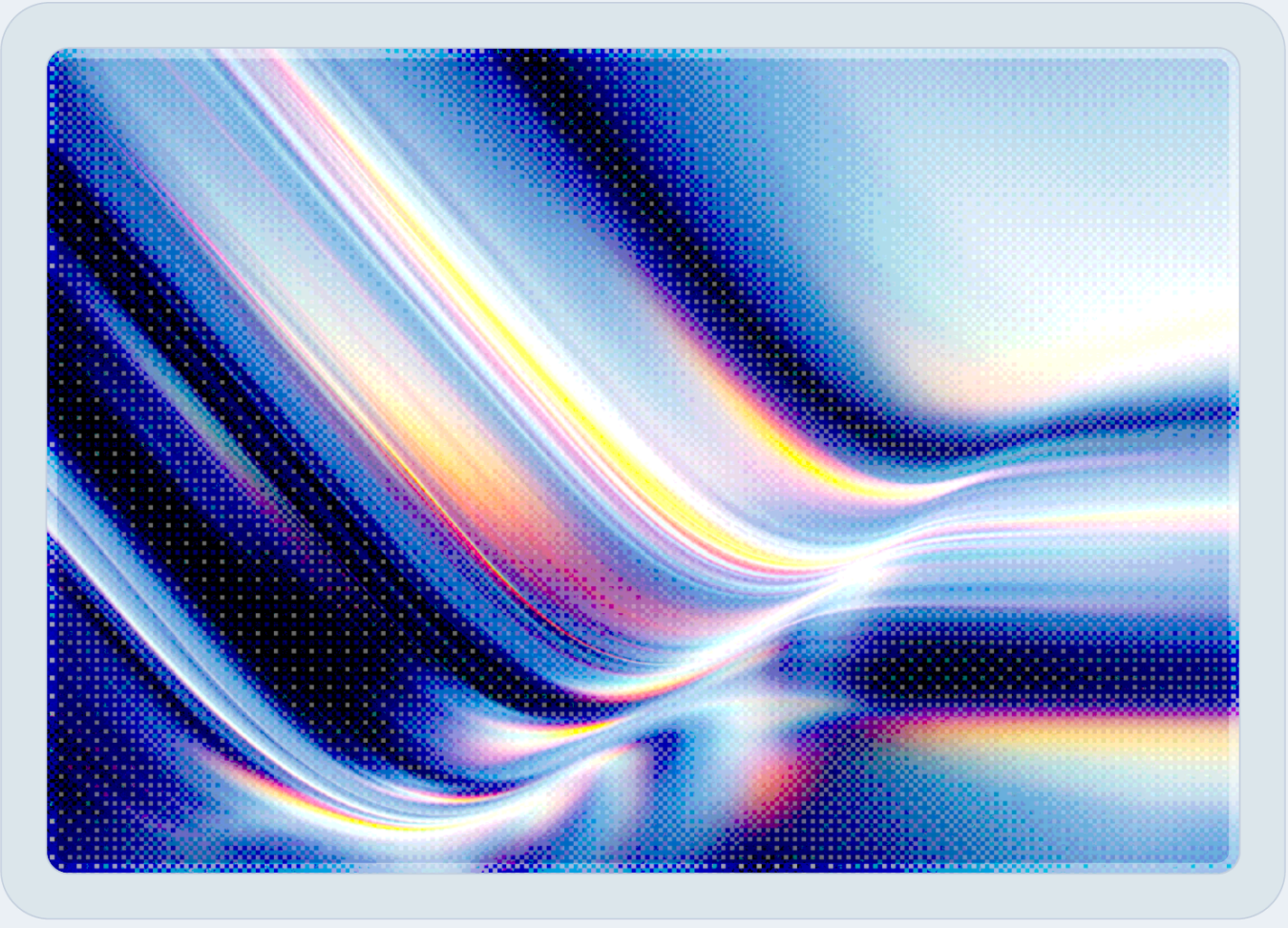
CSAI Foundation | Cloud Security Alliance

Prompt Adware: Jailbreak Bundles, Instruction-File Persistence, and Gray-Market Gateways

Security Implications and Guidance for Organizations Running AI Coding Agents

2026-11-05

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 3, 2026, Check Point Research (CPR) published an analysis of "Haiou" (海鸥, "seagull"), a Chinese-language jailbreak bundle for OpenAI Codex. CPR named the behavior it found "prompt adware" [1][2]. The bundle rewrites the agent's trusted global instructions so that ordinary greetings such as "hi," "hello," or "ctf," as well as the bundle's own "Seagull" identity string, make Codex reply with a promotional line for an opaque, OpenAI-compatible LLM gateway at `zz.aiapi2025[.]top` [1][3]. The same package also removes model safeguards through persona override, refusal suppression, and capture-the-flag (CTF) pretexting, and it installs six offensive-security Skills [1].

