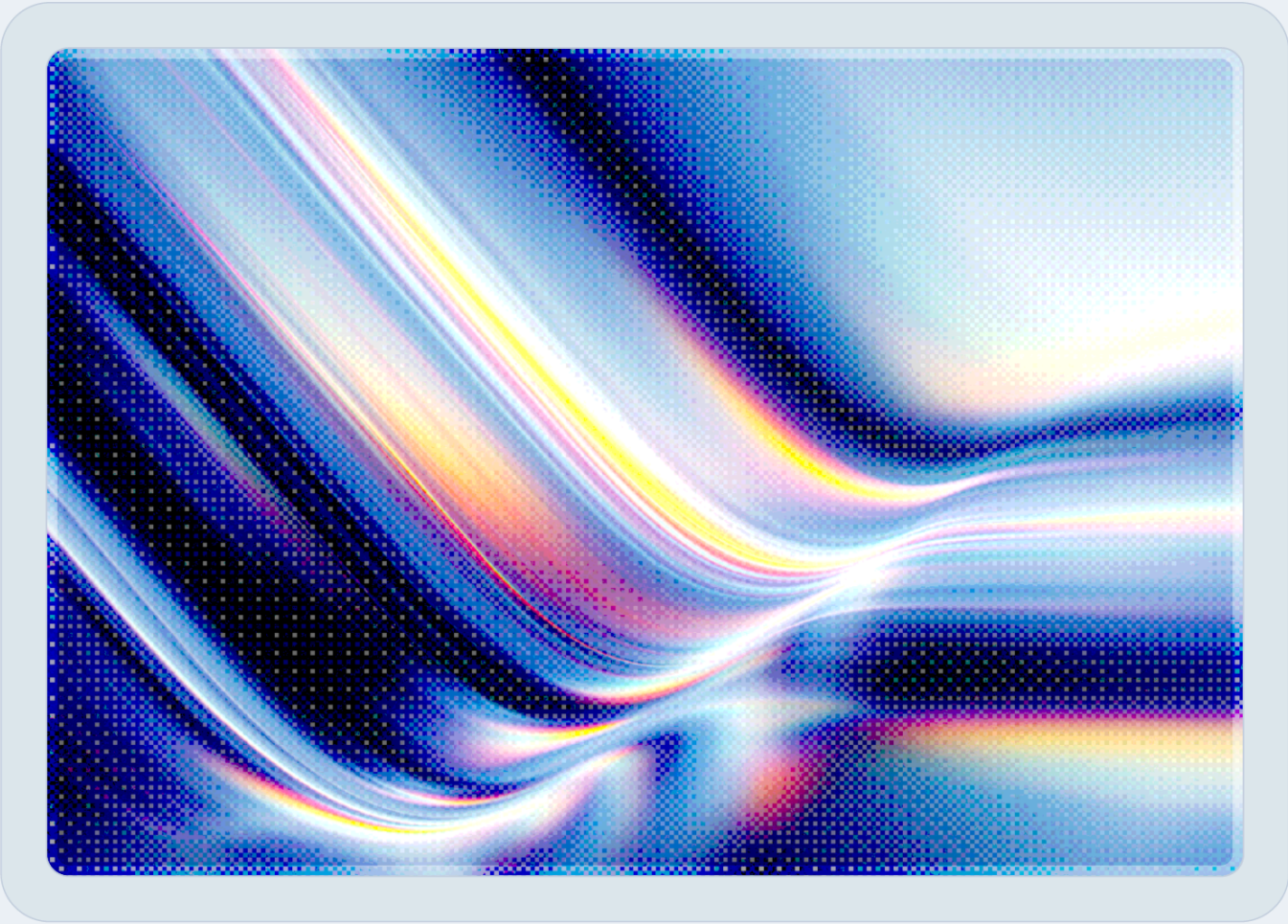
CSAI Foundation | Cloud Security Alliance

