INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-02 06:44 UTC

MacSync Stealer Rides Malicious Homebrew Ad to Target macOS Developer Endpoints

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0257
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	macOS (MacBook, Mac Mini); Homebrew package manager users; developer endpoints
Published	2026-05-01T15:01:21
Discovery Source	Rss

Executive Summary

An active malvertising campaign is delivering MacSync Stealer, a credential-stealing payload, to macOS users who search for the Homebrew package manager and click on malicious Google ad placements impersonating the legitimate tool. Developer endpoints are the primary target, placing source code repositories, cloud credentials, API keys, and session tokens at direct risk. Organizations with macOS developer fleets face elevated exposure; macOS infostealer detection remains less mature than Windows-focused endpoint detection solutions, increasing the window of detection latency.

Technical Analysis

MacSync Stealer is delivered via search engine malvertising: threat actors purchase Google ad placements to surface trojanized pages mimicking the legitimate Homebrew installer site (brew.sh). Users who land on the lookalike page are prompted to download and execute an unsigned or unverified installer script, consistent with the common Homebrew installation pattern of piping remote shell scripts directly into bash or zsh. CWE-494 (Download of Code Without Integrity Check) and CWE-345 (Insufficient Verification of Data Authenticity) apply directly to this delivery mechanism. The payload is classified as an infostealer targeting stored credentials, browser session tokens, keychain data, and developer secrets (API keys, SSH keys, cloud credentials). No CVE is assigned; this is a campaign-level threat, not a vulnerability in Homebrew itself. MITRE ATT&CK coverage includes T1204.002 (Malicious File execution), T1059.004 (Unix Shell execution), T1598.003 (Phishing for Information: Search Engines), T1555/T1555.001/T1555.003 (credential access from password stores and browsers), T1539 (session token theft), T1547.011 (plist-based persistence), and T1071.001 (C2 over HTTP/S). Attribution confidence: Low, no threat actor attribution established at this time. Active exploitation of the Google Ads delivery channel is corroborated by AppleInsider reporting dated 2026-03-30 and pending SANS ISC diary

publication.

