INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-26 06:08 UTC

# ClickFix + AI Brand Impersonation: ACR Infostealer Delivered via Fake Claude Google Ads Targeting macOS

**THREAT CAMPAIGN** | **HIGH** | CVSS 7.5

| | |
|---|---|
| **SCC Item ID** | SCC-CAM-2026-0365 |
| **Type** | Threat Campaign |
| **Severity** | HIGH |
| **CVSS Base Score** | 7.5 |
| **Affected Products** | macOS users; Claude (Anthropic), brand impersonated; Homebrew, brand impersonated; Google Ads platform abused |
| **Published** | 2026-05-25T20:01:48 |
| **Discovery Source** | Rss |

## Executive Summary

Threat actors are running malicious Google Ads that redirect macOS users searching for Homebrew to a spoofed Anthropic Claude download page. The page instructs users to paste and run a terminal command that installs the ACR infostealer, harvesting saved passwords, browser session tokens, and credentials from infected machines. Any organization with macOS users who download developer tools is at risk of credential theft and subsequent account compromise, with no software vulnerability to patch, the attack succeeds entirely through social engineering.

## Technical Analysis

This campaign chains three techniques: malicious Google Ads (T1583.008) targeting users searching for Homebrew, AI brand impersonation of Anthropic Claude (T1036.005) via a spoofed landing page, and ClickFix-style user-executed payload delivery (T1204.002) that instructs victims to paste a terminal command into macOS shell (T1059.004). Execution installs the ACR infostealer, which harvests credentials from browsers and keychains (T1555, T1555.003), captures screenshots (T1113), steals session cookies (T1539), and may perform keylogging (T1056.001). No CVE is associated. The attack exploits social engineering rather than software flaws. Relevant CWEs: CWE-1021 (UI Redress/Clickjacking-style deception), CWE-116 (Improper encoding enabling command injection via user action), CWE-693 (Protection Mechanism Failure, bypassing endpoint controls). Discovery sourced from Bitdefender Labs and Seceon; attribution is unestablished. macOS is specifically targeted, exploiting comparatively lower EDR coverage relative to Windows enterprise

environments. No patch exists, mitigation is entirely preventive and detection-based.

## Action Checklist

1. Step 1: Containment. Identify macOS endpoints whose users may have searched for Homebrew or Claude in the past 30 days. Isolate any machine where a user reports pasting a terminal command from a browser page. Revoke active sessions and rotate credentials for any accounts accessible from potentially compromised systems. Apply account-level isolation and session revocation per NIST AC-2 (Account Management) and AU-2 (Audit Events) to limit lateral movement from compromised developer machines.
2. Step 2: Detection. Search EDR and macOS Unified Log (log show --predicate) for terminal commands executed from browser-spawned processes. Look for curl, bash, or sh executions with encoded or URL-fetched payloads originating from browser parent processes. Audit keychain access events and browser credential store reads (T1555.003). Review DNS logs for anomalous domains contacted post-terminal execution. Apply NIST SI-4 (system monitoring) and AU-6 (audit record review) processes. Check for processes requesting access to ~/Library/Keychains, ~/Library/Application Support/Google/Chrome, or equivalent browser profile paths.
3. Step 3: Eradication. Block known malicious ad-serving domains and spoofed Claude/Anthropic landing page domains at the DNS and web proxy layers (CIS 4.4, CIS 4.5). Submit block requests to Google Ads abuse reporting for identified fraudulent ad creatives. Deploy or update endpoint detection rules for ClickFix-style terminal payload patterns on macOS. Perform system file analysis per NIST SI-4 (System Monitoring) to scan for ACR infostealer artifacts in user home directories and LaunchAgents persistence paths (~/.zshrc, ~/Library/LaunchAgents/). Remove identified persistence mechanisms.
4. Step 4: Recovery. For any confirmed-compromised endpoint, rotate all stored browser credentials and macOS keychain entries. Invalidate and reissue session tokens for SaaS platforms accessible from the affected machine. Verify no LaunchAgent or cron persistence was established post-infection. Enable macOS Gatekeeper and verify it is enforced via MDM profile (CIS 4.6). Confirm endpoint EDR coverage is active on all macOS assets per CIS 1.1 asset inventory.
5. Step 5: Post-Incident. Conduct targeted security awareness training for developers and macOS users on ClickFix-style terminal command injection lures. Enforce a policy prohibiting users from pasting browser-sourced commands into terminal without peer review. Implement NIST AC-6 (Least Privilege) on macOS endpoints to limit which accounts can execute downloaded or user-pasted shell commands. Review macOS EDR coverage gaps, escalate to parity with Windows endpoint monitoring. Apply CIS 8.2 (Collect Audit Logs) to ensure macOS Unified Log and process execution telemetry feeds into the SIEM.

