INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-07-03 14:28 UTC

PamStealer macOS Infostealer Abuses PAM API and AppleScript to Harvest Credentials and Crypto Wallets

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0618
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	macOS (Apple Silicon primary target); web browsers (credential and cookie theft); cryptocurrency wallet browser extensions; iCloud Keychain; Maccy clipboard manager (impersonated via typosquatting)
Published	2026-07-03T04:03:37
Discovery Source	Rss

Executive Summary

A newly identified macOS infostealer named PamStealer is distributed through a fake website impersonating Maccy, a legitimate open-source clipboard manager. The malware targets macOS users, particularly those on Apple Silicon, and harvests browser-stored credentials, iCloud Keychain contents, and cryptocurrency wallet data before exfiltrating confirmed-valid credentials to an attacker-controlled server. Organizations with macOS fleets, developer populations, or employees holding cryptocurrency assets face elevated risk of credential theft and potential account compromise. Reporting originates from a single source as of publication; independent technical corroboration has not been confirmed at this time.

Technical Analysis

PamStealer is a two-stage macOS infostealer with no assigned CVE. A CVSS base score of 7.5 (High) is associated with the raw data; the attack vector is local given social-engineering delivery. Initial access is achieved via a typosquatting domain impersonating the Maccy clipboard manager (T1566, T1036.005). The first stage is a compiled AppleScript dropper that bypasses Gatekeeper by leveraging Script Editor (T1059.002; CWE-693, Protection Mechanism Failure). A JavaScript for Automation (JXA) downloader retrieves the second-stage payload (T1059.007; CWE-494, Download of Code Without Integrity Check). The second stage is a Rust-based binary targeting Apple Silicon (arm64); it performs host fingerprinting, environment and locale checks to evade sandbox analysis (T1082, T1497), and masquerades as Finder or System Settings to blend in

(T1036.005). Technically distinguishing behavior: the stealer invokes the macOS PAM API to locally validate harvested passwords before exfiltration (T1056.002, GUI Input Capture via fake password prompt), ensuring only confirmed-correct credentials are transmitted (CWE-522, Insufficiently Protected Credentials; CWE-289, Authentication Bypass by Alternate Name). Targeted data includes browser-stored credentials and cookies (T1555.003), iCloud Keychain contents (T1555.001), cryptocurrency wallet browser extensions, and clipboard data (T1115). Persistence is established via an embedded arm64 binary at boot or logon autostart (T1547). Exfiltration occurs over encrypted HTTP to the C2 domain avenger-sync[.]live (T1041). Configuration is encrypted and the payload is hidden below blank lines in the dropper (T1027). No patch exists; no CVE has been assigned. Attribution to a named threat actor is not established.

