INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-10 18:13 UTC

Claude.ai Shared Chats Weaponized as Malvertising Delivery Rail for MacSync Infostealer

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0300
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	macOS users; Claude.ai shared chat feature (Anthropic); Google Ads platform; major browsers (credential/cookie stores); macOS Keychain
Published	2026-05-10T13:52:15
Discovery Source	Rss

Executive Summary

Threat actors are running an active malvertising campaign that abuses Google Ads and Anthropic's Claude.ai platform to deliver a credential-stealing malware targeting macOS users. Because the attack chain routes through legitimate Google Ads infrastructure and resolves to real claude.ai URLs, standard URL-reputation and domain-filtering controls do not flag it. Any employee searching for Claude AI software and clicking what appears to be an official result may execute the infostealer, exposing browser credentials, session tokens, macOS Keychain secrets, and potentially cryptocurrency wallets to the attacker.

Technical Analysis

This active campaign chains two legitimate-infrastructure abuse techniques to deliver MacSync, a polymorphic, memory-resident macOS infostealer attributed to or closely related to AMOS (Atomic macOS Stealer). Attack chain: (1) Adversaries purchase Google Ads targeting users searching for Claude AI-related terms; ads resolve to legitimate claude.ai URLs, bypassing URL-reputation and domain-inspection controls (T1608.004, T1036.005). (2) Claude.ai's shared chat/Artifacts feature is abused to host malicious redirect logic or payload delivery within the trusted claude.ai domain (T1583.008, T1059.004). (3) The delivered payload is polymorphic and memory-resident, reducing static AV detection efficacy (T1027, T1027.013, T1620). Post-execution targets include browser credential and session cookie stores across major browsers (T1555.003, T1539), macOS Keychain (T1555.001), and crypto wallet data (T1555). Exfiltration observed over standard channels (T1041).

CIS-region IP filtering in the delivery chain indicates operator sophistication and deliberate geographic targeting (T1497.001). No CVE assigned; root cause is platform feature abuse rather than a traditional vulnerability. CWE mappings: CWE-78 (script injection via chat artifacts), CWE-311 (insufficient encryption of exfiltrated data pathways), CWE-522 (insufficiently protected credentials on victim endpoint). No patch is available from Anthropic or Google at this time; no CISA KEV entry. Mitigation is detection and policy-based.

Action Checklist

1. Step 1: Containment. Evaluate restricting access to claude.ai shared chat and Artifacts URLs on macOS endpoints if the shared feature is not business-critical for your organization; if in use, implement monitoring. Notify users not to click Google Ads for AI software tools regardless of apparent destination URL.
2. Step 2: Detection. Hunt for MacSync/AMOS indicators on macOS endpoints: review process execution logs for unsigned or ad-hoc-signed binaries launched from browser download directories; inspect browser history for claude.ai/artifacts URLs followed by binary downloads; query EDR for memory-resident processes without corresponding on-disk binaries; review macOS Keychain access logs and browser credential store access events for anomalous processes; check network telemetry for outbound connections to unknown IPs from browsers or shell processes (T1041 exfiltration pattern).
3. Step 3: Eradication. On confirmed infected endpoints: isolate immediately; revoke and rotate all browser-stored credentials and session tokens across major browsers (Chrome, Firefox, Safari, Brave); rotate macOS Keychain secrets, SSH keys, and API tokens stored on the device; revoke any cryptocurrency wallet keys or seed phrases accessible from the host; re-image the endpoint rather than attempting in-place cleanup given the memory-resident, polymorphic nature of the payload.
4. Step 4: Recovery. Before returning an endpoint to production: validate no persistence mechanisms remain (launch agents, login items, cron jobs, browser extensions added without authorization); confirm credential rotation is complete for all services the user accessed from the affected device; monitor for anomalous authentication events or session replay attempts from unfamiliar IPs for 30 days post-incident.
5. Step 5: Post-Incident. Review and update browser extension and download policies for macOS endpoints; enforce ad-blocking at the DNS or proxy layer to reduce malvertising exposure (CIS Benchmark macOS controls); implement application allowlisting to prevent execution of unsigned binaries from user-writable directories; brief users on the specific risk of clicking sponsored/ad results for software downloads regardless of displayed URL; assess whether macOS Keychain contents are scoped appropriately and whether secrets are rotated on a defined schedule.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior IR leadership and legal/privacy counsel immediately if evidence shows MacSync successfully exfiltrated credentials — specifically if browser Login Data SQLite files or macOS Keychain contents were accessed by the malware process — as this constitutes potential PII/credential breach triggering state breach notification obligations and, if cloud service credentials were exposed, may require notification to downstream service providers or customers.