## IR / Forensic Enrichment

| | |
|---|---|
| **Triage Priority** | IMMEDIATE |
| **Escalation Criteria** | Escalate to CISO and legal counsel immediately if any confirmed-compromised macOS endpoint had access to production secrets, CI/CD pipeline credentials, cloud IAM keys, or customer PII, as harvested browser-stored credentials and Keychain entries may trigger state breach notification obligations and represent active supply chain compromise risk. |

| | |
|---|---|
| **Recovery Notes** | Post-containment, maintain 90-day enhanced monitoring on all macOS endpoints for re-infection attempts via the same ClickFix Google Ads vector — threat actors running paid ad campaigns typically rotate spoofed domains weekly, so DNS blocklists require continuous updating against threat intelligence feeds (OSINT: URLhaus, PhishTank, and Anthropic's official security disclosure page). Verify credential rotation completeness by auditing SaaS platform login histories for the affected user accounts at 7, 14, and 30 days post-incident, specifically watching for logins from new geographic locations, ASNs, or device fingerprints that would indicate harvested session tokens are still in active attacker use. Confirm Gatekeeper enforcement and macOS Unified Log forwarding remain active via weekly MDM compliance checks or manual verification scripts, as developer users frequently disable security controls that interfere with local build tooling. |
| **Forensic Artifacts** | macOS Unified Log TCC subsystem entries (~/.zsh_history, 'log show --predicate "subsystem == com.apple.TCC"'): Records every process that requested access to Keychain, browser credential directories, and sensitive paths — directly maps which processes ACR infostealer used to harvest credentials and confirms the attack occurred. \| Shell history files (~/.zsh_history, ~/.bash_history): Contain the exact ClickFix terminal command pasted by the user from the spoofed Claude landing page, including the curl URL used to fetch the ACR infostealer payload — the single highest-value artifact for reconstructing the initial access chain. \| Browser SQLite credential stores (~/Library/Application Support/Google/Chrome/Default/Login Data, Cookies; ~/Library/Firefox/Profiles/*/key4.db, logins.json; ~/Library/Safari/): These are the specific files ACR infostealer targets for credential harvesting — their access timestamps and any integrity changes confirm active exfiltration and define the full scope of compromised credentials. \| LaunchAgent plist files (~/Library/LaunchAgents/*.plist, /Library/LaunchAgents/*.plist): ACR infostealer establishes persistence via user-space LaunchAgents pointing to scripts or binaries in user-writable directories — recently created or modified plists with ProgramArguments referencing /tmp, ~/Downloads, or encoded shell commands confirm successful persistence establishment. \| DNS query logs from mDNSResponder (macOS Unified Log: 'log show --predicate "process == mDNSResponder"') and network proxy/firewall logs: Capture the C2 domains and exfiltration endpoints contacted by ACR infostealer immediately after the terminal payload executed — domains queried within 0-300 seconds of the suspicious terminal command execution window are high-confidence IOCs for threat intelligence reporting and blocklist submission. |

### Per-Action IR Details

**Step 1: Containment — Identify macOS endpoints whose users may have searched for Homebrew or Claude in the past 30 days. Isolate any machine where a user reports pasting a terminal command from a browser page. Revoke active sessions and rotate credentials for any accounts accessible from potentially compromised systems. Apply NIST AC-17 remote access controls to limit lateral movement from compromised developer machines.**

**NIST Phase:** Containment

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

**Controls:** NIST AC-17 (Remote Access), NIST AC-2 (Account Management), NIST IR-4 (Incident Handling), CIS 6.2 (Establish an Access Revoking Process)

**Compensating:** Without EDR, query browser history directly: run 'sqlite3 ~/Library/Application\ Support/Google/Chrome/Default/History "SELECT url, last_visit_time FROM urls WHERE url LIKE \'%homebrew%\' OR url LIKE \'%claude%\' ORDER BY last_visit_time DESC LIMIT 50;"' on each macOS endpoint. For Safari, parse ~/Library/Safari/History.db similarly. Isolate by disabling the machine's Wi-Fi via MDM or physically disconnecting. Use the free macOS 'security' CLI to list and invalidate keychain entries: 'security find-generic-password -a ' to enumerate

what may have been harvested.

