INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-28 07:04 UTC

Gaslight: Rust-Based macOS Malware Exploits Prompt Injection to Evade AI-Assisted Analysis

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0593
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	macOS enterprise fleets; AI/ML-assisted malware analysis tools; Security Operations Centers leveraging AI automation
Published	2026-06-26
Discovery Source	Gemini

Executive Summary

A Rust-based macOS malware strain named 'Gaslight' targets AI-assisted malware analysis pipelines by embedding adversarial prompt injection instructions that cause large language model tools to misclassify, abort, or suppress analysis of the sample. Organizations operating macOS enterprise fleets and AI-augmented SOC workflows are at elevated risk of delayed or failed detection if AI automation holds significant triage authority. The business risk is compounded by the malware's Rust implementation, which reduces static analysis artifacts and impedes conventional detection methods.

Technical Analysis

Gaslight is a Rust-based macOS malware strain employing prompt injection (CWE-20, CWE-116) as an AI evasion technique, embedding adversarial natural language instructions within code or metadata to manipulate LLM-based analysis tools into aborting, misclassifying, or deprioritizing the sample. This represents a protection mechanism bypass (CWE-693) targeting the analysis toolchain rather than traditional EDR or AV detection layers. MITRE ATT&CK mappings include T1027 (Obfuscated Files or Information), T1059 (Command and Scripting Interpreter), T1562.001 (Impair Defenses: Disable or Modify Tools), and T1204 (User Execution). Rust's memory safety model reduces exploitable artifacts and complicates both static and dynamic analysis. No CVE, NVD entry, or CISA KEV listing exists for this campaign at time of publication. Source data is Tier 3 (secondary); independent verification against CISA, NVD, and vendor advisories is recommended before operational action. CVSS base: 7.5 (qualitative high); EPSS data unavailable.

Action Checklist

1. Step 1, Containment: Audit macOS endpoints in your enterprise fleet for unrecognized Rust-compiled binaries, particularly those lacking standard code-signing certificates. Isolate any suspected samples from AI-assisted analysis pipelines pending manual review. Reference CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory) to confirm asset visibility before scoping containment.
2. Step 2, Detection: Search macOS endpoint logs and EDR telemetry for unsigned or anomalously signed Rust binaries executing on managed devices. Review AI-assisted triage tool logs for analysis abort events, refusals, or misclassification patterns on macOS samples submitted in the past 90 days. These may indicate prior prompt injection attempts. Apply NIST AU-6 (Audit Record Review, Analysis, and Reporting) to guide log review scope and frequency. Behavioral indicators include LLM analysis tools returning null or low-confidence verdicts on macOS samples, unexpected analysis pipeline exits, and samples queued but never triaged.
3. Step 3, Eradication: Remove identified suspicious binaries from affected endpoints following your established malware removal procedures. For AI-assisted analysis pipelines, implement input sanitization on sample metadata and embedded strings before submission to LLM analysis layers, removing or neutralizing prompt injection payloads before they reach the model context. Apply NIST CM-7 (Least Functionality) to restrict which pipeline components have authority to suppress or deprioritize alerts without human confirmation. No vendor patch is available; mitigation is architectural.
4. Step 4, Recovery: Validate that AI-assisted analysis pipelines are producing consistent triage verdicts on known-clean and known-malicious macOS samples following pipeline hardening. Re-examine any macOS samples that were aborted, misclassified, or suppressed during the exposure window. Apply NIST IR-5 (Incident Monitoring) to track resubmitted samples through to resolution. Confirm Gatekeeper enforcement and notarization requirements are active across all macOS fleet devices per vendor guidance.
5. Step 5, Post-Incident: Review the degree of autonomous decision-making authority granted to AI triage tools in your SOC workflow. Where AI automation has authority to suppress or deprioritize alerts without human review, introduce mandatory human-in-the-loop checkpoints for low-confidence verdicts. Apply NIST IR-8 (Incident Response Plan) to formally document this control gap and update playbooks. Reference CIS 7.1 (Establish and Maintain a Vulnerability Management Process) to incorporate AI toolchain integrity as a standing review item. Consider threat modeling exercises against your AI analysis pipeline using MITRE ATT&CK T1562.001 as a hypothesis anchor.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to senior IR leadership and legal/compliance if LLM verdict logs confirm that Gaslight samples were actively suppressed or misclassified during the exposure window AND any of those samples originated from or touched systems processing PII, PHI, or regulated data — triggering potential breach notification obligations — or if the AI pipeline's autonomous suppression authority resulted in no human analyst reviewing a confirmed-malicious macOS binary within your defined SLA window.