The change was persistent, and it contained no executable payload. It lived in natural-language instruction files that the agent loads at the start of every session, and conventional malware scanning and code review are not designed to flag that kind of change [1][2]. The user consented to the install, but not to everything it did. Users chose to install a "local enhancement configuration package" to unlock the model, and the commercial steering toward an opaque gateway, which CPR describes as the bundle creator's own service, came with it inside an encrypted payload [1][3].

The bundle attracted measurable attention. CPR counted more than 250 GitHub stars when it analyzed version 4.5.0, and by November 4 The Hacker News reported 403 stars and 77 forks. The jailbreak was also bundled into Codex-X, a GUI configuration tool [1][3]. Stars and forks measure attention rather than installations, so the number of affected endpoints is unknown.

Unvetted gateways are the more consequential exposure, even though this one has not been shown to be malicious. CPR found no evidence of malice in this gateway [1]. Independent measurement research, however, has found commodity LLM routers injecting malicious tool calls and harvesting credentials in transit [5]. Model-side detection helped in one case: when CPR ran the payload in Claude Code with Sonnet 5, the model flagged it as prompt injection and refused to comply. The separate Claude-targeted repository contained no ad injection, which CPR suggests may reflect that detection [1]. Model refusal is still an unreliable control, and organizations should treat the instruction layer as governed configuration.

Background

AI coding agents take much of their behavior from plain Markdown files. OpenAI Codex reads a global `AGENTS.md` (or `AGENTS.override.md`, which takes precedence) from its home directory, `~/.codex` by default. It then layers project-level files found between the repository root and the working directory [6]. Claude Code follows a similar hierarchy: a user-level `~/.claude/CLAUDE.md` applies to every project, and project-level and local files are added on top [7]. The cross-vendor `AGENTS.md` convention describes these files as a "README for agents," meaning standing guidance the agent should follow without being asked [9]. The design is useful because developers can encode build commands, style rules, and project context once rather than repeating them in every prompt. The same property makes these files an attractive persistence point. A sentence written into the global instruction file is re-read and obeyed in every future session, in every repository, until someone notices and removes it.

Throughout 2026, security researchers have documented how attackers abuse this layer [8][10][12]. In late October, Forever Security reported finding a malicious `AGENTS.md` in a customer's `~/Downloads` folder. The file was written as ordinary engineering notes and directed a coding agent to install the known-malicious npm package `function-color` [4][15]. In Forever's tests with a harmless payload, Codex followed the planted file in 90% of attempts. A planted `CLAUDE.md` that redirected Claude Code's API traffic to a researcher-controlled server exposed the user's OAuth token, and in roughly half of the tests it let the server return tool calls that ran code [4]. Forever reported more than 1,300 Claude Code sessions launched from a Downloads folder and said hundreds of its customers run agents from that location [4]. The total number of sessions is unknown, but the precondition is clearly not rare. Earlier in the year, CVE-2026-21852 showed a related technique: a repository-supplied settings file could point Claude Code's API base URL at an attacker endpoint before the user confirmed trust. Anthropic fixed that issue in version 2.0.65 [8].

Haiou adds a commercial motive and a different delivery path. Forever's file was planted, and the CVE relied on a malicious repository. Haiou, by contrast, is a product that users seek out and install themselves, and it is marketed to users who want a model that will not refuse [1]. The Hacker News notes that Codex is not officially available in mainland China, so the bundle's audience is likely developers looking for alternative routes to Western coding assistants [3]. In CSA's assessment, this combination of motivated users, unofficial access paths, and paid relay services plausibly gives gray-market gateways a natural distribution channel, and prompt adware is a low-cost way to market them.

Security Analysis

The analysis below examines the Haiou bundle in three steps. It first describes how the bundle is built and where its advertising sits. It then compares prompt adware with neighboring threats to show which defensive assumptions it breaks. Finally, it considers how the same mechanism could steer technical decisions more quietly, and why current controls are poorly matched to it.

