INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-05 06:56 UTC

Hackers Use Malicious Ads to Deliver FlutterShell Backdoor on macOS Systems

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0414
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	macOS systems (specific versions unconfirmed at this time)
Published	2026-06-04
Discovery Source	Gemini

Executive Summary

Threat actors are distributing a macOS backdoor called FlutterShell through malicious online advertisements, tricking users into downloading malware onto their Apple computers. The campaign targets macOS systems broadly; specific versions confirmed affected have not been established by authoritative sources at this time. Organizations with macOS fleets face risk of unauthorized remote access and potential data exposure if users encounter and execute the malicious payload.

Technical Analysis

FlutterShell is a macOS backdoor delivered via malvertising (T1566.002, T1204.002). The malware appears to leverage the Flutter UI framework as a code obfuscation layer to evade static analysis and endpoint detection; Flutter's compiled Dart code is suspected to serve this purpose, consistent with a growing trend of cross-platform malware using compiled code to obscure logic, though detailed reverse engineering confirmation is not available in current sources. Mapped weaknesses include CWE-494 (Download of Code Without Integrity Check) and CWE-610 (Externally Controlled Reference to a Resource in Another Sphere). MITRE ATT&CK techniques: T1036 (Masquerading), T1059 (Command and Scripting Interpreter), T1071 (Application Layer Protocol), T1566.002 (Phishing via malicious ad), T1204.002 (Malicious File). No CVE is assigned. No CISA KEV entry exists. Command-and-control infrastructure, persistence mechanisms, and full capability set are unconfirmed from available source data. Source quality is secondary-tier; available sources are primarily vendor blogs and community reports. No CISA alert or authoritative government advisory has been identified for this campaign at this time.

Action Checklist

1. Step 1: Containment. Identify macOS endpoints whose users may have encountered malicious online advertisements directing them to download unknown applications. Isolate any systems exhibiting anomalous outbound network connections pending investigation. Block execution of unsigned or unnotarized applications via macOS Gatekeeper policy enforcement (CIS Controls v8 2.3, Address Unauthorized Software).
2. Step 2: Detection. Search endpoint logs for execution of Flutter-compiled binaries (look for Dart runtime artifacts, unusual dylib loads, or processes spawned from browser download directories). Review macOS Unified Log and EDR telemetry for T1059-pattern script interpreter invocations and T1071-pattern C2 beaconing over HTTP/HTTPS. IOC patterns are not confirmed from available data; treat any unrecognized macOS binary installed via browser download in the past 30 days as a candidate for review. Reference NIST SP 800-53 AU-6 (Audit Record Review, Analysis, and Reporting) for log review cadence. No confirmed IOC values are available from current source data.
3. Step 3: Eradication. Remove any identified FlutterShell binaries and associated files. Enforce CIS Controls v8 2.3 (Address Unauthorized Software) by removing unauthorized applications. Restrict macOS systems to App Store or explicitly notarized software only. No vendor-issued patch or removal tool has been confirmed for this campaign; follow your endpoint detection and response vendor's guidance for macOS malware removal.
4. Step 4: Recovery. After removal, verify Gatekeeper and XProtect definitions are current on all macOS endpoints. Confirm no persistence mechanisms remain (review LaunchAgents, LaunchDaemons, and login items per NIST D3-SICA, System Init Config Analysis). Monitor outbound connections from previously affected systems for 14 days post-remediation using NIST SP 800-53 AU-6 (Audit Record Review, Analysis, and Reporting).
5. Step 5: Post-Incident. Evaluate user security awareness training coverage for malvertising as a distinct initial access vector. Review advertising content filtering controls at the DNS and proxy layer. Apply CIS Controls v8 4.4 (Implement and Manage a Firewall on Servers) and CIS Controls v8 4.5 (Implement and Manage a Firewall on End-User Devices) to restrict outbound connections from macOS endpoints to approved destinations. Document the control gap and update macOS endpoint policy to enforce application allowlisting.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate to senior IR leadership and legal/privacy counsel immediately if FlutterShell is confirmed active on endpoints with access to PII, PHI, or regulated data, or if C2 communication is confirmed as active, indicating potential data exfiltration has occurred and breach notification obligations under applicable regulations (e.g., HIPAA, GDPR, CCPA) may be triggered.

