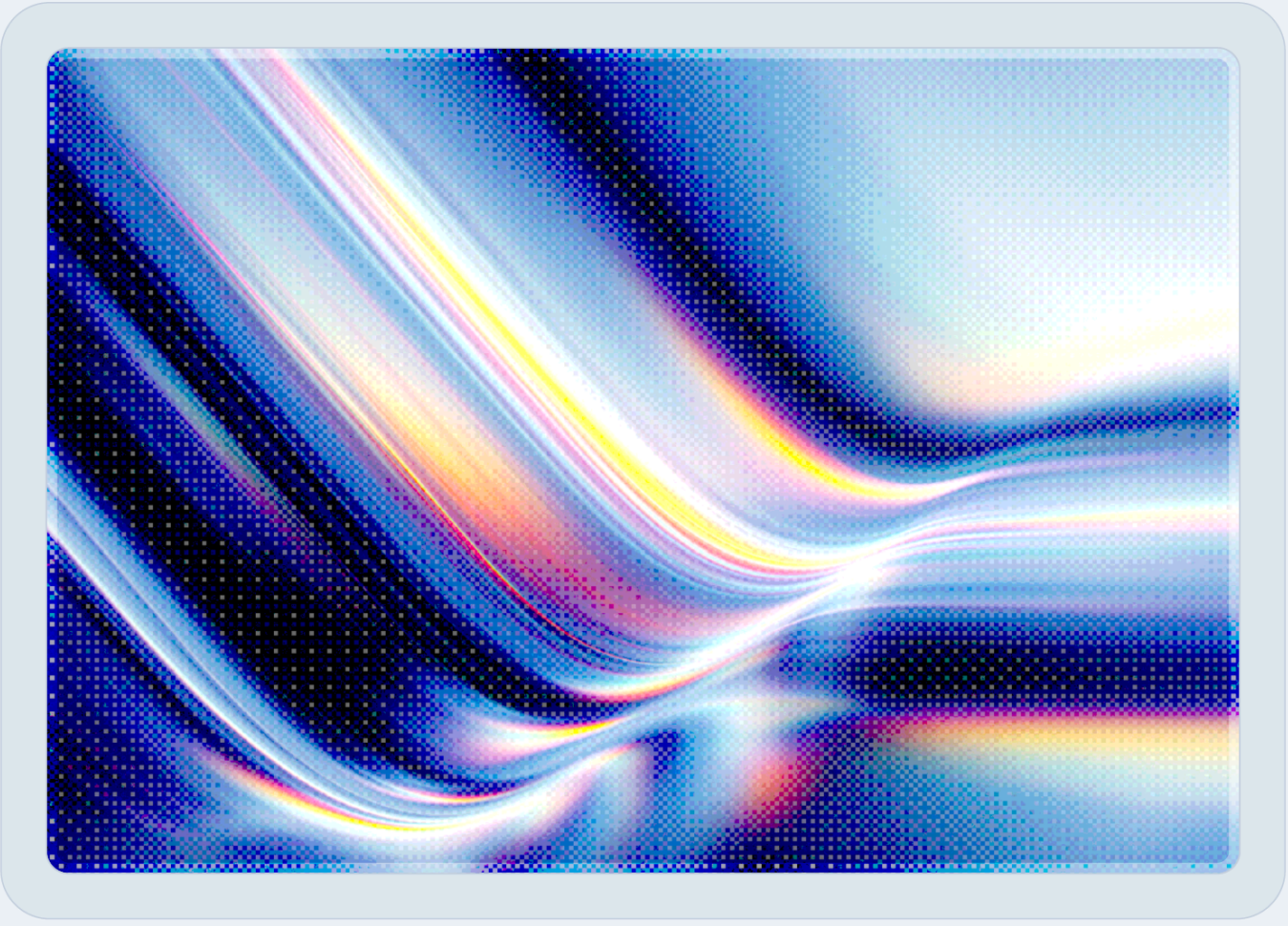
CSAI Foundation | Cloud Security Alliance