**Evidence:** Capture BEFORE isolating: full browser history exports from Chrome (~/Library/Application Support/Google/Chrome/Default/History), Firefox (~/Library/Application Support/Firefox/Profiles/*/places.sqlite), and Safari (~/Library/Safari/History.db) to document navigation to the spoofed Anthropic/Claude landing page domain. Preserve the macOS Unified Log snapshot covering the 30-day window with 'log collect --last 30d --output /tmp/unified_log_.logarchive'. Export currently active network connections via 'netstat -an > /tmp/netstat_.txt' and 'lsof -i > /tmp/lsof_.txt' before network isolation to capture any live C2 or exfiltration sessions initiated by ACR infostealer post-execution.

**Step 2: Detection — Search EDR and macOS Unified Log (log show --predicate) for terminal commands executed from browser-spawned processes. Look for curl, bash, or sh executions with encoded or URL-fetched payloads originating from browser parent processes. Audit keychain access events and browser credential store reads (T1555.003). Review DNS logs for anomalous domains contacted post-terminal execution. Apply NIST SI-4 (system monitoring) and AU-6 (audit record review) processes. Check for processes requesting access to ~/Library/Keychains, ~/Library/Application Support/Google/Chrome, or equivalent browser profile paths.**

**NIST Phase:** Detection Analysis

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

**Controls:** NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-2 (Event Logging), CIS 8.2 (Collect Audit Logs)

**Compensating:** Run the following macOS Unified Log query to detect browser-spawned terminal activity characteristic of ClickFix delivery: 'log show --predicate "process == \"Terminal\" OR process == \"bash\" OR process == \"sh\" OR process == \"zsh\"" --info --last 30d | grep -E "(curl|bash|sh|python)" > /tmp/terminal_activity.txt'. To detect ACR keychain harvesting specifically, run: 'log show --predicate "subsystem == \"com.apple.securityd\" AND category == \"keychain\"" --info --last 30d > /tmp/keychain_access.txt' and look for non-Apple, non-user-initiated processes. For DNS, use 'log show --predicate "process == \"mDNSResponder\"" --info --last 30d > /tmp/dns_queries.txt' and grep for domains not matching known-good baselines. Deploy the free osquery macOS pack and run: 'SELECT name, path, pid, parent, cmdline FROM processes WHERE cmdline LIKE \'%curl%\' OR cmdline LIKE \'%wget%\';;' to find live or recently spawned download processes.

**Evidence:** Capture BEFORE analysis: macOS Unified Log entries filtered to the TCC (Transparency, Consent, and Control) subsystem — 'log show --predicate "subsystem == \"com.apple.TCC\"" --info --last 30d > /tmp/tcc_log.txt' — which records every application that requested access to the Keychain, browser credential directories, or sensitive paths that ACR infostealer would have queried. Export the full process execution tree around the time of suspected paste-and-run event: 'log show --predicate "process == \"Terminal\" OR process == \"iTerm2\"" --info --debug --last 30d > /tmp/terminal_full.txt'. Collect ~/Library/Application Support/Google/Chrome/Default/Login\ Data (SQLite, stores saved passwords ACR targets), ~/Library/Application Support/Google/Chrome/Default/Cookies, and equivalent Firefox/Safari credential stores as forensic images before any remediation overwrites them.

**Step 3: Eradication — Block known malicious ad-serving domains and spoofed Claude/Anthropic landing page domains at the DNS and web proxy layers (CIS 4.4, CIS 4.5). Submit block requests to Google Ads abuse reporting for identified fraudulent ad creatives. Deploy or update endpoint detection rules for ClickFix-style terminal payload patterns on macOS. Use D3-SFA (System File Analysis) to scan for ACR infostealer artifacts in user home directories and LaunchAgents persistence paths (~/.zshrc, ~/Library/LaunchAgents/). Remove identified persistence mechanisms.**

**NIST Phase:** Eradication

**Reference:** NIST 800-61r3 §3.4 — Eradication

**Controls:** NIST SI-2 (Flaw Remediation), NIST SI-3 (Malicious Code Protection), NIST CM-7 (Least Functionality), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices), CIS 2.3 (Address Unauthorized Software)

**Compensating:** Without a commercial web proxy, push DNS blocklist entries for identified spoofed domains to your internal DNS resolver (BIND, Unbound, or Pi-hole) and verify blocks with 'dig @ '. For ACR LaunchAgent persistence scanning without EDR, run: 'find /Users/*/Library/LaunchAgents/ /Library/LaunchAgents/ /Library/LaunchDaemons/ -name "*.plist" -newer /tmp/reference_date.txt -exec plutil -p {} \;' to identify recently created or modified launch items. Scan for ACR dropper artifacts in user home directories using ClamAV (free): 'clamscan -r /Users/ --log=/tmp/clamav_scan.txt'. Write a YARA rule targeting ACR's known behavior of reading Keychain and browser credential SQLite databases and deploy via the free YARA CLI: match on strings like 'Login Data', 'key4.db', and 'securityd' appearing in unexpected binaries in /tmp, ~/.config, or ~/Downloads.

