INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-02 14:01 UTC

FlutterShell Backdoor Weaponizes Flutter's Architecture to Evade Apple Notarization and Hit macOS at Scale

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0392
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	macOS systems; applications masquerading as PodcastsLounge, PDF-Brain, PDF-Ninja; Google Ads platform abused as delivery vector
Published	2026-06-02T10:00:31+00:00
Discovery Source	Rss:T1 Threatintel

Executive Summary

A financially-motivated threat group designated CL-CRI-1089 is actively distributing a macOS backdoor called FlutterShell through malicious Google Ads, using fake applications that passed Apple's notarization process and evaded major antivirus detections at the time of initial analysis. The malware's architecture places all malicious logic on remote attacker-controlled servers, making it invisible to standard static analysis and endpoint security tools. Organizations with macOS fleets whose employees use Google search or download productivity applications face a credible, active risk of credential theft, document exfiltration, and persistent backdoor access.

Technical Analysis

FlutterShell is a macOS backdoor attributed to threat cluster CL-CRI-1089 (Unit 42 threat cluster designation), delivered via malicious Google Ads purchased through shell companies to bypass ad-network vetting. The malware exploits Flutter's WebView-to-native bridge architecture (T1059.007, T1059.004), hosting payload logic remotely on attacker-controlled infrastructure rather than embedding it in the distributed binary. This design satisfies Apple's notarization requirements at submission time (T1553.001) and evades static analysis, as the binary itself is functionally clean. Three known variants masquerade as PodcastsLounge, PDF-Brain, and PDF-Ninja (T1036.005). Capability progression has moved from adware delivery to full backdoor functionality including file system enumeration (T1083), keylogging/input capture (T1056), data exfiltration via web protocols (T1041, T1567, T1071.001), and document classification and selective exfiltration in newer variants, suggesting

adversary use of automated filtering logic. Ingress tooling arrives via remote file copy (T1105). Infrastructure relies on attacker-controlled web services (T1583.006) and digitally signed certificates (T1588.004). Obfuscation techniques include software packing (T1027.002) and embedded payload obfuscation (T1027.009). Relevant weaknesses: CWE-494 (Download of Code Without Integrity Check), CWE-829 (Inclusion of Functionality from Untrusted Control Sphere), CWE-345 (Insufficient Verification of Data Authenticity). No CVE assigned. No vendor patch available. Unit 42 (Palo Alto Networks) is the primary discovery source.

Action Checklist

1. Step 1: Containment. Audit macOS endpoints for the presence of PodcastsLounge, PDF-Brain, and PDF-Ninja application bundles; quarantine any discovered instances immediately. Block outbound connections to known malicious domains and newly registered domains from macOS hosts via firewall policy (NIST SC-7, CIS 4.4). Specific IOC domains and IPs are available in Unit 42's technical report; monitor threat intel feeds for updates. Enforce application allowlisting on macOS assets to prevent execution of unsigned or unvetted third-party applications (NIST CM-7, CIS 2.3).
2. Step 2: Detection. Query EDR telemetry for macOS processes spawning shell commands (bash, sh, zsh) from within Flutter-based application bundles. Monitor for outbound HTTPS traffic from applications in /Applications or ~/Downloads to domains not in your approved inventory (NIST AU-6, NIST SI-4, CIS 8.2). Hunt for applications that passed Apple notarization but retrieve executable logic from remote URLs post-launch; behavioral indicators include WebView processes making requests to non-CDN, non-vendor endpoints shortly after first execution (T1105, T1071.001). Review Google Ads click-through traffic logs if your environment has web proxy visibility. Deploy file integrity monitoring (FIM) on /Applications and /Library/LaunchAgents to detect persistence mechanism installation. Prioritize runtime behavior detection via EDR: monitor for WebView-to-shell-bridge patterns and outbound HTTPS from application processes to non-approved domains.
3. Step 3: Eradication. Remove all identified FlutterShell variants and their persistence mechanisms (LaunchAgents, LaunchDaemons, login items) from affected hosts. Reset all credentials accessible from compromised macOS endpoints, prioritizing browser-stored passwords, SSH keys, and API tokens. Revoke and reissue any certificates or secrets stored on affected systems. Block identified IOC domains and IPs at perimeter (NIST SC-7). Check Apple's published revocation list at https://support.apple.com/en-us/HT202491 for certificate revocation status. Contact Apple Security if you discover additional undocumented variants. Monitor Apple's security updates for official notarization revocation for these specific app bundles.
4. Step 4: Recovery. Reimage affected macOS endpoints from a known-good baseline rather than attempting in-place cleanup, given the backdoor's remote payload model. Validate that no lateral movement occurred from compromised macOS endpoints to Windows or Linux systems via shared credentials or network shares (NIST IR-4). Monitor previously affected user accounts for anomalous authentication events for a minimum of 30 days post-remediation (NIST AU-6). Confirm EDR coverage is active and current on all macOS assets before returning to production.
5. Step 5: Post-Incident. This campaign exploited gaps in application vetting (no enforcement of application allowlisting), ad-network trust (no controls restricting employee software downloads from search ads), and static-analysis-dependent detection tooling (no behavioral or network-based detection for post-launch payload retrieval). Address these gaps: implement macOS application allowlisting enforced via MDM (NIST CM-7, CIS 2.3), deploy DNS-layer filtering to catch C2 beacon attempts (NIST SC-20), establish behavioral detection rules for WebView-to-shell-bridge patterns in your EDR (NIST SI-4), and