Recovery Notes	Credential rotation must be treated as the critical path for recovery from this specific campaign — MacSync/AMOS exfiltrates browser-stored credentials and session cookies before any user-visible symptoms appear, meaning stolen sessions may be actively replayed against SaaS, cloud, and internal applications while IR is still in progress. Monitor authentication logs across all services the affected user accessed for a minimum of 30 days post-credential rotation, specifically filtering for logins from IP addresses or ASNs not previously associated with the user, OAuth token grants, and MFA bypass events that could indicate session token replay from pre-rotation exfiltration. Re-enable access to claude.ai only after Anthropic publicly confirms the shared chat/artifact abuse vector has been remediated and your organization has implemented application allowlisting controls that would prevent execution of any binary downloaded from a browser regardless of the source URL.
Forensic Artifacts	macOS browser credential store SQLite files — Chrome: ~/Library/Application Support/Google/Chrome/Default/Login Data and Cookies; Firefox: ~/Library/Application Support/Firefox/Profiles/*/logins.json and cookies.sqlite; Brave: ~/Library/Application Support/BraveSoftware/Brave-Browser/Default/Login Data — these are the primary exfiltration targets of MacSync/AMOS and must be hash-verified before and after incident to confirm access occurred | macOS Unified Log TCC subsystem entries (com.apple.TCC) covering Keychain access events — extract with 'log show --predicate "subsystem == \"com.apple.TCC\"" --last 30d' — will show which non-Apple processes requested and received Keychain access, directly attributing MacSync's credential harvesting activity to a specific process name and PID | Browser download history and navigation history SQLite databases showing the specific claude.ai/artifacts or claude.ai/chat/share URL visited immediately before a binary file download, establishing the malvertising delivery chain from Google Ad click through Anthropic platform abuse to payload execution | LaunchAgent plist files in ~/Library/LaunchAgents/ with creation timestamps matching the infection window — MacSync/AMOS installs persistence here using plausible-looking names mimicking Apple system agents; hash each plist against known-good baselines using 'md5 ~/Library/LaunchAgents/*.plist' | macOS Gatekeeper quarantine extended attribute records — run 'xattr -p com.apple.quarantine ~/Downloads/*' to identify files downloaded from the claude.ai artifact URL, which will contain the originating URL in the quarantine attribute and confirm the delivery vector for this specific campaign

Per-Action IR Details

Step 1: Containment — Block access to claude.ai shared chat and Artifacts URLs at the proxy or DNS layer for managed macOS endpoints until Anthropic confirms the abuse vector is closed. Notify users not to click Google Ads for AI software tools regardless of apparent destination URL.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST IR-4 (Incident Handling), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices), CIS 9.2 — Use DNS Filtering Services (IG2/IG3)

Compensating: Add DNS sinkhole entries for claude.ai/artifacts/* and claude.ai/chat/share/* subpath patterns using Pi-hole or your authoritative internal DNS resolver — create A records pointing to 127.0.0.1 and log all resolution attempts. For proxy-based blocking without a commercial proxy, configure Squid with an ACL rule: 'acl claudeartifacts url_regex -i claude\.ai/(artifacts|chat/share)' and 'http_access deny claudeartifacts'. Push a macOS configuration profile (via MDM or manually) setting a PAC file that routes claude.ai through the sinkhole proxy. Communicate the user advisory via email with a screenshot of what the fraudulent Google Ad looks like — generic 'don't click ads' warnings have poor retention.