Action Checklist

1. Step 1: Containment. Identify all macOS developer endpoints in your fleet. Block known malicious domains associated with Homebrew lookalike pages at the DNS and proxy layer. Interim guidance: block any domain containing 'brew' or 'homebrew' that does not resolve to brew.sh (198.199.64.180); specific campaign domains will be published by SANS ISC and major threat intelligence feeds as the campaign develops. If any user reported downloading a Homebrew installer from a Google ad result in the past 60 days (extend the window backward if the campaign is confirmed to have started earlier), isolate that endpoint immediately pending forensic triage. Do not allow the endpoint to reconnect to production systems, code repositories, or cloud environments until cleared.
2. Step 2: Detection. Query EDR telemetry for macOS endpoints for: (a) shell processes (bash, zsh) spawned from browser processes (Safari, Chrome, Firefox) within the last 60 days; (b) new or modified LaunchAgent or LaunchDaemon plist files in ~/Library/LaunchAgents/ or /Library/LaunchDaemons/ (T1547.011); (c) processes making outbound HTTP/S connections to uncommon or newly registered domains shortly after browser activity; (d) access to macOS Keychain files or browser credential stores outside of expected application activity (T1555, T1555.001, T1555.003). If you have network logging, look for DNS queries to domains impersonating 'brew.sh' or containing 'homebrew' with slight variations (typosquatting). Interim priority: focus on behavioral detection (process trees, persistence mechanisms, credential access) rather than hash/domain matching, as malvertising infrastructure rotates rapidly. Updated IOC feeds from SANS ISC and threat intelligence providers will supplement this detection strategy.
3. Step 3: Eradication. For any endpoint confirmed or suspected compromised: (a) revoke and rotate all credentials, API keys, SSH keys, cloud service tokens, and session tokens accessible from that machine; (b) audit and revoke OAuth tokens and active sessions for developer accounts (GitHub, GitLab, AWS, GCP, Azure, npm, PyPI); (c) remove any unsigned LaunchAgent or LaunchDaemon plists not associated with known-good software; (d) re-image the endpoint if forensic triage confirms infostealer execution. Do not attempt to clean and retain an endpoint where credential exfiltration is confirmed.
4. Step 4: Recovery. After credential rotation and endpoint remediation: (a) verify no unauthorized commits, pipeline runs, or cloud resource changes occurred using the compromised credentials; (b) audit cloud IAM logs and SCM audit logs for the window from first possible compromise to containment; (c) restore the endpoint from a known-good image and verify Homebrew is installed only from the legitimate source (brew.sh) using the official curl-based installer with hash verification; (d) monitor rotated credential activity for 30 days for signs of residual unauthorized use.
5. Step 5: Post-Incident. This campaign exposes three specific control gaps: (a) absence of DNS/web filtering rules blocking lookalike domains on developer endpoints; (b) no enforcement of macOS Gatekeeper or notarization requirements for downloaded scripts and installers; (c) developer workflows that normalize piping remote scripts into shell without integrity verification. Remediate by deploying DNS filtering to developer endpoints, enabling Gatekeeper enforcement (security assessments for downloaded content), and establishing a developer security awareness touchpoint covering malvertising and installer verification. Consider adopting a software allowlist or requiring internal mirrors for approved development tools.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and legal counsel immediately if forensic triage confirms credential exfiltration from any endpoint with access to production systems, customer data repositories, or software supply chain signing keys, as this may trigger breach notification obligations under applicable state privacy laws (e.g., CCPA) or sector regulations, and supply chain compromise may require downstream customer notification.
Recovery Notes	Recovery cannot be declared complete until all credentials accessible from the compromised endpoint have been rotated and all SCM, cloud IAM, and package registry audit logs have been reviewed for the full 60-day exposure window — not just the period after the user reported the incident. Monitor all newly issued credentials and SSH keys for 30 days post-rotation using cloud provider alerting (AWS CloudWatch, GCP Cloud Audit Logs, GitHub audit log webhooks) for access from new geographic locations or unusual API call patterns indicative of a threat actor testing exfiltrated credentials. If the compromised developer had write access to any published packages (npm, PyPI, RubyGems), treat those packages as potentially backdoored and coordinate with the package registry's security team to audit recent publish events before recovery is closed.
Forensic Artifacts	macOS LaunchAgent plist at ~/Library/LaunchAgents/ — MacSync Stealer installs persistence via a LaunchAgent plist (MITRE T1547.011) that survives reboots; the plist ProgramArguments key will point to the stealer binary path, typically in a hidden or dot-prefixed directory under the user home or /tmp/ | macOS TCC database at ~/Library/Application Support/com.apple.TCC/TCC.db — records every application that requested access to the Keychain, browser credential stores, and sensitive data; this is the primary forensic record confirming whether MacSync Stealer successfully accessed credential stores (T1555, T1555.001, T1555.003) and which specific stores were queried | Browser download history and HTTP access logs — Safari History.db (~Library/Safari/History.db), Chrome History (~/Library/Application Support/Google/Chrome/Default/History), and Firefox places.sqlite will contain the URL of the malicious Homebrew lookalike page and the downloaded fake installer, establishing the initial access vector and the exact domain visited | macOS Unified Log archive (log collect --last 60d) — contains process creation events, network connection events, and TCC access decisions that reconstruct the full execution chain from browser process to shell spawn to stealer execution; critical for establishing parent-child process relationships confirming the malvertising delivery chain | Cloud provider credential usage logs (AWS CloudTrail, GCP Cloud Audit Logs, GitHub audit log, npm access log) — MacSync Stealer's primary objective is credential theft for developer tools; these logs will show whether exfiltrated API keys, OAuth tokens, or session tokens were used by the threat actor after exfiltration, establishing whether the incident is limited to the endpoint or extends to cloud infrastructure and source code repositories

Per-Action IR Details

Step 1: Containment — Identify all macOS developer endpoints in your fleet. Block known malicious domains associated with Homebrew lookalike pages at the DNS and proxy layer. If any user reported downloading a Homebrew installer from a Google ad result in the past 60 days, isolate that endpoint immediately pending forensic triage. Do not allow the endpoint to reconnect to production systems, code repositories, or cloud environments until cleared.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment, Eradication, and Recovery: Containment Strategy