add user awareness training specific to malvertising as a delivery vector (NIST AT-2). Review your Google Ads and browser extension policies for macOS users.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to legal, privacy counsel, and executive leadership immediately if forensic analysis of the FlutterShell C2 communications confirms exfiltration of credentials, PII, PHI, or IP from compromised macOS endpoints, as this triggers breach notification obligations under applicable state, federal, or international privacy regulations (GDPR Art. 33, HIPAA §164.410, state breach notification statutes); also escalate if lateral movement from compromised macOS hosts to production Windows or Linux infrastructure is confirmed, or if the team lacks macOS forensic capability to determine the scope of C2 data exchange.
Recovery Notes	Reimage all confirmed-compromised macOS endpoints from MDM-managed clean baselines rather than attempting in-place remediation, because FlutterShell's remote payload model means the full extent of delivered functionality cannot be determined through static analysis alone, and residual attacker-controlled logic may persist in application caches or staged locations not covered by LaunchAgent removal. Rotate all credentials — browser-stored passwords, SSH private keys, API tokens, and cloud CLI credentials — that were accessible on affected endpoints during the infection window, and revoke any OAuth tokens or session cookies that may have been harvested by the backdoor's remote payload. Monitor all previously compromised user accounts and any systems they authenticated to during the infection window for a minimum of 30 days post-reimage, specifically watching for impossible travel events, new MFA device enrollment, and anomalous API calls in cloud platforms (AWS CloudTrail, GCP Audit Logs, Azure Activity Log) that could indicate credential reuse by CL-CRI-1089 post-eradication.
Forensic Artifacts	macOS Quarantine Extended Attribute database (~/.local/share/recently-used.xbel and com.apple.quarantine xattr on the app bundle): records the originating Google Ads URL, download browser, and exact timestamp of the malicious PodcastsLounge, PDF-Brain, or PDF-Ninja bundle retrieval — the primary artifact linking the infection to the malvertising delivery vector. | macOS Unified Log (collected via `log collect` or `log show`) filtered for the Flutter WKWebView subsystem and process tree: captures the precise moment the notarized Flutter app's WebView component made its first outbound request to the attacker-controlled remote payload server, establishing the T1105 ingress tool transfer event timeline. | LaunchAgent and LaunchDaemon plist files in ~/Library/LaunchAgents, /Library/LaunchAgents, and /Library/LaunchDaemons written during or after first execution of the malicious app bundle: CL-CRI-1089's persistence mechanism specific to macOS, timestamped to the infection window and containing the C2 callback configuration. | macOS TCC (Transparency, Consent, and Control) database at ~/Library/Application Support/com.apple.TCC/TCC.db: records every sensitive permission (Full Disk Access, keychain access, microphone, camera, screen recording) that the FlutterShell bundle requested or was granted, directly scoping the data accessible to the remote payload during execution. | Network packet capture (pcap) of outbound HTTPS connections from the Flutter app bundle's process — specifically the initial C2 beacon pattern (timing, JA3/JA3S TLS fingerprint, SNI value, User-Agent header) and any subsequent payload retrieval or data exfiltration sessions — tied to the specific process ID of the malicious app via `lsof -i -n -P -p ` output preserved at time of detection.