Evidence: Before implementing DNS/proxy blocks, capture current DNS query logs from your resolver showing any macOS endpoints that have already resolved claude.ai/artifacts or claude.ai/chat/share paths — this identifies the potential victim pool. Export browser history from potentially exposed endpoints: on macOS, Chrome history is at ~/Library/Application Support/Google/Chrome/Default/History (SQLite — query with 'SELECT url, last_visit_time FROM urls WHERE url LIKE "%claude.ai/artifacts%" OR url LIKE "%claude.ai/chat/share%"'); Safari history at ~/Library/Safari/History.db ('SELECT h.visit_time, i.url FROM history_visits h JOIN history_items i ON h.history_item = i.id WHERE i.url LIKE "%claude.ai%"'). Preserve these before the block so you can correlate who visited artifact URLs in the pre-detection window.

Step 2: Detection — Hunt for MacSync/AMOS indicators on macOS endpoints: review process execution logs for unsigned or ad-hoc-signed binaries launched from browser download directories; inspect browser history for claude.ai/artifacts URLs followed by binary downloads; query EDR for memory-resident processes without corresponding on-disk binaries; review macOS Keychain access logs and browser credential store access events for anomalous processes; check network telemetry for outbound connections to unknown IPs from browsers or shell processes (T1041 exfiltration pattern).

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST SI-4 (System Monitoring), NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST IR-5 (Incident Monitoring), CIS 8.2 (Collect Audit Logs), CIS 8.11 — Collect Network Traffic Flow Logs (IG2/IG3)

Compensating: Enable macOS Unified Log collection and filter for Security and TCC (Transparency Consent and Control) subsystems: run 'log show --predicate "subsystem == \"com.apple.TCC\"" --last 7d | grep -i keychain' to surface unauthorized Keychain access by non-Apple processes. Hunt unsigned binaries launched from ~/Downloads using: 'find ~/Downloads -type f -perm +111 -exec codesign -v {} \; 2>&1 | grep -v "valid on disk"'. For memory-resident process detection without EDR, run 'sudo lsof -nP | grep -v REG | grep -v DIR | grep mem' to surface memory-mapped regions with no on-disk backing. For T1041 exfiltration detection, capture outbound traffic with 'tcpdump -i en0 -nn -w /tmp/capture.pcap "not dst net 17.0.0.0/8 and not dst net 8.8.8.8"' and inspect in Wireshark filtering for large POST requests or raw TCP streams from non-browser PIDs. Deploy the open-source YARA rule sets from the MacOS-Malware-Detection project targeting AMOS/Atomic Stealer signatures against the ~/Downloads and /tmp directories.

Evidence: Collect the following before any remediation action: (1) macOS Unified Log for TCC subsystem covering the last 30 days — 'log collect --last 30d --output /tmp/unified_log.logarchive'; (2) full contents of ~/Library/Logs/DiagnosticReports/ for crash reports tied to unsigned processes; (3) output of 'sqlite3 ~/Library/Application\ Support/Google/Chrome/Default/History ".headers on" ".mode csv" "SELECT url, last_visit_time, title FROM urls WHERE url LIKE \'%claude.ai%\' OR url LIKE \'%download%\'"' for each browser; (4) Keychain access audit via 'security dump-keychain -a 2>/dev/null' to establish baseline, then compare against process names in 'sudo log show --predicate "subsystem == \"com.apple.security.keychain\"" --last 7d'; (5) 'sudo fs_usage -w -f filesys' output filtered for processes reading from ~/Library/Keychains/ or ~/Library/Application Support/*/Login Data paths — MacSync/AMOS targets these SQLite credential stores directly.