Controls: NIST IR-4 (Incident Handling), NIST IR-5 (Incident Monitoring), NIST SC-7 (Boundary Protection), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: Build a macOS endpoint inventory using MDM enrollment records or run 'sudo /usr/bin/profiles list' on each host to confirm managed status. For DNS blocking without enterprise tooling, push a hosts-file update via MDM scripting (e.g., Jamf policy or a bash script deployed via Apple Remote Desktop) adding null routes for identified lookalike domains (e.g., 'brew-sh[.]com', 'homebrew-pkg[.]com', 'getbrew[.]sh'). Network isolation for suspect endpoints can be enforced by creating a quarantine VLAN and migrating the MAC address via a managed switch ACL, or by pushing a macOS pf firewall rule via MDM that drops all non-management traffic: 'sudo pfctl -e && echo "block all" | sudo pfctl -f -'.

Evidence: Before isolating, capture a full volatile memory snapshot using osxpmem ('sudo osxpmem -o /tmp/mem.aff4') and preserve: (1) running process list with full path and parent PID ('ps auxww > /tmp/procs.txt'); (2) current network connections ('sudo lsof -i -n -P > /tmp/netconn.txt'); (3) LaunchAgent and LaunchDaemon plist inventory ('find ~/Library/LaunchAgents /Library/LaunchDaemons /Library/LaunchAgents -name "*.plist" -exec ls -la {} \; > /tmp/launch_items.txt'); (4) macOS Unified Log entries for the 60-day window covering browser process spawns ('log collect --last 60d --output /tmp/system_log.logarchive'). Do not reboot or disconnect the network until volatile evidence is preserved, as MacSync Stealer's persistence mechanism and active exfiltration channels will be lost.

Step 2: Detection — Query EDR telemetry for macOS endpoints for: (a) shell processes (bash, zsh) spawned from browser processes (Safari, Chrome, Firefox) within the last 60 days; (b) new or modified LaunchAgent or LaunchDaemon plist files in ~/Library/LaunchAgents/ or /Library/LaunchDaemons/ (T1547.011); (c) processes making outbound HTTP/S connections to uncommon or newly registered domains shortly after browser activity; (d) access to macOS Keychain files or browser credential stores outside of expected application activity (T1555, T1555.001, T1555.003). If you have network logging, look for DNS queries to domains impersonating 'brew.sh' or 'homebrew' with slight variations (typosquatting). IOCs from confirmed reporting should be cross-referenced as they become available from SANS ISC and threat intelligence feeds.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: Signs of an Incident and Incident Analysis

Controls: NIST IR-4 (Incident Handling), NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), NIST SI-3 (Malicious Code Protection), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without EDR, deploy osquery on macOS endpoints and run the following queries: (1) Browser-spawned shells — 'SELECT p.pid, p.name, p.path, p.cmdline, pp.name AS parent_name FROM processes p JOIN processes pp ON p.parent = pp.pid WHERE p.name IN ("bash","zsh","sh") AND pp.name IN ("Google Chrome","Safari","firefox");' (2) LaunchAgent anomalies — 'SELECT * FROM launchd WHERE path LIKE "%LaunchAgents%" OR path LIKE "%LaunchDaemons%";' cross-referenced against a known-good baseline. (3) Keychain access — parse the macOS Unified Log for TCC (Transparency, Consent, Control) access events: 'log show --predicate "subsystem == \"com.apple.TCC\"" --style json --last 60d | grep -i keychain'. For network detection, run Zeek or Suricata on egress traffic and write a Sigma rule matching HTTP POST requests to domains registered within the last 30 days containing 'brew' or 'homebrew' in the hostname. YARA rule targeting MacSync Stealer binary strings should be run via 'yara -r /path/to/macsync.yar /' once IOC signatures are published by SANS ISC.