Per-Action IR Details

Step 1: Containment — Audit macOS endpoints for the presence of PodcastsLounge, PDF-Brain, and PDF-Ninja application bundles; quarantine any discovered instances immediately. Block outbound connections to unknown or newly registered domains from macOS hosts via firewall policy (NIST SC-7, CIS 4.4). Enforce application allowlisting on macOS assets to prevent execution of unsigned or unvetted third-party applications (NIST CM-7, CIS 2.3).

NIST Phase: Containment

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

Controls: NIST SC-7 (Boundary Protection), NIST CM-7 (Least Functionality), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 2.3 (Address Unauthorized Software)

Compensating: Run `find /Applications ~/Applications -name 'PodcastsLounge.app' -o -name 'PDF-Brain.app' -o -name 'PDF-Ninja.app' 2>/dev/null` across all macOS endpoints via SSH or MDM remote command (Jamf, Mosyle, or Kandji free tiers). For network blocking without a SIEM, push a hosts-file or DNS-resolver block via MDM configuration profile for any C2 domains identified in threat intel. Use Little Snitch (free trial) or the native macOS pf firewall (`sudo pfctl -e` with a custom anchor rule) to block outbound traffic from the identified app bundle identifiers. For allowlisting, enable Gatekeeper enforcement at the MDM profile level: `spctl --master-enable` and set `spctl --global-enable` for App Store and identified developer restriction.

Evidence: Before quarantining, capture the full application bundle contents: `cp -R /Applications/PodcastsLounge.app /path/to/evidence/` and preserve with `shasum -a 256` hashes. Extract and preserve the embedded Info.plist (`plutil -p /Applications/PodcastsLounge.app/Contents/Info.plist`) to document the bundle identifier, CFBundleIdentifier, CFBundleExecutable, and LSMinimumSystemVersion values. Capture the Gatekeeper assessment record: `spctl -a -vv /Applications/PodcastsLounge.app` to confirm notarization chain. Pull the quarantine extended attribute log: `xattr -l /Applications/PodcastsLounge.app` — this records the originating URL from the Google Ads download, the quarantine timestamp, and the browser that fetched it. Preserve the macOS Unified Log entries for the installation window: `log collect --start '2024-01-01' --output /tmp/fluttershell_unified.logarchive`.

Step 2: Detection — Query EDR telemetry for macOS processes spawning shell commands (bash, sh, zsh) from within Flutter-based application bundles. Monitor for outbound HTTPS traffic from applications in /Applications or ~/Downloads to domains not in your approved inventory (NIST AU-6, NIST SI-4, CIS 8.2). Hunt for applications that passed Apple notarization but retrieve executable logic from remote URLs post-launch — behavioral indicators include WebView processes making requests to non-CDN, non-vendor endpoints shortly after first execution (T1105, T1071.001). Review Google Ads click-through traffic logs if your environment has web proxy visibility. Apply D3-SFA (System File Analysis) to detect unauthorized modification of application bundles.

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs), MITRE ATT&CK T1105 (Ingress Tool Transfer), MITRE ATT&CK T1071.001 (Application Layer Protocol: Web Protocols)

Compensating: Without EDR, enable macOS Unified Logging at debug level for process execution: `sudo log config --mode level:debug,persist:debug`. Hunt shell spawning from Flutter bundles using: `log stream --predicate 'processImagePath CONTAINS "FlutterShell" OR (processImagePath CONTAINS ".app" AND eventMessage CONTAINS "bash")' --style syslog`. For network visibility, run Wireshark or `tcpdump -i en0 -w /tmp/fluttershell_capture.pcap host not ` on suspect hosts. Use osquery to hunt WebView-to-shell patterns: `SELECT pid, name, path, parent, cmdline FROM processes WHERE parent IN (SELECT pid FROM processes WHERE path LIKE '%/Contents/MacOS/%') AND name IN ('bash','sh','zsh');`. Deploy the free Objective-See tool BlockBlock to alert on new LaunchAgent/LaunchDaemon writes, and ProcInfo to enumerate parent-child process trees in real time. Write a YARA rule targeting Flutter binary markers combined with embedded remote URL string patterns within the app bundle's Mach-O executable.