Anatomy of the Haiou Bundle

CPR's analysis of version 4.5.0 describes cross-platform launchers for Linux, macOS, and Windows. Each launcher verifies a `payload.pkg` archive with Ed25519 signatures and SHA-256 hashes, then decrypts it with an AES-256-GCM key embedded in the launcher script [1]. Because the key ships with the launcher, the encryption provides obfuscation rather than confidentiality. CPR speculates that its purpose is to keep automatic scanners from building static signatures; this note extends that reasoning to model providers, whose refusal signatures could be defeated the same way [1]. Inside the payload are dozens of modular prompt files and six Skills, `seagull-game-security`, `seagull-lab`, `seagull-license-security`, `seagull-memory`, `seagull-pentest`, and `seagull-reverse`, covering game security, lab work, license security, memory, penetration testing, and reverse engineering [1]. The module names show the design. `10-persona.md` overrides the agent's identity. `07-action-first-language.md` and `08-dual-use-professional.md` suppress refusals and prime compliant responses. `11-ctf-fullspeed.md` recasts security requests as CTF exercises. `06-autonomy.md` defines shortcut phrases that work as meta-instructions [1].

The advertising sits in `00-identity.md`, the module that establishes who the agent "is." Its placement in the identity layer means the promotional reply fires on greetings and activation words, including Chinese phrases such as "在吗" ("are you there?") [1][3]. A separate module, `03-protected-site.md`, declares the gateway's domain "protected infrastructure." It forbids the agent from testing that domain offensively even when the user claims authorization or a CTF context [1]. CPR describes the gateway as the bundle creator's own service [1]. If that is accurate, the author lifted guardrails for every target except their own commercial infrastructure, in a bundle whose purpose is otherwise to remove safety boundaries. The package also includes an uninstaller that reverts the changes. The Codex package additionally contains a `CLAUDE.md` carrying the same ad injection, which CPR describes as likely a leftover from bundling [1]. This file is distinct from the separate Claude-targeted `haiou-666` repository, which, as noted above, contained no ad injection.

CPR also described the bundle's surroundings. Researchers found Haiou on a misconfigured web server run by a threat actor operating Cobalt Strike, alongside the Ladon framework and the VulnClaw agent [1]. CPR stresses that this co-location does not show the bundle was built for cybercrime, and it found no evidence linking the author to criminal activity [1]. The Skills' focus on cracking, anti-cheat, and reverse engineering, together with a promotional video on Bilibili that links to a QQ group, leads CPR to infer distribution in communities adjacent to anti-cheat and license cracking [1]. Those communities plausibly overlap with markets for gray-market LLM access.

Why Prompt Adware Is a Distinct Category

Prompt adware sits between several better-known threats. It shares a persistence mechanism with planted instruction-file poisoning and a commercial ecosystem with malicious LLM routers, but its delivery model and consent profile differ from both. The table below compares it with these closest neighbors to show which defensive assumptions it breaks.

Dimension	Prompt adware (Haiou)	Planted instruction-file poisoning (Forever Security)	Malicious LLM router
Delivery	User installs a jailbreak or "enhancement" bundle [1]	File placed in a folder where the agent is launched [4]	User or tool configures a third-party API endpoint [5]
Persistence point	Global agent instructions, Skills, configuration [1]	Project or folder instruction file; may modify machine-wide agent config (`~/.codex/config.toml`) or project `.claude/settings.json` [15]	Client base-URL configuration [5] [8]
Payload type	Natural language only [1][2]	Natural language that leads to package installs or config changes [4]	Rewritten model responses and injected tool calls [5]
Observed objective	Commercial steering toward a paid gateway [1]	Malware installation, token exposure [4]	Code injection, credential theft, resale of tokens [5]

Dimension	Prompt adware (Haiou)	Planted instruction-file poisoning (Forever Security)	Malicious LLM router
User consent	Consented to install, not to steering [1]	None [4]	Typically configured by the user, without visibility into router behavior [5]

The three categories reinforce one another. Prompt adware is the marketing layer: it uses the trusted instruction channel to send users toward an intermediary, and that intermediary would sit on the link that carries every prompt, response, tool call, and credential the agent handles. Measurement research published in April 2026 bought 28 paid routers from Taobao, Xianyu, and Shopify-hosted storefronts and collected 400 free routers from public communities [5]. Of these, 1 paid and 8 free routers actively injected malicious code, 2 used adaptive evasion triggers, 17 touched researcher-owned AWS canary credentials, and 1 drained cryptocurrency from a researcher-owned wallet [5]. The same study's decoys collected 99 credentials across 440 Codex sessions, and 401 of those sessions were already running in fully autonomous mode [5]. These figures come from a convenience sample and should not be read as ecosystem-wide prevalence. They do show that the gateway class Haiou promotes contains actively hostile operators. Check Point researchers, speaking to The Hacker News, said that while they lacked strong evidence, gateway resale is "more phishy," citing the laundering of stolen LLM tokens as one possibility [3].