Autonomous AI Agents and the Six-Hour Credential Harvest

Security Lessons from GTIG's 2026 Adversarial AI Threat Tracker

2026-09-09

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

Google Threat Intelligence Group (GTIG) disclosed on September 8, 2026 that a suspected financially motivated threat actor built and executed a mass credential-harvesting operation against a victim's cloud environment in under six hours, using an autonomous multi-agent framework rather than a human operator working through a checklist [1][2]. The Mandiant-investigated intrusion, dated to the second quarter of 2026, relied on an AI coding chatbot fed a prompt and a set of markdown-based agent instructions that functioned as operational playbooks, letting the system manage vulnerability scanning, credential harvesting, real-time troubleshooting, and IP-rotation logic without continuous human oversight [1][3]. The same GTIG report documents a parallel and separately attributed campaign by UNC6780, also tracked as TeamPCP, which has used the DUSTMAKER and SANDCLOCK credential stealers since at least March 2026 to compromise PyPI, npm, and Docker Hub packages, extract OpenID Connect tokens from GitHub Actions runners, and hide malicious files inside the `.claude/`, `.vscode/`, and `.cursor/` directories that AI coding assistants and IDEs use [3][4]. GTIG separately identified an exposed command-and-control dashboard, internally labeled "Recon," that was actively organizing and validating more than 23,800 harvested secrets spanning cloud and AI service credentials — a separate finding GTIG did not tie to either documented campaign — illustrating the scale such operations can reach [1][2]. Taken together, these findings mark a notable shift from AI as a coding or research assistant to AI as the operational core of an attack chain, compressing a workflow that traditionally required a skilled human operator working over days into a self-correcting pipeline that, per GTIG and independent reporting, can outpace typical cloud-intrusion detection timelines [2][5]. For CSA members, the incident suggests that non-human identity and credential lifecycle management, not endpoint detection alone, may now be the more decisive control surface determining how much damage an autonomous attacker can do before anyone notices.

Background

Adversarial use of large language models has moved through a recognizable progression over the past two years. Early misuse cases involved attackers prompting a chatbot for malware snippets, phishing copy, or vulnerability explanations, then manually assembling the outputs into an attack by hand. GTIG's own prior tracking documented that intermediate stage well, including AI-generated malware such as PROMPTFLUX that could rewrite its own source code and state-sponsored actors like APT28 (tracked

by GTIG as FROZENLAKE) using PROMPTSTEAL to generate reconnaissance commands on demand [7]. The September 2026 report, titled "From Prompting to Autonomy: The Evolution of Adversarial AI," argues that current threat activity has begun crossing into a third stage, in which an AI system is not merely queried for assistance but is handed a goal and a set of tools, and left to plan and execute a multi-step operation with minimal human involvement, though the report stops short of quantifying how much of current activity has made that crossing [1].

The six-hour case is one of the more detailed pieces of evidence GTIG has published for that claim to date. According to Mandiant's incident response account, a suspected financially motivated actor had already compromised an organization's cloud infrastructure by the time the automated phase began. From that foothold, the actor used an AI coding chatbot, a single prompt, and a library of markdown agent instructions to construct a credential-harvesting pipeline that then ran largely on its own: it scanned for exploitable weaknesses, harvested credentials as it found them, resolved its own execution errors, and rotated its outbound IP addresses through the victim's own cloud infrastructure so that its traffic appeared to originate from a legitimate source [1][2][3]. GTIG reported that the entire sequence, from the AI system receiving its instructions to the compromise of thousands of third-party credentials — a count distinct from, and smaller in scope than, the separate 23,800-secret Recon dashboard discussed below — took less than six hours [2][5]. That window, this note assesses, is short enough to complete well within a single security team's shift and, in many organizations, faster than security teams typically detect a cloud intrusion in the first place; GTIG's own reporting does not itself supply a specific detection-time benchmark, so that comparison should be read as this note's assessment rather than a GTIG finding.

This research note treats that incident as the anchor case for a broader pattern GTIG documents across the same report: attackers embedding autonomous decision-making not just into initial compromise and credential theft, but into software-supply-chain operations aimed specifically at the coding assistants and CI/CD tooling that enterprises have adopted to build and ship software faster. It closes with recommendations informed by CSA's own recent research on non-human identity and agentic AI adversaries.