Evidence: Capture macOS Unified Log entries showing the Flutter WKWebView process tree: `log show --predicate 'subsystem == "com.apple.webkit" OR processImagePath CONTAINS ".app/Contents/MacOS"' --info --last 24h >

/tmp/webkit_process_log.txt`. Pull all LaunchAgent plists written by or referencing the suspect bundle: `find ~/Library/LaunchAgents /Library/LaunchAgents /Library/LaunchDaemons -newer /Applications/PodcastsLounge.app -ls`. Extract the process network socket table at time of execution using `lsof -i -n -P | grep -E 'PodcastsLounge|PDF-Brain|PDF-Ninja'` and preserve output. For the Google Ads delivery vector, export browser history and download records: `sqlite3 ~/Library/Application\ Support/Google/Chrome/Default/History 'SELECT url, title, last_visit_time FROM urls WHERE url LIKE "%googleads%" OR url LIKE "%doubleclick%";'`. Capture full DNS resolution logs from the macOS resolver: `sudo tcpdump -i en0 port 53 -w /tmp/dns_capture.pcap` during a controlled relaunch in an isolated network segment.

Step 3: Eradication — Remove all identified FlutterShell variants and their persistence mechanisms (LaunchAgents, LaunchDaemons, login items) from affected hosts. Reset all credentials accessible from compromised macOS endpoints, prioritizing browser-stored passwords, SSH keys, and API tokens (D3-CRO: Credential Rotation). Revoke and reissue any certificates or secrets stored on affected systems. Block identified IOC domains and IPs at perimeter (NIST SC-7). No vendor-supplied patch or notarization revocation for these specific app bundles has been confirmed as of this writing — monitor Apple's revocation list for updates.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SC-7 (Boundary Protection), NIST AC-2 (Account Management), NIST AC-17 (Remote Access), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: For persistence removal without EDR, run Objective-See's KnockKnock (free) to enumerate all LaunchAgents, LaunchDaemons, login items, cron jobs, and kernel extensions on the affected host, then cross-reference against the FlutterShell bundle identifiers. Manually enumerate: `launchctl list | grep -v apple` and compare against known-good baselines. Remove malicious entries with `launchctl bootout gui/$(id -u)/`. For credential rotation without a PAM: force-expire all SSH keys found in `~/.ssh/` on affected hosts, rotate all secrets stored in `~/Library/Keychains/` by re-authenticating all services, and notify users to change passwords for any service accessed from the compromised host via the browser. Use `security dump-keychain ~/Library/Keychains/login.keychain-db` (requires user auth) to inventory what credentials were accessible to the backdoor during its execution window.

Evidence: Before any removal, image the full LaunchAgent directory: `tar -czf /tmp/launchagents_evidence.tar.gz ~/Library/LaunchAgents /Library/LaunchAgents /Library/LaunchDaemons`. Dump all login items: `osascript -e 'tell application "System Events" to get the name of every login item'`. Enumerate all SSH keys and their last-accessed timestamps: `ls -la ~/.ssh/ && last -10` to establish the credential exposure window. Extract macOS Keychain entries accessible without re-authentication to document the blast radius of credential exposure. Pull the full browser credential store access log from the macOS Unified Log: `log show --predicate 'subsystem == "com.apple.Safari.SafeBrowsing" OR processImagePath CONTAINS "Chrome"' --last 72h`. Preserve the complete macOS Security audit log (`/var/audit/`) covering the infection window before any system modification.

Step 4: Recovery — Reimage affected macOS endpoints from a known-good baseline rather than attempting in-place cleanup, given the backdoor's remote payload model. Validate that no lateral movement occurred from compromised macOS endpoints to Windows or Linux systems via shared credentials or network files (NIST IR-4). Monitor previously affected user accounts for anomalous authentication events for a minimum of 30 days post-remediation (NIST AU-6, D3-LAM: Local Account Monitoring). Confirm EDR coverage is active and current on all macOS assets before returning to production.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST CP-10 (System Recovery and Reconstitution), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 5.1 (Establish and Maintain an Inventory of Accounts)

Compensating: For reimaging without enterprise tooling, use Apple Configurator 2 (free) to restore affected Macs to a known-good MDM-enrolled baseline. For lateral movement validation without a SIEM, query Active Directory or LDAP authentication logs for the affected user accounts in the 72 hours following estimated compromise: `Get-ADUser -Identity | Get-ADObject -Properties lastLogon` on Windows DCs. On Linux systems sharing NFS mounts or SSH access with the compromised Mac, review `/var/log/auth.log` for the affected username and any SSH key fingerprints that were present on the compromised host. Set up a free Wazuh agent on the reimaged endpoint to alert on any re-emergence of the FlutterShell bundle identifiers post-recovery. Monitor IdP logs (Okta, Azure AD free tier) for the affected user accounts for 30 days for impossible travel, new device enrollment, or MFA push fatigue patterns.