**Evidence:** Capture BEFORE removing persistence: take a full plist export of every LaunchAgent and LaunchDaemon present — 'find /Users/*/Library/LaunchAgents /Library/LaunchAgents /Library/LaunchDaemons -name "*.plist" -exec cp {} /tmp/launchagent_evidence/ \;' — since ACR infostealer establishes persistence via LaunchAgents pointing to scripts or binaries in user-writable directories. Preserve ~/.zshrc, ~/.bash_profile, and ~/.zprofile as-is before modification, as ClickFix payloads may inject curl-based re-download commands into shell initialization files. Hash all artifacts with 'shasum -a 256 ' before removal to support later threat intelligence sharing and to document IOCs for Google Ads abuse reports.

**Step 4: Recovery — For any confirmed-compromised endpoint, rotate all stored browser credentials and macOS keychain entries. Invalidate and reissue session tokens for SaaS platforms accessible from the affected machine. Verify no LaunchAgent or cron persistence was established post-infection. Enable macOS Gatekeeper and verify it is enforced via MDM profile (CIS 4.6). Confirm endpoint EDR coverage is active on all macOS assets per CIS 1.1 asset inventory.**

**NIST Phase:** Recovery

**Reference:** NIST 800-61r3 §3.5 — Recovery

**Controls:** NIST AC-2 (Account Management), NIST AC-3 (Access Enforcement), NIST IA-5 (Authenticator Management), NIST CM-6 (Configuration Settings), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 5.3 (Disable Dormant Accounts)

**Compensating:** Without MDM, verify Gatekeeper status on each macOS endpoint via: 'spctl --status' (should return 'assessments enabled') and re-enable with 'sudo spctl --master-enable' if disabled. To verify no cron persistence, run: 'crontab -l -u ' for each user account and 'sudo crontab -l' for root. For session token invalidation without a SIEM, log directly into each SaaS platform (GitHub, Slack, AWS Console, GCP, Okta, etc.) accessed from the compromised machine and use each platform's native 'revoke all sessions' feature — prioritize platforms where the compromised user holds admin or developer roles. Run 'security delete-keychain ~/Library/Keychains/login.keychain-db' only after exporting and documenting contents, then rebuild the keychain with a new master password — this removes all credentials ACR may have harvested and re-ingested.

**Evidence:** Capture BEFORE re-imaging or credential rotation: export the full macOS Keychain contents (excluding private keys) using 'security dump-keychain ~/Library/Keychains/login.keychain-db > /tmp/keychain_dump_.txt' to document exactly what credentials were stored and therefore in scope for ACR harvesting — this list defines your mandatory credential rotation scope. Run 'sudo log collect --last 30d' one final time post-eradication to establish a clean baseline log archive for comparison against future anomaly detection. Photograph or screenshot any active SaaS session listing pages before revocation to document concurrent or anomalous sessions that may indicate active attacker use of harvested tokens.

**Step 5: Post-Incident — Conduct targeted security awareness training for developers and macOS users on ClickFix-style terminal command injection lures. Enforce a policy prohibiting users from pasting browser-sourced commands into terminal without peer review. Implement NIST AC-6 (Least Privilege) on macOS endpoints to limit which accounts can execute downloaded or user-pasted shell commands. Review macOS EDR coverage gaps — escalate to parity with Windows endpoint monitoring. Apply CIS 8.2 (Collect Audit Logs) to ensure macOS Unified Log and process execution telemetry feeds into the SIEM.**

**NIST Phase:** Post Incident

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

**Controls:** NIST AC-6 (Least Privilege), NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST AT-2 (Literacy Training and Awareness), NIST AU-2 (Event Logging), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

**Compensating:** For teams without SIEM, configure macOS Unified Log forwarding to a central syslog server using the built-in 'syslog' daemon or the free Filebeat agent (Elastic) pointed at a free-tier OpenSearch or Graylog instance — configure it to capture process execution, TCC events, and LaunchAgent changes. To enforce ClickFix-specific least privilege without MDM, use macOS's built-in parental controls or create a separate non-admin user account for daily work, reserving admin credentials for explicit privilege escalation via 'sudo' with session logging enabled in /etc/sudoers ('Defaults logfile=/var/log/sudo.log'). For developer awareness, create a one-page artifact showing the actual fake Claude landing page screenshot and the exact terminal command format ACR uses — concrete visual examples are significantly more effective than abstract phishing awareness for technical users who consider themselves security-savvy.