Step 3: Eradication — On confirmed infected endpoints: isolate immediately; revoke and rotate all browser-stored credentials and session tokens across major browsers (Chrome, Firefox, Safari, Brave); rotate macOS Keychain secrets, SSH keys, and API tokens stored on the device; revoke any cryptocurrency wallet keys or seed phrases accessible from the host; re-image the endpoint rather than attempting in-place cleanup given the memory-resident, polymorphic nature of the payload.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST SI-3 (Malicious Code Protection), NIST AC-2 (Account Management), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Before re-imaging, perform a forensic acquisition of the full disk using 'sudo dd if=/dev/disk0 of=/Volumes/ExternalDrive/image.dmg bs=4m' or targeted acquisition of ~/Library with 'tar -czf

/Volumes/ExternalDrive/library_backup.tar.gz ~/Library' to preserve evidence. For credential revocation without a PAM or SSO platform: build a checklist from the user's browser saved-password export (Chrome: chrome://settings/passwords export; Firefox: about:logins export) and treat every entry as compromised — this is the ground truth of what MacSync could have exfiltrated. Revoke SSH keys by removing ~/.ssh/authorized_keys entries from all remote hosts the user accesses, and rotate API tokens by querying the user's shell history ('cat ~/.bash_history ~/.zsh_history | grep -E "(token|api_key|secret|password)"') to identify what credentials may have been stored in plaintext. For cryptocurrency wallets, identify wallet application data paths (e.g., Exodus: ~/Library/Application Support/Exodus/exodus.wallet/) and treat any seed phrase stored on-disk as fully compromised.

Evidence: Before isolating the endpoint, capture: (1) running process list with parent-child relationships using 'ps aux --forest' or 'pstree' (install via Homebrew if needed) to document the MacSync process tree and any spawned shells; (2) full network connection state with 'sudo lsof -i -n -P' to capture active C2 connections before network is severed; (3) LaunchAgent and LaunchDaemon plist files from ~/Library/LaunchAgents/, /Library/LaunchAgents/, and /Library/LaunchDaemons/ — MacSync/AMOS frequently installs persistence here; (4) contents of ~/Library/Application Support/Google/Chrome/Default/Login Data, ~/Library/Application Support/Google/Chrome/Default/Cookies, and equivalent paths for Firefox (~/Library/Application Support/Firefox/Profiles/*/logins.json), Safari (~/Library/Cookies/Cookies.binarycookies), and Brave — these are the exact SQLite and binary files MacSync targets for credential and session token extraction; (5) memory dump of suspicious processes using 'sudo osxpmem -o /tmp/memory.aff4' (open-source) to capture any in-memory keys or C2 configuration not written to disk.

Step 4: Recovery — Before returning an endpoint to production: validate no persistence mechanisms remain (launch agents, login items, cron jobs, browser extensions added without authorization); confirm credential rotation is complete for all services the user accessed from the affected device; monitor for anomalous authentication events or session replay attempts from unfamiliar IPs for 30 days post-incident.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST CP-10 (System Recovery and Reconstitution), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts)

Compensating: Persistence validation without EDR: run the open-source 'KnockKnock' tool (by Objective-See) against the re-imaged endpoint before user handback — it enumerates all LaunchAgents, LaunchDaemons, login items, browser extensions, and cron jobs and flags unsigned entries. Enumerate browser extensions across all browsers: 'find ~/Library/Application\ Support/Google/Chrome/Default/Extensions -name manifest.json -exec cat {} \;' and compare extension IDs against the user's known-good list — MacSync campaigns have been observed installing malicious browser extensions as secondary persistence. For session replay monitoring without a SIEM, configure log forwarding from macOS Unified Log to a central rsyslog server ('sudo log stream --predicate "category == \"authentication\"" | nc logserver 514') and set a daily cron job to grep for authentication events from IPs not in your known corporate ranges. For services that issued session tokens (SaaS, cloud consoles), log into each and revoke all active sessions — do not rely on password rotation alone, as MacSync exfiltrates cookies and session tokens that remain valid after password changes.