Evidence: Before reimaging, preserve a full forensic image using `dd if=/dev/diskX of=/path/to/evidence/endpoint_image.dd bs=4m` or use Target Disk Mode with a write blocker. Capture the complete process execution history from the macOS Unified Log archive for the entire infection window. Document all network connections established by the Flutter app bundles by pulling the socket statistics: `netstat -anv | grep ESTABLISHED` and `lsof -i 4 -n -P`. Preserve the macOS TCC (Transparency, Consent, and Control) database which records what permissions — camera, microphone, disk access, keychain — FlutterShell requested or was granted: `cp ~/Library/Application\ Support/com.apple.TCC/TCC.db /tmp/tcc_evidence.db`. Extract the macOS Security framework log to document any certificate or code-signing evaluations performed during the infection period.

Step 5: Post-Incident — This campaign exploited gaps in application vetting (no enforcement of application allowlisting), ad-network trust (no controls restricting employee software downloads from search ads), and static-analysis-dependent detection tooling (no behavioral or network-based detection for post-launch payload retrieval). Address these gaps: implement macOS application allowlisting enforced via MDM (NIST CM-7, CIS 2.3), deploy DNS-layer filtering to catch C2 beacon attempts (NIST SC-20), establish behavioral detection rules for WebView-to-shell-bridge patterns in your EDR (NIST SI-4), and add user awareness training specific to malvertising as a delivery vector (NIST AT-2). Review your Google Ads and browser extension policies for macOS users.

NIST Phase: Post Incident

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

Controls: NIST CM-7 (Least Functionality), NIST SC-20 (Secure Name/Address Resolution Service — Authoritative Source), NIST SI-4 (System Monitoring), NIST AT-2 (Literacy Training and Awareness), NIST IR-4 (Incident Handling), CIS 2.3 (Address Unauthorized Software), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For MDM-enforced allowlisting without enterprise spend, configure a free Jamf Now or Mosyle Personal account to push a macOS configuration profile restricting Gatekeeper to App Store only, or use the free tier of Santa (Google's open-source binary allowlisting tool for macOS) to enforce a lockdown mode that blocks any unsigned or non-allowlisted binary. For DNS-layer filtering, deploy Pi-hole (free, self-hosted) or enable Cloudflare Gateway (free tier) as the upstream resolver for macOS DHCP clients, subscribing to threat intel blocklists that include newly registered domains and known malvertising infrastructure. Write a Sigma rule targeting the WebView-to-shell-bridge pattern (WKWebView parent process spawning bash/sh/zsh child) and deploy it against your log source. Conduct a tabletop exercise specifically simulating a Google Ads malvertising delivery of a notarized macOS app — this gap was the primary enabler for CL-CRI-1089 and is likely to be repeated.

Evidence: Document the full attack chain in a lessons-learned report: originating Google Ad URL (from browser quarantine xattr), download timestamp, notarization ticket details (from `stapler validate`), first-execution timestamp (from macOS Unified Log), C2 beacon timing and destination (from pcap), and persistence mechanism installed. Preserve the complete threat intelligence package including the VirusTotal zero-detection snapshot timestamp, Apple notarization ticket, and any WHOIS records for C2 domains as baseline evidence for future detection tuning. Cross-reference the CL-CRI-1089 TTPs against your current MITRE ATT&CK Navigator layer to identify coverage gaps — specifically T1105 (Ingress Tool Transfer via remote payload), T1071.001 (C2 over HTTPS), T1204.002 (User Execution: Malicious File via malvertising), and T1553.001 (Subvert Trust Controls: Gatekeeper Bypass via notarization abuse).