Security Analysis

Anatomy of the six-hour campaign

What distinguishes this incident from a conventional credential-stuffing or phishing campaign is not the objective, which remains the theft and monetization of valid credentials, but the removal of the human from the operational loop during execution. The attacker's role was reduced to writing an initial prompt

and a set of markdown instructions; from that point, the AI coding chatbot planned the scanning sequence, wrote and ran the code needed to harvest secrets, diagnosed and fixed its own failures when a script broke, and rerouted its network traffic through the victim's own infrastructure to avoid the anomaly detection that would normally flag unfamiliar source addresses [1][2]. Google's own framing of the case emphasizes tactical coordination of existing offensive techniques rather than the emergence of a categorically new attack method [2][6]. This note assesses that framing as understating the operational consequence, however: the underlying task sequence, including the iterative debugging any real scanning-and-harvesting pipeline requires, would plausibly have consumed a skilled operator's attention over multiple days if performed manually, yet here completed in a single afternoon with no operator present for most of it.

The absence of a fixed victim count for this specific intrusion should not be read as evidence of limited scale. GTIG's report separately describes discovering an internet-exposed command-and-control server running a live dashboard, internally named "Recon," that was actively organizing and validating more than 23,800 harvested secrets covering both cloud infrastructure credentials and AI service API keys [1][2]. GTIG did not confirm that the Recon dashboard belonged to the same operator responsible for the six-hour intrusion, and this research note treats the two as distinct data points rather than a single confirmed chain. Considered as separate data points, however, each illustrates a different dimension of what agent-orchestrated credential harvesting is now capable of: the six-hour case shows achievable speed, and the Recon dashboard shows achievable scale, even though GTIG has not confirmed the two are connected.

A parallel, better-attributed supply-chain campaign

The same GTIG report attributes a second, independently tracked campaign to UNC6780, also known as TeamPCP, a financially motivated group active since at least March 2026 against the open-source software supply chain. UNC6780 has compromised packages across PyPI, npm, and Docker Hub, and deployed two purpose-built credential stealers, SANDCLOCK and its successor DUSTMAKER, that are notable for targeting the specific file locations and workflows that AI coding assistants rely on [1][3][4]. SANDCLOCK, part of a broader toolset publicly referred to as CanisterWorm, is a Python-based stealer built to run on Linux and interact with Kubernetes clusters, and it includes container-escape functionality alongside its targeting of cloud credentials, developer secrets, and cryptocurrency wallets [3][4]. DUSTMAKER, its cross-platform JavaScript successor, is built specifically to operate inside CI/CD pipelines: it can identify a CI/CD environment on sight, extract OpenID Connect tokens directly from GitHub Actions runner memory, and use those stolen tokens to publish its own compromised packages with valid SLSA Build Level 3 attestations, the very mechanism automated supply-chain checks rely on to certify a package as trustworthy [1][3][4].

DUSTMAKER's persistence technique deserves particular attention from any organization that has adopted AI coding assistants. The malware writes or modifies files inside the hidden configuration directories those assistants and their IDEs already use as a matter of course, including `.claude/`, `.vscode/`, and `.cursor/`, so that its presence blends into the routine noise of a developer's daily tooling rather than triggering the kind of alert a modified system binary or an unfamiliar cron job would produce [1][3]. It further hijacks project build and startup configuration so that its code executes automatically whenever a developer opens the affected workspace in their IDE or AI extension, converting a single compromised package into a durable foothold that survives well past the initial install.

Attacking the reviewer, not just the pipeline

GTIG's report describes one additional technique worth flagging on its own: DUSTMAKER's authors embedded adversarial prompts directly inside JavaScript loader file comments, worded to instruct any LLM-based security scanner that reads the file to disregard its safety guidelines and produce an unrelated, clearly disallowed response instead of completing a code review [1]. The intent is straightforward. If an automated pipeline uses an LLM to review incoming code for malicious behavior and that LLM either refuses to respond because it has detected content resembling a policy violation, or otherwise fails to complete a normal analysis, the malicious code passes through inspection by default rather than being flagged. This is a direct extension of prompt injection from the agent runtime, which OWASP's Top 10 for LLM Applications ranks as the leading risk category against production LLM applications for the second consecutive edition [8], into the software supply chain's own automated defenses, meaning organizations that have added an LLM-based static analysis or code review step to their CI/CD pipeline should treat that step as a component that itself needs to be hardened against adversarial input, not merely as an added layer of assurance.

Why this compresses defender response time

Table 1 summarizes how each documented technique shortens the window defenders have to detect and interrupt the operation, relative to the equivalent manual technique it replaces.