Action Checklist

1. Containment: Block the C2 domain avenger-sync[.]live at DNS and perimeter firewall across all macOS endpoints. Additionally, block the typosquatting delivery domain if identified by your threat intelligence feeds. Prioritize Apple Silicon endpoints and systems used by developers or cryptocurrency-active employees.
2. Detection: Query endpoint telemetry for AppleScript execution via Script Editor (osascript or Script Editor process spawning unexpected child processes), JXA downloader activity, and Rust-based binaries masquerading as Finder or System Settings. Review DNS logs for queries to avenger-sync[.]live. Examine launch agent and login item entries (per NIST AU-2, Event Logging) for unknown arm64 binaries. Alert on PAM API invocations outside of normal system authentication flows. Reference MITRE ATT&CK T1059.002, T1059.007, T1056.002, T1547, T1041 for detection rule framing.
3. Eradication: Remove any identified malicious launch agents, login items, or binaries masquerading as Finder or System Settings. Rotate all credentials that may have been accessed on affected endpoints, including browser-stored passwords and iCloud Keychain entries (D3-CRO, Credential Rotation). Revoke and reissue any API keys or tokens stored in affected browsers or keychains. Disable or remove unauthorized software identified through software inventory review (CIS 2.3, Address Unauthorized Software).
4. Recovery: Re-image or perform a clean OS reinstall on confirmed-compromised endpoints where feasible. Verify Gatekeeper and System Integrity Protection are enabled and properly configured. Confirm no persistence mechanisms remain via audit of launch agents, login items, and startup configuration (D3-SICA, System Init Config Analysis). Restore credentials only after endpoint is verified clean. Monitor D3-LAM (Local Account Monitoring) signals for anomalous authentication activity post-remediation.
5. Post-Incident: Enforce application allowlisting or notarization checks to prevent execution of unsigned or unnotarized AppleScript droppers; review macOS endpoint policy against CIS 4.6 (Securely Manage Enterprise Assets and Software). Implement or verify MFA on all externally exposed applications and remote access (CIS 6.3, CIS 6.4, Require MFA for Externally-Exposed Applications and Remote Network Access; D3-MFA, Multi-factor Authentication). Audit user account permissions on macOS endpoints to enforce least privilege (NIST AC-6, Least Privilege; D3-UAP, User Account Permissions). Brief macOS-using staff on typosquatting risks and the danger of downloading software from non-official sources.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to legal, privacy, and executive leadership if forensic evidence confirms iCloud Keychain exfiltration or browser credential theft affecting employee accounts that access regulated systems (PII, PHI, or financial data), as this may trigger breach notification obligations under CCPA, GDPR, or HIPAA; additionally escalate if cryptocurrency wallet seed phrases were stored in affected browser extensions, given irreversible financial loss potential and the active, confirmed C2 infrastructure at avenger-sync[.]live indicating an ongoing campaign.
Recovery Notes	After reimaging and credential rotation, monitor all rotated accounts for authentication attempts using the pre-rotation credentials for a minimum of 30 days — PamStealer exfiltrates validated credentials to an attacker-controlled server, meaning stolen credentials may be weaponized in follow-on attacks (credential stuffing, account takeover) well after the initial infection is remediated. Verify that iCloud Keychain sync is disabled or scoped appropriately on recovered endpoints until new credentials are fully provisioned, to prevent re-sync of compromised secrets from iCloud to a clean device. Establish a 90-day enhanced monitoring period on developer endpoints and any systems holding cryptocurrency-related assets, focusing on PAM API invocations, AppleScript/JXA process execution, and DNS queries to newly registered domains resembling legitimate macOS utility software.
Forensic Artifacts	macOS Unified Log archive (logarchive) from affected endpoints covering 72 hours prior to detection — specifically filter for osascript, Script Editor, and unexpected child process spawning under Finder or System Settings process names, which PamStealer used to masquerade its Rust binary | Browser Login Data SQLite databases from Chrome (`~/Library/Application Support/Google/Chrome/Default/Login Data`) and Firefox (`~/Library/Application Support/Firefox/Profiles/*/logins.json`) in their pre-eradication state, capturing the credential stores PamStealer targeted for harvesting | macOS PAM configuration directory `/etc/pam.d/` — examine each service file for unauthorized module insertions (e.g., a malicious `.so` library path added to the `sudo` or `login` PAM stack) left behind by PamStealer's PAM API abuse | Launch agent plist files and associated binaries from `/Library/LaunchAgents`, `/Library/LaunchDaemons`, and `~/Library/LaunchAgents` — PamStealer establishes persistence here; capture full plist content, binary SHA-256 hash, code signing status (`codesign -dvvv `), and Gatekeeper assessment (`spctl -a -v `) for each unknown entry | Network packet capture (pcap) of traffic to avenger-sync[.]live collected prior to DNS/firewall block, including TLS SNI fields and any unencrypted credential payloads, plus DNS resolver query logs showing first-resolution timestamps and all internal endpoints that queried the C2 or typosquatting delivery domain

Per-Action IR Details

Containment — Block the C2 domain avenger-sync[.]live at DNS and perimeter firewall across all macOS endpoints. Additionally, block the typosquatting delivery domain if identified by your threat intelligence feeds. Prioritize Apple Silicon endpoints and systems used by developers or cryptocurrency-active employees.

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: On macOS endpoints without enterprise DNS filtering, push a local /etc/hosts entry mapping avenger-sync[.]live to 127.0.0.1 via a shell script distributed by MDM or SSH: `sudo sh -c 'echo "127.0.0.1 avenger-sync.live" >> /etc/hosts'`. At the perimeter, apply a pfSense or OPNsense block rule for the C2 FQDN and any resolved IPs. Use `dig avenger-sync.live` to capture current A/CNAME records before blocking, and log those IPs for

further firewall rules.