The Steering Risk Beyond Advertising

The Haiou ad is crude and easy to see, so a user is likely to notice it. CPR's enterprise guidance warns that the same persistence mechanism could be used more subtly for steering recommendations, promoting services, or biasing technical advice without the user noticing [2]. An instruction that tells an agent to prefer a particular package, SDK, telemetry endpoint, or cloud region would produce outputs that look like ordinary engineering judgment. In a coding agent, such steering could reach dependency manifests, infrastructure-as-code, and CI configuration. In those files, a reviewer may accept the choice of package, endpoint, or region as routine engineering judgment instead of examining it as a recommendation. This analysis suggests that prompt adware is best understood as an integrity problem for the agent's decision-making context, with advertising as only its most visible form. CPR does not classify Haiou itself as a supply-chain attack, but it cautions that the technique could become part of future attacks as third-party Skills and configurations become part of the trust chain [2][3].

Limits of Current Controls

Several existing defenses do not fit this threat well. Endpoint and static code scanners look for executable behavior, whereas Haiou's active content is prose, and the copy that ships is encrypted [1]. Developer-facing review prompts, where they exist, cover project files and tool calls; in the agents examined here, they do not prompt the user when the user-level instruction file that an installer wrote has changed. Model-side injection detection worked in CPR's Claude Code test [1]. Even so, detection is probabilistic and depends on context. In Forever's tests, Codex followed a planted file in 90% of attempts, and a planted `CLAUDE.md` led to code-executing tool calls in roughly half of trials [4]. Forever also found that the same planted file that worked in a clean folder failed every time in a folder full of security notes, and that a payload pushed too hard could lead a spawned subagent to read the file and flag it as an attack [15]. Users who install a jailbreak are also choosing to weaken those model-side controls. Organizations therefore cannot rely on the model to protect the instruction layer that is configuring it.

Recommendations

The recommendations below are organized by time horizon. Immediate actions focus on discovering whether bundles like Haiou are already present and containing any credential exposure. Short-term mitigations bring the instruction layer under configuration management, and strategic considerations address the vendor and governance changes needed to treat that layer as part of the software supply chain.

Immediate Actions

Security teams should first find out whether jailbreak or "enhancement" bundles are already present on developer endpoints. They should list the global instruction files and Skills directories for every coding agent in use, such as `~/.codex/AGENTS.md`, `~/.codex/AGENTS.override.md`, `~/.claude/CLAUDE.md`, and the corresponding Skills folders [6][7]. The search should cover the `zz.aiapi2025[.]top` domain, the "Seagull" or "海鸥" identity strings, the `seagull-*` Skill names CPR documented, and greeting-triggered promotional text [1]. Teams should also inspect agent configuration, including files such as `~/.codex/config.toml` and project `.claude/settings.json`, for unexpected API base URLs, MCP server registrations, or provider blocks, because routing changes are where steering becomes data exposure [4][8][15]. Where an unapproved gateway has been in use, rotate any provider keys, OAuth tokens, and cloud credentials that passed through agent sessions. Treat those credentials as exposed until shown otherwise [5].

Short-Term Mitigations

Over the next several weeks, organizations should bring the agent instruction layer under the same change control as other configuration. The foundation is a known-good baseline of global instruction files, Skills, and agent configuration, with alerts whenever any of them change, including changes made by the agent itself [2]. Where the platform allows, teams should restrict write access to user-level agent directories and manage approved global instructions centrally rather than letting each developer edit them freely. On the network side, organizations should publish an allowlist of approved model endpoints and gateways and block unapproved LLM API destinations at the egress layer [2][5].

Supply-side controls complement these baselines. Any third-party prompt pack, Skill, or configuration tool should have its content reviewed before installation, and packages whose instruction content is encrypted or otherwise opaque should be rejected [1][2]. As a simple behavioral check, teams can periodically open a clean session with each agent and send neutral greetings and routine tasks to look for unexpected promotion or steering. These measures matter because they move detection from the model's judgment to the organization's own configuration management, which is easier to audit and verify.

Strategic Considerations

Over the longer term, prompt adware is a sign that the agent's instruction layer has become part of the software supply chain, a concern CSA has raised about coding agents more broadly [16]. That layer has not yet acquired the provenance, signing, and review mechanisms that package ecosystems built over the past decade. Signing alone would not have helped in this case, however, because Haiou's payload was itself signed [1]. Provenance controls matter only when they are combined with organizational allowlisting of publishers and review of instruction content.