**Evidence:** Preserve as post-incident documentation: the full browser history exports showing navigation to the spoofed Anthropic/Claude Google Ad landing page, the exact terminal command text that was pasted and executed (recoverable from shell history: '~/.zsh_history' or '~/.bash_history'), and the complete list of credentials confirmed in scope via the Keychain dump. These artifacts constitute the incident record required by NIST IR-8 (Incident Response Plan) for lessons learned and should be used to build detection signatures — specifically a Sigma rule matching browser process spawning Terminal followed within 60 seconds by a curl or sh command — for deployment in your SIEM or as an osquery scheduled query.

## Detection Guidance

Primary detection surface is macOS process telemetry. Flag any sh, bash, or zsh process spawned with a parent process of a browser (Chrome, Safari, Firefox) or clipboard manager, especially where the command contains curl, wget, or base64 decode chains. Query EDR for macOS processes accessing ~/Library/Keychains/ or browser profile credential stores (T1555.003) from non-browser parent processes. In DNS logs, hunt for newly registered domains impersonating 'claude', 'anthropic', or 'homebrew' with TLD variations. In web proxy logs, flag redirects from Google Ads click-tracking URLs (googleadservices.com, googlesyndication.com) to domains not matching official anthropic.com or brew.sh. Behavioral indicator: user-initiated terminal sessions that immediately fork a network connection (curl/wget to external IP) within seconds of browser activity. NIST SI-4 monitoring and AU-12 audit record generation should capture process execution chains if macOS endpoint logging is properly configured and forwarded to SIEM.

## Indicators of Compromise

| Type | Value | Context | Confidence |
|---|---|---|---|
| DOMAIN | `Specific spoofed Claude/Anthropic domains not confirmed in available source material` | Spoofed landing pages impersonating Anthropic Claude download site — exact domains not disclosed in available sources; monitor for newly registered domains containing 'claude', 'anthropic', or 'homebrew' with variant TLDs | LOW |
| URL | `Malicious Google Ads click-through URLs not confirmed in available source material` | Google Ads redirecting Homebrew search queries to spoofed pages — specific ad URLs not disclosed in available sources | LOW |

## Framework Mappings

### MITRE-ATTACK

- **T1555** — Credentials from Password Stores
- **T1583.008** — Malvertising
- **T1555.003** — Credentials from Web Browsers
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1056.001** — Keylogging
- **T1113** — Screen Capture
- **T1204.002** — Malicious File
- **T1539** — Steal Web Session Cookie
- **T1059.004** — Unix Shell
- **T1566** — Phishing
- **T1557** — Adversary-in-the-Middle

### NIST-800-53R5

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

### CIS-V8

- **6.3** — Require MFA for Externally-Exposed Applications
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **8.2** — Collect Audit Logs

### HIPAA-SECURITY

- **164.312(d)** — Person or Entity Authentication

### SOC2-TSC

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

### ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

### NIST-CSF-2

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

## MITRE ATT&CK Mapping

| Technique ID | Technique Name | Tactic |
|---|---|---|
| T1555 | Credentials from Password Stores | Credential-Access |
| T1583.008 | Malvertising | Resource-Development |
| T1555.003 | Credentials from Web Browsers | Credential-Access |
| T1036.005 | Match Legitimate Resource Name or Location | Defense-Evasion |
| T1056.001 | Keylogging | Collection |
| T1113 | Screen Capture | Collection |
| T1204.002 | Malicious File | Execution |
| T1539 | Steal Web Session Cookie | Credential-Access |
| T1059.004 | Unix Shell | Execution |
| T1566 | Phishing | Initial-Access |
| T1557 | Adversary-in-the-Middle | Credential-Access |

## Sources

| Source | URL | Tier |
|---|---|---|
| **Security News** | https://malware-traffic-analysis.net/2026/05/11/index.html | T3 |
| **Claude Fraud - When Trusted Tools Become the Attack Surface - 7AI** | https://blog.7ai.com/claude-fraud-malware-campaign-ai-developer-tools | T3 |
| **Windows and macOS Malware Spreads via Fake “Claude Code ...** | https://www.bitdefender.com/en-us/blog/labs/fake-claude-code-google... | T3 |
| **macOS Malware Campaign Uses Fake Claude Ads on Google Search** | https://seceon.com/macos-malware-campaign-uses-fake-claude-ads-on-g... | T3 |
| **Cybercriminals are abusing Claude LLM artifacts and Google Ads ...** | https://www.instagram.com/p/DU8E9GYiBqY/ | T3 |

Generated 2026-05-26 06:08 UTC by TJS Security Command Center