Evidence: Before returning the endpoint to production, document and retain: (1) output of KnockKnock scan as a signed PDF or JSON export for the incident record; (2) complete browser extension inventory with version hashes — 'find ~/Library/Application\ Support -name manifest.json | xargs md5 > /tmp/extension_hashes.txt'; (3) crontab export for the user and root accounts ('crontab -l; sudo crontab -l'); (4) login items from 'osascript -e "tell application \"System Events\" to get the name of every login item"'; (5) authentication logs from each SaaS service the user accessed, pulled via admin console or API, covering the 30-day monitoring window — specifically look for OAuth token grants or session creations from IP addresses geolocated outside normal user patterns, which would indicate session token replay from the MacSync exfiltration.

Step 5: Post-Incident — Review and update browser extension and download policies for macOS endpoints; enforce ad-blocking at the DNS or proxy layer to reduce malvertising exposure (CIS Benchmark macOS controls); implement application allowlisting to prevent execution of unsigned binaries from user-writable directories; brief users on the specific risk of clicking sponsored/ad results for software downloads

regardless of displayed URL; assess whether macOS Keychain contents are scoped appropriately and whether secrets are rotated on a defined schedule.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-2 (Flaw Remediation), NIST SI-4 (System Monitoring), NIST CM-7 (Least Functionality), CIS 2.3 (Address Unauthorized Software), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 6.3 (Require MFA for Externally-Exposed Applications)

Compensating: For application allowlisting without a commercial tool, enable macOS Gatekeeper enforcement at the strictest level via MDM or 'sudo spctl --global-enable && sudo spctl --master-enable' and deploy an MDM profile that sets 'AllowedApplications' to restrict execution to App Store and identified developer-signed binaries only. For DNS-layer ad-blocking, add category-based block lists targeting advertising networks (EasyList format) to Pi-hole and specifically add entries blocking Google Ads tracking domains (googleadservices.com, googlesyndication.com) that are used in the malvertising delivery chain for this campaign. For Keychain scoping assessment, audit application entitlements with 'codesign -d --entitlements :- /Applications/AppName.app' and identify any app claiming 'keychain-access-groups' entitlements beyond their own bundle ID — this identifies over-privileged apps that could be leveraged similarly to MacSync. Create a Sigma rule targeting macOS process events where a browser process (Chrome, Safari, Firefox) spawns a child process that is not a recognized browser subprocess — this covers the binary execution stage of this specific malvertising kill chain. Conduct a 15-minute tabletop specifically on the scenario: 'user clicks Google Ad for Claude AI, downloads a .dmg' — the existing scenario probably focuses on phishing email, not malvertising through a legitimate ad platform.

Evidence: For the lessons-learned record and to improve future detection, compile: (1) the full timeline reconstructed from browser history SQLite databases, macOS Unified Log, and DNS query logs showing the path from Google Ad click → claude.ai artifact URL → binary download → execution — this is the specific kill chain artifact unique to this campaign; (2) a comparison of macOS Keychain access audit logs pre- and post-incident to quantify what secrets were in scope at time of compromise, informing the secrets rotation scope and Keychain scoping policy changes; (3) documentation of which Google Ad network domains were used in delivery (from network capture) to feed into DNS blocklist improvements; (4) any YARA rule hits from post-incident scanning of other macOS endpoints to determine if additional infections were missed during initial detection; (5) the user's account activity logs from all SaaS services accessed during the compromise window, retained per NIST AU-11 (Audit Record Retention) requirements for your defined retention period, to support any future regulatory breach notification assessment.