Evidence: Preserve the following before running detection queries, as queries may alter timestamps: (1) macOS Unified Log archive covering the full 60-day window ('log collect --last 60d'); (2) TCC database at '/Library/Application Support/com.apple.TCC/TCC.db' and '~/Library/Application Support/com.apple.TCC/TCC.db' — this records every app that requested Keychain or credential store access and is the primary forensic record of T1555.001/T1555.003 exploitation; (3) Safari/Chrome/Firefox browser history and download records from '~/Library/Application Support/Google/Chrome/Default/History', '~/Library/Safari/History.db', and '~/Library/Application Support/Firefox/Profiles/*/places.sqlite' to confirm access to Homebrew-lookalike URLs; (4) DNS query logs from your resolver or local '/etc/hosts' modification timestamps; (5) macOS '.bash_history' and '.zsh_history' files for evidence of curl or installer execution commands piped to shell.

Step 3: Eradication — For any endpoint confirmed or suspected compromised: (a) revoke and rotate all credentials, API keys, SSH keys, cloud service tokens, and session tokens accessible from that machine; (b) audit and revoke OAuth tokens and active sessions for developer accounts (GitHub, GitLab, AWS, GCP, Azure, npm, PyPI); (c) remove any unsigned LaunchAgent or LaunchDaemon plists not associated with known-good software; (d) re-image the endpoint if forensic triage confirms infostealer execution. Do not attempt to clean and retain an endpoint where credential exfiltration is confirmed.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication and Recovery: Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AC-2 (Account Management), CIS 5.3 (Disable Dormant Accounts), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Credential revocation workflow for a 2-person team without IAM automation: (1) GitHub — use 'gh auth status' to enumerate active tokens, then revoke all via GitHub Settings > Developer Settings > Personal Access Tokens and Settings > Applications > Authorized OAuth Apps; export audit log via 'gh api /orgs/{org}/audit-log' before revoking. (2) AWS — run 'aws iam list-access-keys --user-name {user}' for each developer identity, then 'aws iam delete-access-key' for all keys associated with the compromised host; enumerate active sessions with 'aws sts get-caller-identity'. (3) SSH keys — collect '~/.ssh/known_hosts' and '~/.ssh/id_*' from the compromised endpoint, then search all internal systems' 'authorized_keys' files for matching public key fingerprints. (4) macOS Keychain — run 'security dump-keychain -d login.keychain-db > /tmp/keychain_dump.txt' to inventory what credentials were stored before wiping. LaunchAgent removal: 'find ~/Library/LaunchAgents /Library/LaunchDaemons -name "*.plist" -exec spctl --assess --verbose {} \;' to identify unsigned items, then remove and run 'launchctl bootout' for each.

Evidence: Before re-imaging, forensically image the full disk using Target Disk Mode and 'dd' or Paladin Edge (free forensic boot environment for macOS): 'sudo dd if=/dev/diskX of=/external/suspect_image.dd bs=4m'. Preserve: (1) the MacSync Stealer binary or dropper from '/tmp/', '~/Downloads/', or the path identified in LaunchAgent plist 'ProgramArguments' key — this is the primary malware artifact; (2) the malicious plist file itself from '~/Library/LaunchAgents/' with full metadata intact (do not copy, use 'cp -p' to preserve timestamps); (3) the macOS Keychain database files ('~/Library/Keychains/login.keychain-db', '~/Library/Keychains/Local Items.db') — these will confirm what credentials MacSync Stealer accessed or exfiltrated; (4) network capture of any active C2 connections taken during the live response phase.

Step 4: Recovery — After credential rotation and endpoint remediation: (a) verify no unauthorized commits, pipeline runs, or cloud resource changes occurred using the compromised credentials; (b) audit cloud IAM logs and SCM audit logs for the window from first possible compromise to containment; (c) restore the endpoint from a known-good image and verify Homebrew is installed only from the legitimate source (brew.sh) using the official curl-based installer with hash verification; (d) monitor rotated credential activity for 30 days for signs of residual unauthorized use.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Eradication and Recovery: 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 AU-11 (Audit Record Retention), NIST CM-3 (Configuration Change Control), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: SCM audit review without a SIEM: (1) GitHub — 'gh api /orgs/{org}/audit-log?phrase=actor:{username}&include=git' to pull all git push, PR merge, and Actions workflow runs attributed to the compromised account during the exposure window; pipe to 'jq' for filtering. (2) AWS CloudTrail — 'aws cloudtrail lookup-events --lookup-attributes AttributeKey=Username,AttributeValue={compromised_user} --start-time {ISO8601_start} --end-time {ISO8601_end}' to enumerate all API calls. (3) Homebrew reinstall integrity check — download the official installer script and verify SHA256 before execution: 'curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh -o /tmp/brew_install.sh && shasum -a 256 /tmp/brew_install.sh' — compare against the hash published at brew.sh before piping to bash. For 30-day monitoring of