Recovery Notes	Following pipeline hardening, maintain a 30-day heightened monitoring period during which all macOS sample verdicts from the AI triage layer require secondary analyst confirmation before alert closure, regardless of confidence score. Resubmit the full backlog of suppressed and aborted macOS samples from the 90-day exposure window and track each to a human-confirmed verdict, logging results in IR-5 incident tracking. At the 30-day mark, conduct a structured review of new verdict distributions to confirm the pipeline is no longer producing anomalous suppression rates on macOS samples before returning to standard autonomous triage authority.
Forensic Artifacts	macOS Unified Log archive (logarchive format) covering process execution, code-signing validation, and Gatekeeper assessment events for the 90-day exposure window — primary source for identifying when and where Gaslight binaries executed on fleet endpoints | AI/LLM triage tool structured verdict logs (JSON/CSV) showing analysis_status of 'aborted', 'skipped', 'suppressed', or confidence scores below threshold for macOS/Mach-O sample submissions — direct evidence of successful prompt injection bypass | Raw sample metadata and embedded string fields submitted to the LLM analysis layer, preserved before pipeline hardening — these contain the adversarial prompt injection payload text in plaintext and establish the attacker's instruction set | Memory dump (osxpmem AFF4 format) from any host where a Gaslight binary was observed executing, capturing Rust runtime heap contents including any injected instruction strings loaded into process memory at time of execution | codesign and spctl assessment output for each identified Rust binary (`codesign -dv --verbose=4` and `spctl --assess -vv`), documenting signing authority, team ID, and Gatekeeper disposition — establishes the code-signing evasion method used to bypass macOS trust controls

Per-Action IR Details

Step 1: Containment — Audit macOS endpoints in your enterprise fleet for unrecognized Rust-compiled binaries, particularly those lacking standard code-signing certificates. Isolate any suspected samples from AI-assisted analysis pipelines pending manual review. Reference CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory) to confirm asset visibility before scoping containment.

NIST Phase: Containment

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

Controls: CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), NIST CM-8 (System Component Inventory), NIST IR-4 (Incident Handling)

Compensating: Run `find / -type f -perm +111 2>/dev/null | xargs file | grep 'Mach-O'` on each managed macOS host to enumerate executables, then pipe through `codesign -dv --verbose=4 ` to identify unsigned or ad-hoc-signed Rust binaries. Use osquery with `SELECT * FROM signature WHERE path LIKE '/Applications/%' AND signed = '0';` and extend to user-writable paths such as ~/Library and /tmp. A 2-person team can script this across fleet hosts via SSH in a single sweep.

Evidence: Before isolating any suspect host or removing binaries from the analysis pipeline, capture: (1) full memory dump using osxpmem or `sudo osxpmem -o /tmp/mem.aff4` to preserve in-memory Rust runtime state and any injected prompt strings loaded into process heap; (2) output of `ps aux`, `lsof -nP`, and `netstat -an` to record active processes and open connections from the Gaslight binary at time of discovery; (3) Gatekeeper assessment log at `/var/log/DiagnosticMessages` and `spctl --assess -vv ` output; (4) quarantine attribute metadata via `xattr -l ` to determine whether the binary bypassed Gatekeeper quarantine checks. These are destroyed upon host isolation or binary removal.

Step 2: Detection — Search macOS endpoint logs and EDR telemetry for unsigned or anomalously signed Rust binaries executing on managed devices. Review AI-assisted triage tool logs for analysis abort events, refusals, or misclassification patterns on macOS samples submitted in the past 90 days — these may indicate

prior prompt injection attempts. Apply NIST AU-6 (Audit Record Review, Analysis, and Reporting) to guide log review scope and frequency. Behavioral indicators include LLM analysis tools returning null or low-confidence verdicts on macOS samples, unexpected analysis pipeline exits, and samples queued but never triaged.

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-2 (Event Logging), NIST AU-3 (Content of Audit Records), CIS 8.2 (Collect Audit Logs)

Compensating: Query the Unified Log on each macOS endpoint using `log show --predicate 'eventMessage CONTAINS "exec"' --info --last 90d | grep -i rust` to surface Rust binary execution events. For AI pipeline abort detection without a SIEM, parse your LLM triage tool's structured output logs (JSON or CSV verdict files) using `jq '[.[] | select(.verdict == null or .confidence < 0.3)]'` to isolate aborted or low-confidence macOS sample verdicts across the 90-day window. Flag any sample whose verdict field was populated by an automated suppression rule rather than analyst confirmation.