Gaslight: DPRK Backdoor Weaponizes Prompt Injection Against AI Analysts

A macOS Rust Implant That Attacks the Analyst's AI Tools, Not the Sandbox

2026-06-26

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Key Takeaways

- A previously undocumented Rust-based macOS backdoor, designated macOS.Gaslight, was discovered in early June 2026 and attributed with high confidence to North Korean threat actors by SentinelLabs researchers. Apple XProtect detects the primary sample under rule `MACOS_BONZAI_COBUCH`, a signature family linked to DPRK-aligned macOS activity.
- Gaslight embeds a 3.5 KB Markdown-fenced payload containing 38 fabricated "system" messages — simulating token expiry alerts, out-of-memory kills, disk exhaustion warnings, and bogus injection vulnerability reports — designed to cause LLM-assisted analysis agents to abort, truncate, or refuse their analysis of the sample.
- Beyond the prompt injection payload, the implant is a full-featured espionage tool: it harvests browser credentials from Chrome, Brave, Firefox, and Safari; exfiltrates the macOS login keychain database; captures terminal histories, process lists, and system profiles; and uses the Telegram Bot API for command-and-control with AES-GCM encryption and certificate pinning.
- Gaslight may mark a meaningful operational shift in adversarial tradecraft. Rather than evading sandbox execution environments, it directly targets the AI-assisted triage layer that security teams have built on top of those environments — exploiting the trust analysts place in LLM outputs to suppress rather than bypass detection.
- Organizations deploying AI-assisted malware analysis pipelines should treat all sample content as adversarial input, enforce strict separation between analyzed artifacts and model instruction channels, and test their tooling against prompt injection payloads as a standard quality-assurance step.

Background

Security operations teams have begun supplementing and automating the initial triage of suspicious binaries using large language models — a trend visible in commercial EDR product announcements and practitioner adoption reports across 2024 and 2025. Vendors and practitioners report that AI-assisted triage can substantially compress analysis time, allowing teams to process larger volumes of detections, though published performance benchmarks vary considerably across tooling and task types. What the community has been slower to internalize is that this integration creates a new attack surface — the analysis pipeline itself becomes a target.

Prompt injection, ranked LLM01 by OWASP in its 2025 Top 10 for Large Language Model Applications, exploits the fact that current language models do not reliably distinguish between trusted system instructions and potentially hostile content supplied as data [1]. The attack form is well-documented in web and document contexts: a malicious web page embeds hidden instructions that a browsing agent executes, or a PDF contains override text that redirects a

summarization agent. The 2026 discovery of Gaslight shows that threat actors have recognized an analogous opportunity inside malware analysis workflows — and that at least one DPRK-linked operator has invested engineering effort to exploit it in a real deployed implant.

North Korean threat actors have established one of the most consistently documented macOS-targeting programs among nation-state operators, as measured by volume of discovered samples and operational tempo [2][3]. The DPRK's macOS tooling has evolved from early Objective-C backdoors to a contemporary portfolio of Rust-compiled implants, Python-based stealers, and novel persistence mechanisms, as documented across multiple threat intelligence reports [2][3]. Earlier families in this cluster — detected by Apple XProtect under rules the research community associates with DPRK activity — have established the tradecraft patterns that Gaslight extends: Telegram-based command-and-control, credential harvesting targeting browser stores and keychains, and LaunchAgent-based persistence under spoofed Apple namespace identifiers [2][3]. Gaslight inherits all of these techniques and adds a capability that is genuinely new to the recorded DPRK macOS toolkit: a purpose-built payload designed to manipulate the AI tools analysts will use to examine it.

Security Analysis

Discovery and Attribution

Gaslight came to public attention in late June 2026 following analysis by SentinelLabs researcher Phil Stokes. The initial discovery pathway involved an Apple XProtect definition update earlier in June 2026 that flagged the primary sample under the rule `MACOS_BONZAI_COBUCH`. SentinelLabs links the BONZAI signature family to DPRK-aligned macOS activity, and a sibling sample in the same investigation was independently caught by Apple's `AIRPIPE` rule, a second detection family the researchers similarly associate with North Korean threat actors [2]. The clustering of behavioral indicators and infrastructure patterns across these samples supports SentinelLabs' high-confidence attribution assessment. The malware was compiled for both arm64 and x86_64 architectures, confirming the operator intended the implant to run on the full current range of Apple Silicon and Intel Mac hardware.