Technique	Manual equivalent	Autonomous version documented by GTIG
Vulnerability scanning and credential harvesting	Operator runs and monitors scanning tools, manually pivots on results	AI agent plans and executes the full scan-to-harvest sequence from a single prompt [1][2]

Technique	Manual equivalent	Autonomous version documented by GTIG
Error recovery	Operator debugs failed scripts, adjusts approach	Agent performs real-time troubleshooting without operator involvement [1][3]
Evasion	Operator manually configures proxy infrastructure	Agent rotates IP addresses through the victim's own cloud infrastructure automatically [1][2]
Supply-chain trust bypass	Attacker forges or steals a signing credential manually	DUSTMAKER extracts CI/CD OIDC tokens and self-publishes with valid SLSA attestations [1][3][4]
Defeating automated code review	Attacker obfuscates code to evade static analysis	Adversarial prompts embedded in code comments cause LLM reviewers to refuse or misfire [1]

None of the individual techniques in the table is unprecedented; credential harvesting, IP rotation, and OIDC token theft have all been documented independently in prior incidents. What is new is that a single AI system executed the entire left-to-right sequence in one continuous run, without the coordination overhead, fatigue, or decision latency a human operator introduces at each handoff. That compression is consistent with why GTIG frames this as a genuine evolution in adversarial AI use rather than an incremental improvement in tooling, and it is the central reason enterprise identity and credential controls, which can act automatically and continuously, now matter more than detection processes that assume a human-paced attacker on the other end.

Recommendations

Immediate Actions

Security teams should audit any repository, developer workstation, or CI/CD environment for hidden or modified files inside AI assistant configuration directories, including `.claude/`, `.vscode/`, `.cursor/`, and equivalent paths for other coding assistants and IDE extensions, since DUSTMAKER's persistence mechanism specifically depends on those locations going unreviewed [1][3]. Any

organization using self-hosted GitHub Actions runners or comparable CI/CD infrastructure should rotate OIDC signing configurations and review recent package publications for unexpected SLSA-attested releases, given DUSTMAKER's demonstrated ability to extract runner tokens and self-publish trusted-looking packages [1][3][4]. Teams that have added LLM-based code review or security scanning to a CI/CD pipeline should treat that scanner as an attack surface in its own right and confirm it cannot be steered off task by adversarial content embedded in the code it is reviewing.

Short-Term Mitigations

Because both documented campaigns depended on long-lived, broadly scoped credentials, whether cloud infrastructure secrets, developer tokens, or CI/CD signing material, organizations should prioritize moving toward short-lived, narrowly scoped credentials for any workload an AI coding agent or CI/CD pipeline can touch, and should implement continuous rather than periodic monitoring for unusual outbound network activity originating from cloud compute resources, since GTIG's six-hour case specifically relied on the victim's own infrastructure to mask its traffic [1][2]. Package provenance verification should be extended beyond checking that a signature or attestation exists to actively correlating publication events against expected release cadences and known maintainer activity, since a validly signed package published through a stolen OIDC token will otherwise pass automated trust checks without issue [1][3][4]. Given that the underlying attack completed in well under a business day, incident response playbooks built around detection windows measured in hours rather than days should be treated as a baseline requirement rather than an aspirational target for any environment running AI coding assistants against production infrastructure.

Strategic Considerations

Longer term, enterprises should treat AI agent identity and credential governance as a first-class extension of existing identity and access management discipline rather than a bespoke problem requiring entirely new tooling, assigning distinct, attributable identities to agents and CI/CD automation rather than allowing them to operate under shared service accounts or inherited human permissions. Security leadership should budget for the reality that autonomous attack tooling will continue to compress the time between initial compromise and material loss, which argues for investment in automated, policy-driven containment (real-time credential revocation, automatic compute quota enforcement, and anomaly-triggered access suspension) that does not wait on a human analyst to act. Finally, because this incident and the parallel UNC6780 campaign both targeted the AI coding assistant ecosystem specifically, organizations should factor AI-tooling supply-chain risk into vendor and open-source

dependency reviews with the same rigor already applied to traditional software dependencies, rather than treating MCP servers, IDE extensions, and coding-agent plugins as inherently lower risk because of their newer AI framing.