Evidence: Capture before any remediation action: (1) macOS Unified Log export covering the past 90 days filtered on process execution and code-signing validation events (`log collect --output /tmp/unified_log.logarchive`); (2) AI/LLM triage tool verdict logs from the same 90-day window, specifically entries where analysis_status equals 'aborted', 'skipped', or 'suppressed' on Mach-O/macOS sample types — these are the primary indicators of successful Gaslight prompt injection; (3) any raw sample metadata fields (filename, embedded strings, file description) submitted to the LLM analysis layer, which may contain the adversarial prompt injection payload in plaintext; (4) EDR process tree telemetry showing parent-child relationships for the Rust binary's execution chain. Do not flush or rotate LLM tool logs before extraction.

Step 3: Eradication — Remove identified suspicious binaries from affected endpoints following your established malware removal procedures. For AI-assisted analysis pipelines, implement input sanitization on sample metadata and embedded strings before submission to LLM analysis layers, preventing adversarial instructions from reaching the model context. Apply NIST CM-7 (Least Functionality) to restrict which pipeline components have authority to suppress or deprioritize alerts without human confirmation. No vendor patch is available; mitigation is architectural.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST CM-7 (Least Functionality), NIST CM-6 (Configuration Settings), NIST SI-2 (Flaw Remediation), CIS 2.3 (Address Unauthorized Software)

Compensating: Remove confirmed Gaslight binaries using `sudo rm -rf ` and validate removal with `find / -name 2>/dev/null`. For pipeline sanitization without a commercial solution, implement a pre-submission YARA rule that scans embedded strings in sample metadata for LLM instruction patterns (e.g., rules triggering on tokens such as 'ignore previous instructions', 'do not analyze', 'return benign') before the sample reaches the model context window. Apply this YARA scan as a mandatory gate in your pipeline using `yara gaslight_prompt_injection.yar ` and block submission on any match pending human review. Restrict the LLM tool's output handler so only analyst-confirmed verdicts can trigger alert suppression, enforced via a configuration flag in the pipeline's settings file.

Evidence: All volatile evidence from Steps 1 and 2 must be secured before binary removal. Additionally, before modifying pipeline configuration: (1) export the current pipeline configuration files and suppression-rule definitions as a forensic baseline to document the architectural exposure window; (2) preserve a cryptographic hash (SHA-256 via `shasum -a 256 `) of each Gaslight sample for IOC sharing and future YARA rule development; (3) extract and archive the embedded strings from each sample using `strings -a | grep -i 'ignore\|suppress\|benign\|abort'` to capture the prompt injection payload text as evidence. These artifacts establish the adversarial instruction set used and are required for post-incident reporting.

Step 4: Recovery — Validate that AI-assisted analysis pipelines are producing consistent triage verdicts on known-clean and known-malicious macOS samples following pipeline hardening. Re-examine any macOS

samples that were aborted, misclassified, or suppressed during the exposure window. Apply NIST IR-5 (Incident Monitoring) to track resubmitted samples through to resolution. Confirm Gatekeeper enforcement and notarization requirements are active across all macOS fleet devices per vendor guidance.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-5 (Incident Monitoring), NIST CM-6 (Configuration Settings), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Resubmit previously aborted or suppressed macOS samples through the hardened pipeline and compare new verdicts against prior null/low-confidence outputs — a verdict flip from suppressed to malicious confirms the prompt injection bypass was active. Validate Gatekeeper enforcement on all fleet hosts using `sudo spctl --status` (expected output: `assessments enabled`) and `defaults read /Library/Preferences/com.apple.security GKAutoRearm` to confirm auto-rearm is active. For the 30-day monitoring window, schedule a daily cron job to run `log show --predicate 'subsystem == "com.apple.ManagedClient"' --last 24h | grep -i 'unsigned\|quarantine\|rejected'` and email output to the SOC distribution list.

Evidence: During recovery validation, collect and retain: (1) side-by-side comparison logs of pre-hardening versus post-hardening LLM verdict outputs for all resubmitted macOS samples, timestamped, to document the scope of analysis failure caused by Gaslight prompt injection; (2) `spctl --assess -vv` output for all Gatekeeper policy changes applied fleet-wide as a configuration audit trail; (3) pipeline execution logs confirming the YARA pre-submission gate is firing correctly on test samples containing known prompt injection strings. These form the evidentiary record that eradication and hardening were effective.