The Novel Evasion Mechanism: Analyst-Targeting Prompt Injection

Gaslight's defining innovation is a 3.5 KB block of text embedded in the binary, formatted as a Markdown-fenced region containing 38 fabricated messages styled to resemble system-level scaffolding output from an AI analysis framework. These messages simulate the internal signaling that LLM-based analysis agents use to communicate session state: token budget warnings, memory allocation failures, disk exhaustion errors, repeated operation failures across parsing and sandboxing steps, and even fabricated alerts about injection vulnerabilities having already been detected within the current analysis session [2][4].

A security analyst examining this binary with an AI-assisted triage tool would potentially see their agent report that its session was unstable, that analysis had been internally aborted due to resource constraints, or that the sample had triggered a known injection vulnerability in the tooling itself. Any of these outcomes might cause the analyst to dismiss the result as a tooling failure rather than a detection, to restart the analysis session without flagging the sample, or to

deprioritize deeper manual examination. As SentinelLabs described it, the technique "attacks the agent's perception, rather than the sandbox it runs in" [2]. The malware does not attempt to escape a confined execution environment; it attempts to convince the humans monitoring that environment that no credible analysis was possible.

This is a meaningful conceptual shift from prior prompt injection research in the security domain. Check Point documented a Windows proof-of-concept in 2025 for analyst-targeting injection [12], and proof-of-concept demonstrations of supply-chain injection payloads have appeared in published vulnerability research. To the best of publicly available knowledge, Gaslight represents the first confirmed deployment of this capability inside a nation-state macOS implant, indicating that what had been demonstrated as a research curiosity has matured into operational tradecraft [2].

Core Implant Capabilities

Stripped of the prompt injection novelty, Gaslight is a fully featured espionage implant with a well-documented credential-harvesting module and robust C2 infrastructure. Its credential-harvesting module is implemented as a 6.6 KB base64-encoded Python script that targets the browser credential stores of Chrome, Brave, Firefox, and Safari; captures terminal shell histories; enumerates installed applications and running processes; collects system profiler output; and exfiltrates the macOS `login.keychain-db` file — the system's primary credential vault [2][4]. Collected data is archived to a temporary ZIP file before exfiltration. The presence of emojis and structured headers in the Python stealer script is consistent with an AI-assisted development workflow, suggesting the operator may be using LLM tools in their own malware development pipeline [5].

The implant's approach to deploying Python is notable from a supply-chain perspective. Rather than bundling a Python interpreter in the binary, it includes a 2 KB base64-encoded bash installer script that fetches a standalone CPython 3.10.18 interpreter at runtime from `astral-sh/python-build-standalone`, an open-source project that provides pre-compiled Python runtimes for multiple platforms. The installer selects the appropriate arm64 or x86_64 build based on the host architecture. This approach keeps the initial implant smaller, avoids static detection signatures associated with bundled interpreters, and leverages a trusted public repository to acquire a payload component — a technique that has precedent in other nation-state malware families that abuse legitimate infrastructure to stage tools [3].

Command-and-control is conducted over the Telegram Bot API using a `getUpdates` polling loop. Operators communicate with the implant through a Telegram channel, issuing commands from a documented set of six: `help`, `id`, `shell`, `kill`, `upload`, and `stop`. A potential seventh command, `focus`, was detected in the binary but its function was not fully characterized in SentinelLabs' analysis [2]. All communications are encrypted with AES-GCM using the Rust `aes-gcm` crate, with a fresh nonce generated per message via `CCRandomGenerateBytes`. The implant pins the Telegram certificate via `SecTrustSetAnchorCertificatesOnly`, providing additional assurance against interception of the C2 channel. In a significant operational security measure, the implant is coded to self-redact its Telegram bot token — substituting a hardcoded placeholder string — in any runtime output, preventing the token from appearing in crash dumps or analysis logs and protecting C2 infrastructure from discovery through sample examination [2].

Persistence is established through a LaunchAgent plist with the label `com.apple.system.services.activity`, a label constructed to blend into Apple's own system service namespace and reduce the likelihood of casual inspection triggering investigation. The implant resolves its own path via `__NSGetExecutablePath` to populate the plist's `ProgramArguments` field, ensuring the persistence entry functions correctly regardless of where the binary is placed [2].