Evidence: Before pushing block rules, capture active network connections from all potentially exposed Apple Silicon endpoints using `netstat -an | grep ESTABLISHED` or `lsof -i -n -P` and export full output. Run `sudo tcpdump -i en0 -w /tmp/pamsstealer_pre_block.pcap host avenger-sync.live` for 60 seconds on any endpoint suspected of active infection to capture in-flight exfiltration traffic, credential payloads, and TLS SNI metadata. Query DNS server query logs for historical resolution of avenger-sync[.]live to establish first-contact timestamps and identify all endpoints that beaconed. Preserve macOS Unified Log entries related to network connections: `log show --predicate 'subsystem == "com.apple.network"' --last 24h > /tmp/network_unified.log`.

Detection — Query endpoint telemetry for AppleScript execution via Script Editor (osascript or Script Editor process spawning unexpected child processes), JXA downloader activity, and Rust-based binaries masquerading as Finder or System Settings. Review DNS logs for queries to avenger-sync[.]live. Examine launch agent and login item entries (per NIST AU-2 — Event Logging) for unknown arm64 binaries. Alert on PAM API invocations outside of normal system authentication flows. Reference MITRE ATT&CK T1059.002, T1059.007, T1056.002, T1547, T1041 for detection rule framing.

NIST Phase: Detection Analysis

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

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

Compensating: Without EDR, enable macOS Unified Logging and run: `log show --predicate 'process == "osascript" OR process == "Script Editor"' --last 48h` to surface PamStealer's AppleScript/JXA dropper execution. Check for unexpected arm64 Mach-O binaries in launch agent paths: `find /Library/LaunchAgents /Users/*/Library/LaunchAgents -name '*.plist' -exec plutil -p {} \;` and cross-reference binary paths with `file ` to confirm arm64 architecture. Use `eslogger exec` (macOS 13+) or the free OpenBSM audit framework (`sudo praudit /dev/auditpipe`) to monitor PAM module invocations. Write a YARA rule targeting Rust PE headers and the PamStealer string patterns (e.g., iCloud Keychain API calls, wallet extension paths) and scan with: `yara -r pamsstealer.yar /Applications /Users/*/Library`.

Evidence: This is an analysis step that does not alter live state; however, capture volatile data concurrently before any downstream containment or eradication actions. Specifically preserve: macOS Unified Log for the past 72 hours (`log collect --last 72h --output /tmp/unified_log.logarchive`); a snapshot of all running processes and their binary paths (`ps aux > /tmp/ps_snapshot.txt`); the full contents of `/Library/LaunchAgents`, `/Library/LaunchDaemons`, `~/Library/LaunchAgents`, and Login Items via `sfltool dumpbtm`; and PAM configuration files at `/etc/pam.d/` to detect unauthorized module insertions. Hash all suspicious binaries with `shasum -a 256` before any removal.

Eradication — Remove any identified malicious launch agents, login items, or binaries masquerading as Finder or System Settings. Rotate all credentials that may have been accessed on affected endpoints, including browser-stored passwords and iCloud Keychain entries (D3-CRO — Credential Rotation). Revoke and reissue any API keys or tokens stored in affected browsers or keychains. Disable or remove unauthorized software identified through software inventory review (CIS 2.3 — Address Unauthorized Software).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-2 (Account Management), NIST SI-2 (Flaw Remediation), CIS 2.3 (Address Unauthorized Software), CIS 5.3 (Disable Dormant Accounts)

Compensating: Use `launchctl list` and `launchctl remove ` to unload and remove malicious launch agents without rebooting. Delete malicious plist files from `/Library/LaunchAgents` and `~/Library/LaunchAgents` and confirm removal with `ls -la`. For credential rotation without a PAM, use Safari's built-in password manager audit (`Security > Passwords`) and Chrome/Firefox's saved passwords export to enumerate accounts requiring rotation. For iCloud Keychain, use `security find-generic-password` and `security delete-generic-password` CLI commands on confirmed-compromised entries. Cryptocurrency wallet operators must treat all seed phrases stored in browser extensions (MetaMask, Phantom, etc.) on affected endpoints as fully compromised and generate new wallets.