CSA Resource Alignment

This incident sits within ground CSA has already covered in its recent agentic AI security research. CSA's own incident analysis, "Hugging Face's Autonomous AI Agent Breach," documented the first publicly disclosed production breach driven end-to-end by an autonomous AI agent: a malicious dataset abused two code-execution paths in Hugging Face's dataset-processing pipeline, after which the agent carried out privilege escalation and lateral movement across internal infrastructure over a single weekend with no operator in the loop, establishing the same "AI as operational core of an attack chain" pattern that GTIG's six-hour case now extends with a more recent and more thoroughly instrumented example [9]. "Defining Non-Human Identity (NHI)" is the most directly applicable CSA resource for the credential and identity dimension of this incident: it argues that non-human identities lack the deterministic ownership and HR-driven lifecycle events that govern human accounts, leaving them exposed to persistent over-privilege and static credential exposure — the same structural gap that let both DUSTMAKER and the six-hour campaign exploit long-lived, broadly scoped credentials [10]. CSA's own threat-intelligence coverage of comparable incidents reinforces the point from the supply-chain side: "Miasma: Red Hat npm Supply Chain Worm" documents a parallel campaign in which credentials harvested from a compromised maintainer account were used to publish malicious npm packages carrying valid SLSA provenance attestations, demonstrating that supply-chain signing fails once upstream maintainer credentials, rather than the signing infrastructure itself, are compromised [11]; and "Three AI Coding Agents, One GitHub Issue: CI/CD Secrets Exposed" shows that prompt injection delivered through an ordinary developer-facing surface, such as a GitHub issue, can already extract CI/CD secrets from multiple mainstream coding agents, the same class of AI-tooling supply-chain risk this note's Strategic Considerations section urges organizations to take seriously [12]. CSA's survey report "Identity and Access Gaps in the Age of Autonomous AI" supplies an empirical baseline behind that gap, finding that most AI agents lack distinct identities and instead inherit existing permissions, which expands the attack surface through over-privileged access — the same pattern of over-privileged, automatable credential exposure GTIG documented in production [13]. Finally, for organizations building a structured threat-modeling response, CSA's MAESTRO framework and the AI Controls Matrix (AICM) v1.1 provide the layered threat categories and auditable control objectives, spanning identity and access management, supply chain integrity, and application security, needed to translate this note's recommendations into a governance program rather than a one-time remediation [14][15].

References

[1] Google Threat Intelligence Group. "GTIG AI Threat Tracker: From Prompting to Autonomy – The Evolution of Adversarial AI." Google Cloud Blog, September 8, 2026.

[2] The Hacker News. "Autonomous AI Agents Compromise Thousands of Credentials in Under Six Hours." The Hacker News, September 8, 2026.

[3] SiliconANGLE. "Google Says Attackers Used AI Agents to Steal Credentials in Under Six Hours." SiliconANGLE, September 8, 2026.

[4] SANS Internet Storm Center. "TeamPCP Supply Chain Campaign: Update 007 - Cisco Source Code Stolen via Trivy-Linked Breach, Google GTIG Tracks TeamPCP as UNC6780." SANS ISC Diary, September 2026.

[5] Help Net Security. "Threat Actors Are Giving AI Agents a Bigger Role in Cyberattacks." Help Net Security, September 8, 2026.

[6] CyberInsider. "Google Warns Hackers Are Deploying AI Agents in Autonomous Attacks." CyberInsider, September 8, 2026.

[7] Google Threat Intelligence Group. "GTIG AI Threat Tracker: Advances in Threat Actor Usage of AI Tools." Google Cloud Blog, November 5, 2025.

[8] OWASP GenAI Security Project. "OWASP Top 10 for LLM Applications 2025." OWASP, November 18, 2024.

[9] Cloud Security Alliance. "Hugging Face's Autonomous AI Agent Breach." CSA AI Safety Initiative, July 19, 2026.

[10] Cloud Security Alliance. "Defining Non-Human Identity (NHI)." Cloud Security Alliance, July 22, 2026.

[11] Cloud Security Alliance. "Miasma: Red Hat npm Supply Chain Worm." CSA Lab Space, June 3, 2026.

[12] Cloud Security Alliance. "Three AI Coding Agents, One GitHub Issue: CI/CD Secrets Exposed." CSA Lab Space, August 8, 2026.

[13] Cloud Security Alliance. "Identity and Access Gaps in the Age of Autonomous AI." Cloud Security Alliance, March 23, 2026.

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

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