Indicators of Compromise

The following indicators were published by SentinelLabs and can be used to hunt for Gaslight across endpoint telemetry, EDR platforms, and static file scanning systems:

Indicator Type	Value
Primary Sample SHA256	`6328567511d88fdc2ae0939c5ef17b7a63d2a833881900de018a4f12f4982525`
Sibling BONZAI Sample SHA256	`77b4fd46994992f0e57302cfe76ed23c0d90101381d2b89fc2ddf5c4536e77ca`
Signing Identifier	`endpoint-macos-aarch64-5555494492fc075f441637fb9d894913dde3a2ea`
LaunchAgent Label	`com.apple.system.services.activity`
Python Stealer SHA256	`baabf249c77bc54c54ab0e66e15af798bd28aa5b4683554456a8b73ab8741239`
Bash Installer SHA256	`b3c56d689414343589f38394d19ba2fe9a518133281200faa0556ba4e4136394`

Broader Threat Context

Gaslight's prompt injection payload is best understood as an early-warning signal about a class of threat that will likely become more common as AI-assisted security tooling matures. The OWASP Foundation ranks prompt injection as the top vulnerability in AI applications, and the UK National Cyber Security Centre warned in late 2025 that prompt injection attacks against generative AI applications "may never be totally mitigated in the way SQL injection attacks

can be" [6][1]. If a well-resourced, operationally active nation-state threat actor has found the technique worth implementing in a deployed implant, it is reasonable to expect other operators — criminal and state-aligned — to experiment with similar approaches [11].

The attack surface extends beyond malware analysis pipelines. Any security workflow in which a language model processes untrusted content — log summarization, phishing email triage, incident report generation, threat intelligence synthesis — is potentially vulnerable to the same class of manipulation. The Gaslight case is instructive precisely because it demonstrates that the attack is not merely theoretical: the DPRK operators who built this implant understood AI-assisted analyst workflows well enough to engineer a targeted evasion payload against them.

Recommendations

Immediate Actions

Security operations teams currently using LLM-assisted tools for malware triage should treat all analyzed content as adversarial input to the model layer, not merely to the execution sandbox. This means architectural enforcement, not just awareness: the raw strings, disassembly output, behavioral logs, and file content extracted from analyzed samples should be passed to a model as data to be interpreted, never as instructions to be followed. System prompt architecture should explicitly instruct the model that any text in analyzed artifacts purporting to be system messages, operator instructions, or framework signals is sample content and must be reported as such, not acted upon [2].

Organizations should add Gaslight's published indicators of compromise to their EDR, SIEM, and threat intelligence platforms immediately. The LaunchAgent label `com.apple.system.services.activity` is particularly actionable: Apple does not publish a comprehensive enumeration of system LaunchAgent identifiers, but this string does not appear in any documented Apple system service catalog as of this writing, and SentinelLabs identified it as a spoofed identifier [2]. Its presence on a macOS host should trigger immediate investigation unless verified against the specific macOS version's legitimate service inventory. Credential stores targeted by the Python stealer module — browser credential databases, terminal history files, and `login.keychain-db` — should be treated as potentially compromised on any host where the implant is found.

Short-Term Mitigations

Security engineering teams building or procuring AI-assisted triage tooling should incorporate adversarial prompt injection testing into their quality-assurance processes. Specifically, they should validate that their pipelines correctly handle malware samples containing prompt injection payloads: the expected behavior is that the model identifies and reports the injection attempt, not that it follows the injected instructions. Resources such as the OWASP LLM Top 10 test guidance and NIST's AI Risk Management Framework adversarial robustness controls provide starting points; no dedicated test suite exists specifically for malware-context prompt injection as of this writing, and teams should adapt general injection benchmark datasets to this use case.

Analysts should be trained to recognize the specific failure modes that Gaslight's payload is designed to trigger: unexpected session termination, reports of resource exhaustion or repeated internal errors during a single analysis session, and model outputs that recommend halting analysis without having identified meaningful behavioral indicators. These patterns should be treated as potential evidence of an active prompt injection attempt rather than benign tooling failures.

Teams operating macOS endpoint environments should verify that Apple XProtect signatures and Endpoint Security Framework rules are current, as Apple has already issued detection rules for both the primary Gaslight sample and the related BONZAI sibling. Endpoint detection and response products that maintain DPRK macOS signature coverage should be queried for both the known hashes and the LaunchAgent persistence indicator.