Step 5: Post-Incident — Review the degree of autonomous decision-making authority granted to AI triage tools in your SOC workflow. Where AI automation has authority to suppress or deprioritize alerts without human review, introduce mandatory human-in-the-loop checkpoints for low-confidence verdicts. Apply NIST IR-8 (Incident Response Plan) to formally document this control gap and update playbooks. Reference CIS 7.1 (Establish and Maintain a Vulnerability Management Process) to incorporate AI toolchain integrity as a standing review item. Consider threat modeling exercises against your AI analysis pipeline using MITRE ATT&CK T1562.001 as a hypothesis anchor.

NIST Phase: Post Incident

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

Controls: NIST IR-8 (Incident Response Plan), NIST IR-2 (Incident Response Training), NIST IR-3 (Incident Response Testing), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Conduct a facilitated tabletop exercise where the 2-person SOC team manually traces the path a Gaslight-style prompt-injected sample would take through the current triage workflow, identifying every decision point where automated suppression is possible without analyst confirmation. Document each such point as a control gap in the updated IR playbook. Establish a quarterly review checklist for AI triage tool configuration that includes: (a) audit of suppression rule authority levels, (b) review of low-confidence verdict handling policies, and (c) a YARA rule refresh targeting new prompt injection patterns. No commercial tooling required — this is a process and documentation exercise.

Evidence: For the lessons-learned record: (1) the full timeline of LLM analysis aborts and suppressed verdicts across the 90-day exposure window, annotated with which samples were later confirmed malicious; (2) the pipeline configuration state at time of compromise documenting which components held autonomous suppression authority — this is the root-cause artifact; (3) the updated IR playbook sections governing AI triage authority limits and human-in-the-loop thresholds, version-controlled and dated, to demonstrate the control gap was formally closed.

Detection Guidance

Primary detection focus is on two parallel surfaces: macOS endpoint behavior and AI pipeline anomalies. On macOS endpoints, query EDR telemetry for execution of Rust-compiled binaries (identifiable by Rust-specific runtime signatures in binary metadata) that lack valid Apple notarization or developer code signatures. Flag binaries spawning shell interpreters (T1059) or attempting to modify security tool configurations (T1562.001). On AI-assisted analysis pipelines, instrument your LLM-based malware analysis layer to log all analysis abort events, refusals, and low-confidence returns on macOS samples; a spike in these outcomes for a specific sample or file family warrants immediate manual review. Behavioral indicators consistent with T1027 include samples with heavily obfuscated metadata fields or unusual Unicode sequences in embedded strings, a common vehicle for prompt injection payloads. Apply NIST AU-2 (Event Logging) to ensure macOS endpoint events and pipeline analysis events are captured in scope. No confirmed IOCs (hashes, IPs, domains) are available in the provided source data; detection must rely on behavioral and heuristic indicators until authoritative IOCs are published by a primary source.

Framework Mappings

MITRE-ATTACK

- **T1027** — Obfuscated Files or Information
- **T1059** — Command and Scripting Interpreter
- **T1562.001** — Disable or Modify Tools
- **T1204** — User Execution

NIST-800-53R5

- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CM-7** — Least Functionality
- **SI-7** — Software, Firmware, and Information Integrity
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SI-10** — Information Input Validation

OWASP-TOP10-2021

- **A03:2021** — Injection

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **8.2** — Collect Audit Logs

ISO-27001-2022

- **A.8.26** — Application security requirements

NIST-CSF-2

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1027	Obfuscated Files or Information	Defense-Evasion
T1059	Command and Scripting Interpreter	Execution
T1562.001	Disable or Modify Tools	Defense-Evasion
T1204	User Execution	Execution

Sources

Source	URL	Tier
Uses of AI in Enterprise Cybersecurity: Risks, Opportunities, Strategies	https://www.eye.security/blog/uses-of-ai-in-enterprise-cybersecurit...	T3
AI Cybersecurity: 6 Solutions Transforming AppSec and the SOC	https://checkmarx.com/learn/ai-security/ai-cybersecurity-6-solution...	T3
AI SOC Implementation for Enterprise Security Teams - Swimlane	https://swimlane.com/blog/ai-soc-implementation-enterprise/	T3
Cybersentinel AI for local machine cybersecurity - Facebook	https://www.facebook.com/groups/cto.platform/posts/2502142080231819/	T3
SOC automation trends and strategies : How AI and ... - Seceon	https://seceon.com/soc-automation-trends-and-strategies-how-ai-and-...	T3

Generated 2026-06-28 07:04 UTC by TJS Security Command Center