Recovery Notes	After FlutterShell binary removal and persistence mechanism eradication, verify clean state by re-imaging high-risk endpoints if forensic confirmation of full scope is not achievable — given the backdoor's remote access capability, partial remediation carries significant residual risk. Monitor all previously affected macOS endpoints for outbound beaconing at irregular HTTPS intervals for a minimum of 14 days, as dormant C2 check-in schedules may not surface immediately. Confirm Gatekeeper, XProtect, and macOS security updates are current across the entire macOS fleet before returning any isolated system to the production network.
Forensic Artifacts	macOS QuarantineEventsV2 SQLite database (~/.Library/Preferences/com.apple.LaunchServices.QuarantineEventsV2) — records originating ad URL, download timestamp, and application that initiated the FlutterShell download, directly reconstructing the malvertising delivery chain | Browser download history databases (Chrome: ~/Library/Application Support/Google/Chrome/Default/History; Safari: ~/Library/Safari/Downloads.plist) — identify the specific malicious advertisement URL and the filename under which FlutterShell was presented to the user | macOS Unified Log archive (collected via `log collect --last 72h`) — contains Gatekeeper assessment events, dylib load events for Dart runtime libraries, and process execution records for the FlutterShell binary and any child processes it spawned | LaunchAgent and LaunchDaemon plist files in ~/Library/LaunchAgents/, /Library/LaunchAgents/, and /Library/LaunchDaemons/ — FlutterShell persistence mechanism artifacts identifying the scheduled execution path and command-line arguments used to maintain backdoor access across reboots | Network packet capture (tcpdump/Wireshark) of outbound connections from the affected endpoint — evidence of T1071.001 C2 beaconing behavior over HTTP/HTTPS, including destination IP, beacon interval, and User-Agent strings associated with the Flutter HTTP client library

Per-Action IR Details

Step 1: Containment — Identify macOS endpoints whose users may have clicked on advertisements directing them to download unknown applications. Isolate any systems exhibiting anomalous outbound network connections pending investigation. Block execution of unsigned or unnotarized applications via macOS Gatekeeper policy enforcement (CIS 2.3 — Address Unauthorized Software).

NIST Phase: Containment

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

Controls: CIS 2.3 (IG1/IG2/IG3) — Address Unauthorized Software, AC-4 — Information Flow Enforcement, AC-6 — Least Privilege

Compensating: Run `spctl --status` on each macOS endpoint to confirm Gatekeeper is enabled and set to 'assessments enabled'. Use `osquery` with query `SELECT * FROM processes WHERE path LIKE '/Users/%/Downloads/%' OR path LIKE '/private/tmp/%'` to surface processes launched from browser download directories — a hallmark of FlutterShell's malvertising delivery path. Block outbound C2 candidates at the perimeter firewall by denying all non-approved egress from macOS subnets pending investigation. This can be enforced by a 2-person team using macOS `pfctl` with a deny-by-default outbound ruleset on isolated VLANs.

Evidence: BEFORE isolating: capture a full process list via `ps auxww` and a network connection snapshot via `netstat -anp tcp` or `lsof -i` to record active outbound connections from the suspect endpoint. Collect browser download history from `~/Library/Application Support/Google/Chrome/Default/History` (Chrome) or `~/Library/Safari/Downloads.plist` (Safari) to reconstruct which malicious advertisement URL triggered the FlutterShell download. Image the macOS Unified Log via `log collect --last 72h --output /path/to/archive.logarchive` before any remediation actions alter log state.

Step 2: Detection — Search endpoint logs for execution of Flutter-compiled binaries (look for Dart runtime artifacts, unusual dylib loads, or processes spawned from browser download directories). Review macOS Unified Log and EDR telemetry for T1059-pattern script interpreter invocations and T1071-pattern C2 beaconing over HTTP/HTTPS. IOC patterns are not confirmed from available data — treat any unrecognized

macOS binary installed via browser download in the past 30 days as a candidate for review. Reference AU-6 (Audit Record Review, Analysis, and Reporting) for log review cadence. No confirmed IOC values are available from current source data.

NIST Phase: Detection Analysis

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

Controls: AU-6 — Audit Record Review, Analysis, And Reporting, AU-12 — Audit Record Generation, AU-2 — Event Logging

Compensating: Without EDR, use the macOS Unified Log subsystem: run `log show --predicate 'subsystem == "com.apple.security.gk"' --last 30d` to surface Gatekeeper override events that would indicate a user bypassed quarantine warnings to run the FlutterShell binary. Query for dylib injection artifacts with `osquery`: `SELECT * FROM shared_libraries WHERE path NOT LIKE '/usr/lib/%' AND path NOT LIKE '/System/%'`. Write a YARA rule targeting Flutter/Dart runtime strings (e.g., `dart:io`, `dart:isolate`, `FlutterEngine`) and scan all binaries in `/Users/*/Downloads/` and `/Applications/` using `yara -r flutter_shell.yar /`. For network detection, capture egress with `tcpdump -i en0 -w /tmp/capture.pcap 'not host trusted_ip'` and analyze in Wireshark filtering for periodic beaconing intervals characteristic of T1071.001 C2 over HTTPS.