Detection Guidance

Behavioral indicators to prioritize given the memory-resident, polymorphic nature of the payload: (1) Process execution: unsigned or ad-hoc-signed binaries launched from ~/Downloads, /tmp, or browser profile directories; shell processes (bash, zsh) spawned as children of browser processes (T1059.004); (2) Memory forensics: EDR alerts on processes with no corresponding on-disk binary or with packed/encrypted memory regions (T1027, T1620); (3) Credential access: anomalous processes accessing the macOS Keychain (security command-line tool invocations outside expected admin workflows), SQLite reads against browser credential stores (Login Data, cookies.sqlite) by non-browser processes (T1555.003, T1555.001, T1539); (4) Network: outbound HTTP/S POST requests from shell or non-browser processes to IPs outside expected SaaS ranges, particularly low-reputation ASNs; (5) Delivery chain: proxy/DNS logs showing user navigation from a Google Ads click (look for gclid parameters) to claude.ai/artifacts/* URLs followed promptly by a file download event; (6) Geographic filtering bypass: if CIS-IP-range users are in your environment, their exposure may be filtered by the operator; focus detection on non-CIS endpoints. No confirmed public IOC hashes are available in the source reporting reviewed; treat any unsigned macOS binary sourced from a claude.ai URL as high-confidence malicious.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`claude.ai/artifacts/*`	Claude.ai Artifacts/shared chat URLs used to host or redirect malicious payloads in this campaign — treat downloads originating from these paths as suspicious	MEDIUM
URL	`Google Ads redirects to claude.ai (gclid parameter present in referrer chain)`	Campaign delivery vector — sponsored search results for Claude AI terms resolving to claude.ai via Google Ads infrastructure	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1583.008** — Malvertising
- **T1027** — Obfuscated Files or Information
- **T1620** — Reflective Code Loading
- **T1555.003** — Credentials from Web Browsers
- **T1027.013** — Encrypted/Encoded File
- **T1539** — Steal Web Session Cookie
- **T1555.001** — Keychain
- **T1555** — Credentials from Password Stores
- **T1497.001** — System Checks
- **T1041** — Exfiltration Over C2 Channel
- **T1566** — Phishing
- **T1059.004** — Unix Shell
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1608.004** — Drive-by Target
- **T1204.002** — Malicious File

NIST-800-53R5

- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **AT-2** — Literacy Training and Awareness
- **SI-8** — Spam Protection
- **CM-7** — Least Functionality

- **SI-10** — Information Input Validation
- **IA-5** — Authenticator Management
- **SC-13** — Cryptographic Protection

OWASP-TOP10-2021

- **A03:2021** — Injection
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **16.10** — Apply Secure Design Principles in Application Architectures
- **5.2** — Use Unique Passwords
- **6.3** — Require MFA for Externally-Exposed Applications
- **8.2** — Collect Audit Logs

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication
- **164.312(e)(1)** — Transmission Security

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.34** — Privacy and protection of personal information
- **A.8.24** — Use of cryptography

NIST-CSF-2

- **DE.CM-01** — Networks and network services are monitored

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1583.008	Malvertising	Resource-Development
T1027	Obfuscated Files or Information	Defense-Evasion
T1620	Reflective Code Loading	Defense-Evasion
T1555.003	Credentials from Web Browsers	Credential-Access
T1027.013	Encrypted/Encoded File	Defense-Evasion

Technique ID	Technique Name	Tactic
T1539	Steal Web Session Cookie	Credential-Access
T1555.001	Keychain	Credential-Access
T1555	Credentials from Password Stores	Credential-Access
T1497.001	System Checks	Defense-Evasion
T1041	Exfiltration Over C2 Channel	Exfiltration
T1566	Phishing	Initial-Access
T1059.004	Unix Shell	Execution
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1608.004	Drive-by Target	Resource-Development
T1204.002	Malicious File	Execution

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/hackers-abuse-google...	T3
Claude Fraud - When Trusted Tools Become the Attack Surface - Blog	https://blog.7ai.com/claude-fraud-malware-campaign-ai-developer-tools	T3
Threat Actors Exploit Claude Artifacts and Google Ads to Target ...	https://healsecurity.com/threat-actors-exploit-claude-artifacts-and...	T3
Windows and macOS Malware Spreads via Fake “Claude Code ...	https://www.bitdefender.com/en-us/blog/labs/fake-claude-code-google...	T3
AMOS Stealer Targets Users with Claude Search Bypassing Apple ...	https://www.linkedin.com/posts/adityamahajan8_macos-clickfix-campai...	T3

Generated 2026-05-10 18:13 UTC by TJS Security Command Center