Detection Guidance

Primary detection surface is behavioral and network-based; static analysis will not catch this campaign. Focus on the following: (1) EDR process tree analysis: look for Flutter/Dart runtime processes spawning native shell interpreters (bash, sh, zsh) or making outbound network calls to non-CDN endpoints within the first 60 seconds of application launch. (2) DNS and proxy logs: identify macOS hosts resolving newly registered domains or domains with no prior organizational history, particularly from user-space application processes. Flag any application in /Applications or ~/Downloads initiating DNS resolution to single-use or dynamic DNS infrastructure. (3) File system monitoring: watch for new LaunchAgent or LaunchDaemon plist files created by non-system processes, especially those referencing paths inside application bundles. (4) Network exfiltration indicators: look for sustained or scheduled outbound HTTPS sessions from macOS endpoints to low-reputation domains, particularly those carrying document-type file uploads (multipart/form-data, base64-encoded payloads). (5) Application inventory audit: cross-reference installed macOS applications against your approved software inventory (CIS 2.1); flag PodcastsLounge, PDF-Brain, PDF-Ninja, and any other Flutter-based applications not explicitly approved. (6) Google Ads exposure: if your organization has web proxy logs, identify employees who clicked Google Ads directing to software download pages and follow up with targeted endpoint investigation on those hosts. Specific file hashes and IOC domains are available in Unit 42's technical report. Confidence in detections will be high-to-medium when combining behavioral indicators with IOC-based detection; implement both immediately.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`PodcastsLounge (application bundle)`	Malicious macOS application masquerading as a podcasting app; known FlutterShell delivery vehicle per Unit 42	HIGH
DOMAIN	`PDF-Brain (application bundle)`	Malicious macOS application masquerading as a PDF utility; known FlutterShell delivery vehicle per Unit 42	HIGH
DOMAIN	`PDF-Ninja (application bundle)`	Malicious macOS application masquerading as a PDF utility; known FlutterShell delivery vehicle per Unit 42	HIGH

Framework Mappings

MITRE-ATTACK

- **T1566** — Phishing
- **T1553.001** — Gatekeeper Bypass
- **T1567** — Exfiltration Over Web Service
- **T1583.008** — Malvertising
- **T1056** — Input Capture

- **T1583.006** — Web Services
- **T1083** — File and Directory Discovery
- **T1027.002** — Software Packing
- **T1041** — Exfiltration Over C2 Channel
- **T1588.004** — Digital Certificates
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1059.007** — JavaScript
- **T1105** — Ingress Tool Transfer
- **T1071.001** — Web Protocols
- **T1059.004** — Unix Shell
- **T1027.009** — Embedded Payloads
- **T1082** — System Information Discovery
- **T1027** — Obfuscated Files or Information
- **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

OWASP-TOP10-2021

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

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **8.2** — Collect Audit Logs

NIST-CSF-2

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1566	Phishing	Initial-Access
T1553.001	Gatekeeper Bypass	Defense-Evasion
T1567	Exfiltration Over Web Service	Exfiltration
T1583.008	Malvertising	Resource-Development
T1056	Input Capture	Collection
T1583.006	Web Services	Resource-Development
T1083	File and Directory Discovery	Discovery
T1027.002	Software Packing	Defense-Evasion
T1041	Exfiltration Over C2 Channel	Exfiltration
T1588.004	Digital Certificates	Resource-Development
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1059.007	JavaScript	Execution
T1105	Ingress Tool Transfer	Command-And-Control
T1071.001	Web Protocols	Command-And-Control
T1059.004	Unix Shell	Execution
T1027.009	Embedded Payloads	Defense-Evasion
T1082	System Information Discovery	Discovery
T1027	Obfuscated Files or Information	Defense-Evasion
T1204.002	Malicious File	Execution

Sources

Source	URL	Tier
Unit 42	https://unit42.paloaltonetworks.com/flutterbridge-new-fluttershell-...	T3
Malicious Google Ads: Mac Password Stealer Blocked	https://underdefense.com/blog/malicious-google-ads-blocked/	T3
Google Ads spread Mac malware disguised as popular browser	https://www.foxnews.com/tech/google-ads-spread-mac-malware-disguise...	T3

Source	URL	Tier
'Harmful' Google ads masquerading as how-tos are tricking Mac users	https://www.macworld.com/article/3045954/harmful-google-ads-masquer...	T3
Mac Malware Infiltrates Google Ads | Bob Carver posted on the topic	https://www.linkedin.com/posts/bobcarver_cybersecurity-macos-google...	T3

Generated 2026-06-02 14:01 UTC by TJS Security Command Center