Evidence: BEFORE removing any binary or launch agent plist, and BEFORE rotating any credential: acquire a full memory image using `osxpmem` or `MacQuisition` to capture PAM API hooking artifacts, decrypted credential buffers in memory, and any injected code in browser processes. Capture browser profile directories in their current state (`~/Library/Application Support/Google/Chrome/Default/`, `~/Library/Application Support/Firefox/Profiles/`) — these contain the Login Data SQLite database and cookies that PamStealer targeted. Export the current keychain state: `security dump-keychain -d login.keychain-db > /tmp/keychain_dump_pre_eradication.txt` (requires user auth, preserves record of what was accessible). Document all identified malicious binary SHA-256 hashes and plist contents before deletion for IOC sharing and post-incident reporting.

Recovery — Re-image or perform a clean OS reinstall on confirmed-compromised endpoints where feasible. Verify Gatekeeper and System Integrity Protection are enabled and properly configured. Confirm no persistence mechanisms remain via audit of launch agents, login items, and startup configuration (D3-SICA — System Init Config Analysis). Restore credentials only after endpoint is verified clean. Monitor D3-LAM (Local Account Monitoring) signals for anomalous authentication activity post-remediation.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST CM-2 (Baseline Configuration), NIST SI-2 (Flaw Remediation), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 7.3 (Perform Automated Operating System Patch Management)

Compensating: For teams without MDM, verify Gatekeeper status via `spctl --status` (should return `assessments enabled`) and SIP via `csrutil status` (should return `enabled`). Use Apple Configurator 2 (free) to restore a known-good macOS image on Apple Silicon via DFU mode, bypassing any persistent firmware-level implants. Post-reinstall, validate launch agent directories are empty: `find /Library/LaunchAgents /Library/LaunchDaemons ~/Library/LaunchAgents -type f`. Monitor local authentication logs via `log show --predicate 'subsystem == "com.apple.securityd"' --last 24h` for repeated PAM authentication failures or unusual sudo invocations that might indicate credential reuse by the attacker on rotated accounts.

Evidence: Before reimaging, acquire a complete forensic image of the endpoint storage using `dd` or a write-blocker-attached disk image to preserve evidence for potential legal proceedings. Capture a final process listing, network connection state, and open file handles: `lsof -n > /tmp/lsof_pre_reimage.txt`. Confirm the integrity of macOS system files post-reinstall using `sudo /usr/bin/csrutil authenticate --show` and verify no unauthorized kernel extensions are loaded: `kextstat | grep -v com.apple`. After credential restoration, baseline authentication event logs in macOS Unified Log for the recovered endpoint to establish a clean reference point for anomaly detection.

Post-Incident — Enforce application allowlisting or notarization checks to prevent execution of unsigned or unnotarized AppleScript droppers; review macOS endpoint policy against CIS 4.6 (Securely Manage Enterprise Assets and Software). Implement or verify MFA on all externally exposed applications and remote access (CIS 6.3, CIS 6.4 — Require MFA for Externally-Exposed Applications and Remote Network Access; D3-MFA — Multi-factor Authentication). Audit user account permissions on macOS endpoints to enforce least privilege (NIST AC-6 — Least Privilege; D3-UAP — User Account Permissions). Brief macOS-using staff on typosquatting risks and the danger of downloading software from non-official sources.

NIST Phase: Post Incident

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

Controls: NIST AC-6 (Least Privilege), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.4 (Require MFA for Remote Network Access), CIS 6.5 (Require MFA for Administrative Access), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without an MDM, distribute a macOS configuration profile via Apple Configurator 2 or direct script that enforces Gatekeeper to 'App Store and identified developers' (`sudo spctl --global-enable && sudo spctl --master-enable`). To detect future typosquatting download attempts, configure DNS query logging on your recursive resolver and alert on newly registered domains resembling tools common in your macOS fleet (use free DNSTwist against an inventory of approved software names). For least-privilege enforcement, audit local admin membership on all endpoints: `dscl . -read /Groups/admin GroupMembership` and remove non-essential admin rights. Run a YARA scan monthly using updated rules for Rust-compiled macOS infostealers in the `/Applications` and `~/Downloads`

directories.

Evidence: For the lessons-learned record, consolidate all forensic artifacts collected during the incident: memory images, pre-eradication keychain dumps, browser profile copies, launch agent plists, and pcap files from pre-block C2 traffic. Document the full timeline from initial typosquatting download of the fake Maccy installer through PAM API abuse, iCloud Keychain access, and credential exfiltration to avenger-sync[.]live. Preserve SHA-256 hashes of all malicious binaries and plist files as organizational IOCs for future detection rule updates and threat intelligence sharing (e.g., submission to CISA or an ISAC). Archive macOS Unified Log collections from all affected endpoints per your retention policy (NIST AU-11 — Audit Record Retention) to support any regulatory breach notification requirements if harvested credentials included employee PII.