Evidence: BEFORE analysis actions alter artifacts: export the macOS Unified Log archive as noted in Step 1. Query `sqlite3 ~/Library/Application\ Support/com.apple.sharedfilelist/com.apple.LSSharedFileList.RecentApplications.sfl2` to identify recently launched applications including FlutterShell candidates. Check `~/Library/Preferences/com.apple.LaunchServices.QuarantineEventsV2` (SQLite) — this database records the originating URL and download timestamp for every quarantined file, directly linking the malicious ad URL to the downloaded FlutterShell payload. Collect `/private/var/log/system.log` and `/private/var/log/install.log` for the relevant 30-day window before any log rotation occurs.

Step 3: Eradication — Remove any identified FlutterShell binaries and associated files. Enforce CIS 2.3 (Address Unauthorized Software) by removing unauthorized applications. Restrict macOS systems to App Store or explicitly notarized software only. No vendor-issued patch or removal tool has been confirmed for this campaign; follow your endpoint detection and response vendor's guidance for macOS malware removal.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: CIS 2.3 (IG1/IG2/IG3) — Address Unauthorized Software, CIS 2.2 (IG1/IG2/IG3) — Ensure Authorized Software is Currently Supported, SI-3 — Malicious Code Protection

Compensating: No vendor removal tool is confirmed for FlutterShell. Manually enumerate and remove persistence artifacts before deleting the main binary — in this order: (1) `launchctl list | grep -v com.apple` to surface non-Apple LaunchAgents/Daemons installed by FlutterShell; (2) remove identified plist files from `~/Library/LaunchAgents/`, `/Library/LaunchAgents/`, and `/Library/LaunchDaemons/`; (3) delete the FlutterShell application bundle and any associated Dart runtime files from `/Users/*/Downloads/` or `/Applications/`. After removal, run `spctl --assess --verbose /path/to/binary` on any remaining unknown binaries to verify notarization status. Enforce Gatekeeper via MDM profile (or `sudo spctl --master-enable`) to prevent re-execution of unnotarized binaries.

Evidence: BEFORE eradication: take a forensic copy of the FlutterShell binary itself (SHA-256 hash it with `shasum -a 256 /path/to/binary`) for future IOC sharing and YARA rule development. Preserve all identified LaunchAgent/LaunchDaemon plist files as evidence of persistence mechanism. Document the full file path, creation timestamp (`GetFileInfo -d /path/to/binary`), and extended attributes (`xattr -l /path/to/binary`) including the `com.apple.quarantine` flag — its presence or absence indicates whether the user was warned before executing the FlutterShell payload.

Step 4: Recovery — After removal, verify Gatekeeper and XProtect definitions are current on all macOS endpoints. Confirm no persistence mechanisms remain (review LaunchAgents, LaunchDaemons, and login items per D3-SICA — System Init Config Analysis). Monitor outbound connections from previously affected systems for 14 days post-remediation using AU-6 (Audit Record Review, Analysis, and Reporting).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: AU-6 — Audit Record Review, Analysis, And Reporting, AU-11 — Audit Record Retention, CIS 4.4 (IG1/IG2/IG3) — Implement and Manage a Firewall on Servers, CIS 4.5 (IG1/IG2/IG3) — Implement and Manage a Firewall on End-User Devices

Compensating: Verify XProtect and Gatekeeper currency without MDM using `softwareupdate --list` and confirming `XProtect.bundle` version in `/Library/Apple/System/Library/CoreServices/XProtect.bundle/Contents/Resources/`. For persistence verification, run: `launchctl list`, `ls -la ~/Library/LaunchAgents/ /Library/LaunchAgents/ /Library/LaunchDaemons/`, and `osascript -e 'tell application "System Events" to get the name of every login item'` — any entry referencing a non-Apple, non-corporate binary is a FlutterShell re-infection candidate. For 14-day outbound monitoring without SIEM, schedule a daily `tcpdump` cron job capturing egress from the recovered endpoint and review with Wireshark for periodic beaconing patterns consistent with T1071 C2 behavior.

Evidence: BEFORE returning system to production: document baseline process list and active network connections post-remediation as a clean reference state. Confirm `com.apple.quarantine` extended attribute behavior is restored (i.e., newly downloaded files are being quarantined). Verify the QuarantineEventsV2 database no longer shows unresolved entries for the FlutterShell download URL — its continued presence could indicate the browser profile was not cleaned and re-download is possible.