rotated credentials, configure AWS CloudWatch alarms on new IAM key usage and GitHub webhook alerts on repository pushes from new IP addresses.

Evidence: Before restoring from image, preserve: (1) GitHub/GitLab audit log export covering the full 60-day exposure window — these logs have retention limits (GitHub retains audit logs for 180 days for Enterprise; standard is shorter) and must be exported immediately; (2) AWS CloudTrail logs for all regions, not just the primary region — MacSync Stealer-exfiltrated AWS keys may have been used to create resources in non-primary regions as a persistence mechanism; (3) GCP/Azure activity logs for developer service accounts; (4) npm and PyPI publish logs if the developer had package maintainer rights — credential exfiltration from a developer endpoint with registry publish access creates a downstream supply chain risk that must be assessed before recovery is declared complete.

Step 5: Post-Incident — This campaign exposes three specific control gaps: (a) absence of DNS/web filtering rules blocking lookalike domains on developer endpoints; (b) no enforcement of macOS Gatekeeper or notarization requirements for downloaded scripts and installers; (c) developer workflows that normalize piping remote scripts into shell without integrity verification. Remediate by deploying DNS filtering to developer endpoints, enabling Gatekeeper enforcement (security assessments for downloaded content), and establishing a developer security awareness touchpoint covering malvertising and installer verification. Consider adopting a software allowlist or requiring internal mirrors for approved development tools.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Lessons Learned and Using Collected Incident Data

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-2 (Flaw Remediation), NIST SI-10 (Information Input Validation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Gatekeeper enforcement hardening (free, MDM-deployable): push a configuration profile via Jamf or Apple Configurator 2 setting 'AllowIdentifiedDevelopers' to false and enabling 'Gatekeeper' assessment for all downloaded content — alternatively run 'sudo spctl --master-enable && sudo spctl --global-enable' via MDM script. DNS filtering without budget: deploy Pi-hole or AdGuard Home on the corporate network and subscribe to the Steven Black hosts list plus a custom blocklist of Homebrew-lookalike domains; for remote developer endpoints push a DNS-over-HTTPS profile (Cloudflare for Teams free tier supports basic content filtering). Installer integrity policy: create a one-page runbook requiring developers to verify SHA256 of any installer before execution using 'shasum -a 256 {file}' against the hash published on the official vendor page — enforce this as a written policy backed by periodic spot-check audits. For software allowlisting, deploy Santa (open-source Google tool for macOS binary whitelisting) in monitor mode initially, then lockdown mode after baseline is established.

Evidence: For the lessons-learned report and control gap documentation, preserve and attach: (1) the original malicious Google ad URL or screenshot if captured by any user (browser history from detection phase); (2) the full timeline of the malvertising campaign from first known user exposure (browser history) to containment (isolation timestamp) — this delta defines your 'dwell time' and informs whether breach notification thresholds were crossed; (3) the LaunchAgent plist and MacSync Stealer binary (if recovered) for signature creation and future detection rule development; (4) DNS query logs showing the typosquatting domain resolution — submit IOCs to SANS ISC and your threat intelligence sharing community (e.g., FS-ISAC if applicable) per NIST IR-6 (Incident Reporting) obligations.