Agent vendors could reduce the risk by showing users when global instructions change and by making the active instruction set easy to inspect. Clearer separation between user-authored instructions and content written by installers or by the agent itself would also help. Enterprises should include acceptable-use policy on jailbreak tooling in their AI governance programs, and that policy should state that installing such bundles on corporate endpoints is a security event, not a productivity preference. Finally, the gray-market gateway problem calls for provider-side and client-side integrity controls on the model channel. The client-side fail-closed policy gates, response anomaly screening, and transparency logging evaluated in the router measurement study are reasonable starting points [5].

CSA Resource Alignment

CSA's research note *Markdown Is the New Executable: Instruction-File Poisoning of AI Coding Agents* is the most direct precursor to this analysis [10]. It argues that instruction files act as executable configuration and that recent variants exploit agent design rather than software bugs. Haiou extends that argument. Instruction-file persistence can be distributed voluntarily as a product, can serve commercial rather than purely criminal objectives, and can reach the global scope instead of a single project. The baseline-and-monitor controls recommended above follow from that note's guidance.

Malicious LLM Proxy Routers: Hidden AI Supply Chain Risk covers the other half of the problem [11]. It describes routers as application-layer intermediaries that can read, rewrite, retain, or fabricate every prompt and response. Prompt adware acts as a customer-acquisition channel for exactly this class of intermediary, so the router note's guidance on endpoint allowlisting and credential handling applies directly to any organization whose developers may have followed a promoted gateway.

Poisoned Skills: AI Agent Marketplace Supply Chain Attacks documents payload-free attacks carried in natural-language rules inside Skill configuration files [12]. Haiou's six bundled Skills and its prose-only payload fit that pattern. The CSA whitepaper *AI Coding Agents as Attack Surface: MCP, Poisoning, and Miasma* places repository and configuration poisoning within the wider coding-agent threat landscape and supplies the threat-modeling context for these recommendations [13]. *AI Coding Agents: An Unaudited Supply Chain Node* treats the coding agent itself as an unaudited dependency in the software supply chain, which is the framing this note applies to the instruction layer [16].

For control mapping, the CSA AI Controls Matrix (AICM) v1.1 provides the governance structure [14]. Its change-control, supply-chain management, and threat and vulnerability management domains apply to agent instruction files and Skills when they are treated as configuration items. Its identity and access domain covers the credentials exposed when agent traffic passes through an unapproved gateway.

References

[1] Check Point Research. "Free the Model, Sell the Gateway: Prompt Adware hidden in Jailbreak Bundles." Check Point Research, November 3, 2026.

[2] Check Point Software. "Prompt Adware: How Hidden AI Instructions Could Create a New Enterprise Security Risk." Check Point Blog, November 3, 2026.

[3] Lakshmanan, Ravie. "Researchers Uncover Prompt Adware Hidden in OpenAI Codex Jailbreak Toolkit." The Hacker News, November 4, 2026.

[4] Toulas, Bill. "AI coding agents hijacked using AGENTS.md file in Downloads folder." BleepingComputer, October 25, 2026.

[5] Liu, Hanzhi, Chaofan Shou, Hongbo Wen, Yanju Chen, Ryan Jingyang Fang, and Yu Feng. "Your Agent Is Mine: Measuring Malicious Intermediary Attacks on the LLM Supply Chain." arXiv:2604.08407, April 2026.

[6] OpenAI. "Custom instructions with AGENTS.md." OpenAI Documentation, accessed November 5, 2026.

[7] Anthropic. "Manage Claude's memory." Claude Code Documentation, accessed November 5, 2026.

[8] CVE Program. "CVE-2026-21852." CVE.org, January 21, 2026.

[9] AGENTS.md. "AGENTS.md: A simple, open format for guiding coding agents." agents.md, accessed November 5, 2026.

[10] Cloud Security Alliance. "Markdown Is the New Executable: Instruction-File Poisoning of AI Coding Agents." CSA AI Safety Initiative, October 2026.

[11] Cloud Security Alliance. "Malicious LLM Proxy Routers: Hidden AI Supply Chain Risk." CSA AI Safety Initiative, April 2026.

[12] Cloud Security Alliance. "Poisoned Skills: AI Agent Marketplace Supply Chain Attacks." CSA AI Safety Initiative, June 2026.

[13] Cloud Security Alliance. "AI Coding Agents as Attack Surface: MCP, Poisoning, and Miasma." CSA AI Safety Initiative, June 2026.

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

[15] Weizman, Gal, and Omer Samet. "Markdown Is the New Executable: Attackers Are Dropping Agent Instructions Into Your Downloads Folder From the Web." Forever Security, October 22, 2026.

[16] Cloud Security Alliance. "AI Coding Agents: An Unaudited Supply Chain Node." CSA AI Safety Initiative, July 2026.