduce the value of a stolen credential. Darktrace recommends moving to OAuth or federated identity where supported, and, where static keys remain, using short-lived keys scoped to a single workspace and kept out of dotfiles, environment variables, and shell history [1][2]. Claude Code's `apiKeyHelper` mechanism can fetch short-lived credentials from a vault on demand instead of keeping a long-lived key on disk [4]. Workspace scoping limits which memory stores and files a key can reach, and workspace resource limits cap the spend an attacker can generate [10]. Spend limits contain financial damage, however; they do not prevent exfiltration.

Detection engineering should then move from destination-based to behavior-based analytics. Effective detections are likely to combine signals: agent harness processes correlated with their child processes and command-line arguments, outbound-to-inbound data ratios to the AI provider that differ sharply from the user's baseline, and large uploads to file endpoints from hosts that normally only send prompts [1][2]. Where possible, AI API traffic should pass through an inspecting egress proxy or gateway that can tell inference calls from storage and file calls. Platform-level usage data should be collected so endpoint and provider records can be compared [10].

Strategic Considerations

Over the longer term, this demonstration means AI platform credentials need to be treated as non-human identities with broad reach. A credential that can store data and host files, and that can direct an installed agent through those features, belongs in the same governance tier as cloud administrator tokens, with ownership, rotation, scoping, and monitoring to match. Agent deployments should also move toward the isolation patterns CSA has described for coding agents: ephemeral sandboxes, per-workload identity, scoped capabilities, and audit trails that the agent cannot modify [16]. An agent running in a disposable VM with its own scoped credential limits what a stolen key can reach.

Organizations should also ask their AI platform providers to make these resources more visible. Darktrace's Bazalgette described the credential scope and visibility issues as "trade-offs between capability and exposure" [2], and the provider-side features that would shift that balance are straightforward to name. Useful features would include alerts when a new memory store or large file appears in a workspace, the ability to turn off storage and file features for keys that only need inference, and exportable logs that security operations teams can ingest. Procurement and vendor assessments for AI services can include these requirements, and red teams should add AI-API C2 scenarios to their adversary emulation plans.

CSA Resource Alignment

The most directly relevant CSA artifact is the whitepaper *Living Off the Agent: Agentic AI Systems as Enterprise Attack Infrastructure — The LOTA Attack Class* [14]. It defines the class of attacks in which adversaries take over an agent's trusted access and capabilities rather than deploying malware. Its sections on credential harvest, orchestrator and memory manipulation, and detection failures in traditional controls anticipate several elements of the Darktrace scenario, notably the harvesting of the agent's credentials, the use of shared memory as a control channel, and the failure of signature- and destination-based controls. Its defense principles on least-privilege access and identity hygiene for non-human identities match the recommendations above.

CSA's research note *AI Chatbots as Covert Command-and-Control Infrastructure* [15] documented the earlier stage of this trend: malware using trusted AI services as C2 relays, including SesameOp. The present note follows that line of work from AI services used as mailboxes to AI agents used as executors, and its egress and behavioral-detection guidance still applies. *AI Toolchain Hijacked: IDE Plugin API Key Theft* [13] covers the credential-theft campaigns that supply the precondition for this attack, and its advice to rotate AI keys and move them into vault-backed storage is the first line of defense here.

For architecture, *Sandboxing Agentic AI: Least-Privilege Patterns for MCP and Coding Agents* [16] provides reference patterns for isolation, workload identity, capability scoping, and tamper-resistant audit that limit what a hijacked coding agent can do. At the control level, the CSA AI Controls Matrix (AICM) v1.1 [17] provides auditable requirements in its Identity and Access Management, Threat and Vulnerability Management, and Application and Interface Security domains. Organizations can use these to formalize the credential scoping, agent access restriction, and sandboxing measures in this note.

References

[1] Darktrace (M. Tajtelbaum and S. Lister). "Living off the AI: Shared Memories as a Covert C2 Channel." Darktrace Blog, November 12, 2026.

[2] Help Net Security (M. Zorz). "Stolen Claude API keys can turn Claude Code into a remote-controlled implant." Help Net Security, November 12, 2026.

[3] Anthropic. "Using agent memory." Claude Platform Documentation, accessed November 13, 2026.

[4] Anthropic. "Authentication." Claude Code Documentation, accessed November 13, 2026.

[5] Anthropic. "Permission modes." Claude Code Documentation, accessed November 13, 2026.

[6] Microsoft Incident Response. "SesameOp: Novel backdoor uses OpenAI Assistants API for command and control." Microsoft Security Blog, November 3, 2025.

[7] MITRE. "AML.T0096: AI Service API." MITRE D3FEND (mirror of the MITRE ATLAS technique), accessed November 13, 2026.

[8] Check Point Research. "Caught in the Hook: RCE and API Token Exfiltration Through Claude Code Project Files." Check Point Research, February 25, 2026. (Covers both CVE-2025-59536 and CVE-2026-21852.)

[9] Straiker (C. Vincent and A. Rousseau). "Is Your Claude Code Real? How to Spot the ACRStealer Malware." Straiker Blog, May 27, 2026.

[10] Anthropic. "Workspaces." Claude Platform Documentation, accessed November 13, 2026.

[11] Anthropic. "Files API." Claude Platform Documentation, accessed November 13, 2026.

[12] Bromure (R. Deraison). "The orders came through the API." Bromure Blog, November 13, 2026.

[13] Cloud Security Alliance. "AI Toolchain Hijacked: IDE Plugin API Key Theft." CSA AI Safety Initiative, June 21, 2026.

[14] Cloud Security Alliance. "Living Off the Agent: Agentic AI Systems as Enterprise Attack Infrastructure – The LOTA Attack Class." CSA AI Safety Initiative, May 18, 2026.

[15] Cloud Security Alliance. "AI Chatbots as Covert Command-and-Control Infrastructure: Emerging Threat Patterns and Enterprise Defenses." CSA AI Safety Initiative, March 8, 2026.

[16] Cloud Security Alliance. "Sandboxing Agentic AI: Least-Privilege Patterns for MCP and Coding Agents." CSA AI Safety Initiative, May 30, 2026.

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

[18] National Institute of Standards and Technology. "CVE-2026-21852 Detail." National Vulnerability Database, accessed November 13, 2026.