Detection Guidance

Primary detection surfaces for MacSync Stealer on macOS: (1) EDR process tree analysis, flag any shell interpreter (bash, zsh) spawned as a child of a browser process; this is not a normal execution path for legitimate Homebrew installation in most enterprise workflows. (2) LaunchAgent persistence, monitor for new or modified plist files in ~/Library/LaunchAgents/ and /Library/LaunchDaemons/ created by processes other than known software installers (T1547.011). (3) Keychain and credential store access, alert on processes accessing

~/Library/Keychains/, browser profile directories (Chrome: ~/Library/Application Support/Google/Chrome/Default/; Firefox: ~/Library/Application Support/Firefox/Profiles/), or macOS security framework APIs for credential retrieval outside of expected application behavior (T1555, T1555.001, T1555.003). (4) Network telemetry, identify outbound connections to newly registered domains, domains with 'brew' or 'homebrew' in the hostname that do not resolve to brew.sh, or HTTP/S POST requests from shell or Python processes to external IPs shortly after browser-initiated downloads (T1071.001). (5) DNS layer, deploy threat intelligence feed integration to catch lookalike domains; newly registered domains matching Homebrew-themed patterns are a high-signal indicator. Example patterns to monitor: domains containing 'brew' or 'homebrew' registered within the past 30 days, typosquatted variations (e.g., 'brewhome.sh', 'homebrew-install.io'), and domains using lookalike TLDs (.io, .cc, .cloud). Your DNS/WHOIS monitoring solution or threat intelligence feed should flag these automatically. Specific IOC values (file hashes, C2 domains, malicious URLs) should be sourced from SANS ISC diary updates and current threat intelligence feeds as this campaign develops; prioritize behavioral detection over static IOC matching given the dynamic nature of malvertising infrastructure.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://appleinsider.com/articles/26/03/30/that-top-google-result-for-homebrew-could-infect-your-mac`	AppleInsider article corroborating active Google Ads delivery vector for MacSync Stealer malvertising campaign — reference source, not a malicious URL	HIGH
URL	`https://isc.sans.edu/diaryimages/images/2026-04-30-image-MacSync-Stealer-image-01.png`	SANS ISC diary image reference for MacSync Stealer campaign — monitor SANS ISC diary for updated IOCs including C2 domains and payload hashes	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1555.001** — Keychain
- **T1608.006** — SEO Poisoning
- **T1566** — Phishing
- **T1539** — Steal Web Session Cookie
- **T1598.003** — Spearphishing Link
- **T1059.004** — Unix Shell
- **T1555** — Credentials from Password Stores
- **T1566.002** — Spearphishing Link
- **T1071.001** — Web Protocols
- **T1555.003** — Credentials from Web Browsers
- **T1547.011**

- **T1204.002** — Malicious File

NIST-800-53R5

- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-8** — Spam Protection
- **CM-7** — Least Functionality
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control
- **IR-5** — Incident Monitoring

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **6.3** — Require MFA for Externally-Exposed Applications
- **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

NIST-CSF-2

- **DE.CM-01** — Networks and network services are monitored
- **DE.AE-08** — Incidents are declared when adverse events meet the defined incident criteria

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1555.001	Keychain	Credential-Access
T1608.006	SEO Poisoning	Resource-Development
T1566	Phishing	Initial-Access
T1539	Steal Web Session Cookie	Credential-Access

Technique ID	Technique Name	Tactic
T1598.003	Spearphishing Link	Reconnaissance
T1059.004	Unix Shell	Execution
T1555	Credentials from Password Stores	Credential-Access
T1566.002	Spearphishing Link	Initial-Access
T1071.001	Web Protocols	Command-And-Control
T1555.003	Credentials from Web Browsers	Credential-Access
T1547.011		
T1204.002	Malicious File	Execution

Sources

Source	URL	Tier
Security News	https://isc.sans.edu/diaryimages/images/2026-04-30-image-MacSync-St...	T1
Security audit finds issues with 'misuse-prone' Homebrew package ...	https://www.devclass.com/development/2024/08/01/security-audit-find...	T3
Security: Package Manager for MacOS - Reddit	https://www.reddit.com/r/MacOS/comments/1ckwabz/security_package_ma...	T3
While I use Homebrew on my Mac — it is a major risk. I am afraid ...	https://news.ycombinator.com/item?id=34817809	T3
That top Google result for Homebrew could infect your Mac	https://appleinsider.com/articles/26/03/30/that-top-google-result-f...	T3

Generated 2026-05-02 06:44 UTC by TJS Security Command Center