Step 5: Post-Incident — Evaluate user security awareness training coverage for malvertising as an initial access vector. Review advertising content filtering controls at the DNS and proxy layer. Apply CIS 4.4 (Implement and Manage a Firewall on Servers) and CIS 4.5 (Implement and Manage a Firewall on End-User Devices) to restrict outbound connections from macOS endpoints to approved destinations. Document the control gap and update macOS endpoint policy to enforce application allowlisting.

NIST Phase: Post Incident

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

Controls: CIS 4.4 (IG1/IG2/IG3) — Implement and Manage a Firewall on Servers, CIS 4.5 (IG1/IG2/IG3) — Implement and Manage a Firewall on End-User Devices, CIS 2.1 (IG1/IG2/IG3) — Establish and Maintain a Software Inventory, AU-6 — Audit Record Review, Analysis, And Reporting

Compensating: For DNS-layer malvertising blocking without commercial tooling, deploy Pi-hole or configure a DNS RPZ (Response Policy Zone) with ad-network blocklists (e.g., Steven Black's unified hosts list) to prevent browsers from resolving malicious ad-delivery domains that served FlutterShell. For application allowlisting on macOS without a commercial solution, configure a macOS MDM profile (or manually via `santactl` from Google's open-source Santa tool) to block execution of any binary not in an approved hash list — directly addressing the FlutterShell delivery mechanism of executing a freshly downloaded, unrecognized binary. Document the confirmed control gap (absence of DNS ad filtering and application allowlisting) in a post-incident report referencing this FlutterShell campaign as the triggering event.

Evidence: Collect and preserve: the malicious advertisement URL(s) reconstructed from QuarantineEventsV2 and browser history; the SHA-256 hash(es) of the FlutterShell binary collected during eradication; all LaunchAgent plist files used for persistence; and the 14-day post-remediation network capture from Step 4. These artifacts collectively constitute the incident record and serve as the basis for internal IOC sharing and any future YARA or Sigma rule development targeting Flutter-compiled macOS backdoors delivered via malvertising.

Detection Guidance

No confirmed IOCs (hashes, domains, IPs) are available from current source data; do not fabricate or assume. Detection should be behavioral. Query EDR telemetry for: (1) macOS processes spawned from browser download directories executing within 60 seconds of download, (2) Dart runtime or Flutter engine dylib loads from non-App Store application bundles, (3) outbound connections to non-categorized or newly registered domains initiated by recently installed applications. Review macOS Unified Log for LaunchAgent or LaunchDaemon registrations outside of known software baselines (NIST D3-SICA, System Init Config Analysis).

Enable NIST SP 800-53 AU-2 (Event Logging) across macOS fleet if not already active. Monitor local account activity per NIST D3-LAM (Local Account Monitoring) for privilege escalation attempts post-infection. Alert on CWE-494 patterns: unsigned binary execution following a user-initiated download event.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`not confirmed`	No C2 domains have been confirmed from available source data. Do not fabricate.	LOW
HASH	`not confirmed`	No file hashes for FlutterShell binaries have been confirmed from available source data.	LOW

Framework Mappings

MITRE-ATTACK

- **T1036** — Masquerading
- **T1059** — Command and Scripting Interpreter
- **T1071** — Application Layer Protocol
- **T1566.002** — Spearphishing Link
- **T1204.002** — Malicious File

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **AT-2** — Literacy Training and Awareness
- **SI-8** — Spam Protection
- **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
T1036	Masquerading	Defense-Evasion
T1059	Command and Scripting Interpreter	Execution
T1071	Application Layer Protocol	Command-And-Control
T1566.002	Spearphishing Link	Initial-Access
T1204.002	Malicious File	Execution

Sources

Source	URL	Tier
Security Vulnerability for Apple Devices - Yale Cybersecurity	https://cybersecurity.yale.edu/news/security-vulnerability-apple-de...	T1
macOS Spotlight Vulnerability Discovered by Microsoft - Reddit	https://www.reddit.com/r/apple/comments/1mbl9g8/macos_spotlight_vul...	T3
Apple Alerted to macOS Security Vulnerability Uncovered With AI Tool	https://www.facebook.com/MacRumors/posts/apple-alerted-to-macos-sec...	T3
About the security content of macOS Sequoia 15.5 - Apple Support	https://support.apple.com/en-us/122716	T3
Microsoft Uncovers macOS Vulnerability CVE-2024-44243 Allowing ...	https://thehackernews.com/2025/01/microsoft-uncovers-macos-vulnerab...	T3

Generated 2026-06-05 06:56 UTC by TJS Security Command Center