Detection Guidance

Primary behavioral indicators: (1) Script Editor or osascript spawning unexpected child processes, particularly processes named Finder, System Settings, or other system-sounding names that are not located in their standard paths. (2) DNS or proxy logs showing queries or connections to avenger-sync[.]live. (3) Launch agent or login item entries referencing unknown arm64 binaries, audit /Library/LaunchAgents, ~/Library/LaunchAgents, and /Library/LaunchDaemons. (4) PAM API calls (pam_authenticate) invoked outside of expected system authentication contexts, particularly from non-system processes presenting GUI password prompts. (5) Outbound encrypted HTTP connections to avenger-sync[.]live from macOS endpoints. (6) JXA (osascript with JavaScript) execution downloading remote payloads. IOCs confirmed in source reporting: C2 domain, avenger-sync[.]live (confidence: medium, single-source). Delivery mechanism, typosquatting site impersonating Maccy clipboard manager (specific domain not confirmed in available reporting). MITRE ATT&CK mapping supports detection rule development across T1059.002, T1059.007, T1056.002, T1547, T1082, T1497, T1041, T1555.001, T1555.003. Per NIST AU-2 (Event Logging) and CIS 8.2 (Collect Audit Logs), confirm macOS endpoint logging covers process execution, network connections, and authentication events. Note: source corroboration is single-source at time of writing; treat IOCs as medium confidence pending additional vendor or researcher confirmation.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`avenger-sync[.]live`	C2 exfiltration endpoint — PamStealer transmits harvested credentials via encrypted HTTP to this domain, per source reporting	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1204.002** — Malicious File
- **T1539** — Steal Web Session Cookie
- **T1115** — Clipboard Data
- **T1497** — Virtualization/Sandbox Evasion
- **T1553.001** — Gatekeeper Bypass

- **T1027** — Obfuscated Files or Information
- **T1082** — System Information Discovery
- **T1566** — Phishing
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1059.002** — AppleScript
- **T1056.002** — GUI Input Capture
- **T1547** — Boot or Logon Autostart Execution
- **T1555.003** — Credentials from Web Browsers
- **T1555.001** — Keychain
- **T1059.007** — JavaScript
- **T1041** — Exfiltration Over C2 Channel

NIST-800-53R5

- **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
- **CM-7** — Least Functionality
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control
- **IA-5** — Authenticator Management

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures

CIS-V8

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

HIPAA-SECURITY

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

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures
- **CC9.2** — Manages risks associated with vendors and business partners

ISO-27001-2022

- **A.5.21** — Managing information security in the ICT supply chain
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1204.002	Malicious File	Execution
T1539	Steal Web Session Cookie	Credential-Access
T1115	Clipboard Data	Collection
T1497	Virtualization/Sandbox Evasion	Defense-Evasion
T1553.001	Gatekeeper Bypass	Defense-Evasion
T1027	Obfuscated Files or Information	Defense-Evasion
T1082	System Information Discovery	Discovery
T1566	Phishing	Initial-Access
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1059.002	AppleScript	Execution
T1056.002	GUI Input Capture	Collection
T1547	Boot or Logon Autostart Execution	Persistence
T1555.003	Credentials from Web Browsers	Credential-Access
T1555.001	Keychain	Credential-Access
T1059.007	JavaScript	Execution
T1041	Exfiltration Over C2 Channel	Exfiltration

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/07/pamstealer-uses-fake-maccy-sites-...	T2
MacStealer macOS Malware Detection: Novel Malicious ...	https://socprime.com/blog/macstealer-macos-malware-detection-novel-...	T3

DISCLAIMER

This intelligence report is produced by Tech Jacks Solutions Security Command Center (SCC) for informational purposes only. It does not constitute professional security advice, legal counsel, or an incident response engagement. The information herein is derived from publicly available sources and AI-assisted analysis; while every effort is made to ensure accuracy, Tech Jacks Solutions makes no warranties regarding completeness or timeliness. Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-07-03 14:28 UTC by TJS Security Command Center