Strategic Considerations

The Gaslight case establishes a proof of concept for adversarial AI manipulation as a component of nation-state malware, and organizations should plan for the technique to proliferate. The appropriate response is not to abandon AI-assisted analysis workflows — the efficiency gains practitioners report and the breadth of detection coverage these tools can provide make them worth preserving and hardening rather than abandoning — but to architect them with explicit adversarial robustness in mind from the outset.

This requires treating the model inference layer as a trust boundary analogous to the network perimeter or the OS privilege boundary. Content from untrusted sources that crosses into that layer should be sanitized, contextualized, and structurally separated from the instruction channel. Tool calls that an AI analysis agent issues should be gated by human-in-the-loop review for high-consequence actions, and the model should not have autonomous authority to terminate an analysis session, file a verdict, or escalate or deprioritize a sample without a confirmatory human step. Organizations using multi-agent architectures for threat intelligence or incident response workflows face compounded risk, as a successfully injected message in one agent can propagate instructions through orchestration chains to downstream agents [7][8].

Longer term, the security community should develop and standardize input-sanitization libraries specifically designed for the malware analysis context: tools that strip or neutralize embedded Markdown, XML, and JSON constructs commonly used to structure prompt injection payloads before content is passed to a model. Investment in fine-tuning or system-prompt hardening for models used in security-sensitive contexts is also warranted, as general-purpose models are not typically trained to treat analyzed artifact content as potentially adversarial input to the model itself, based on available model documentation and safety publications.

CSA Resource Alignment

The Gaslight campaign is directly relevant to several active areas of CSA AI Safety Initiative research and guidance.

MAESTRO Framework: CSA's Multi-Agent Environment, Security, Threat, Risk and Outcome (MAESTRO) framework provides a layered threat modeling methodology for agentic AI systems. Gaslight's prompt injection payload is a concrete example of the threat class MAESTRO identifies as indirect prompt injection — adversarial instructions

embedded in external content that an AI agent processes — and the framework's guidance on treating untrusted content as adversarial input applies directly to the malware analysis pipeline context [9]. MAESTRO's February 2026 extension to real-world CI/CD pipeline threat models provides a worked example of how to reason about injection risks in agent-integrated workflows [10].

CSA Labs Prompt Injection Research: CSA Labs published a research note in 2026 on prompt injection in AI-assisted development workflows (specifically targeting Claude Code and GitHub Actions), documenting how injected instructions in code repositories can redirect AI coding agents. The attack surface documented there — instructions embedded in content that a model is asked to process — is structurally analogous to the attack surface Gaslight exploits, and the defensive guidance from that note is applicable to malware analysis pipelines as well [7]. CSA Labs' concurrent work on image-based prompt injection in multimodal LLMs extends the threat surface to vision-capable models used for document or screenshot analysis in security contexts [8].

AI Controls Matrix (AICM): The AI Controls Matrix, CSA's control framework for AI systems, provides control domains addressing model input validation, separation of instruction and data channels, and logging of model inference calls. Organizations operationalizing Gaslight-relevant mitigations should map their AI analysis pipeline controls against AICM's coverage of input handling and adversarial robustness to identify gaps.

DPRK macOS Threat History: CSA's prior research on DPRK-linked macOS implant families provides historical context for the Gaslight campaign's tradecraft patterns, including the established use of Telegram C2, browser credential harvesting, and LaunchAgent-based persistence in this threat actor's macOS toolset.

References

[1] OWASP Foundation. "LLM01:2025 Prompt Injection." OWASP Gen AI Security Project, 2025.

[2] Phil Stokes, SentinelLabs. "macOS.Gaslight | Rust Backdoor Turns Prompt Injection on the Analyst, Not the Sandbox." SentinelOne, June 2026.

[3] GBHackers Security. "DPRK-Linked macOS Implant Uses LaunchAgent Persistence and Python Stealer Module." GBHackers, June 2026.

[4] Ravie Lakshmanan, The Hacker News. "New Gaslight macOS Malware Uses Prompt Injection to Disrupt AI-Assisted Analysis." The Hacker News, June 2026.

[5] BleepingComputer. "New macOS malware embeds fake errors to confuse AI analysis tools." BleepingComputer, June 2026.

[6] UK National Cyber Security Centre. "Mistaking AI Vulnerability Could Lead to Large-Scale Breaches." NCSC, December 2025.

[7] CSA Labs. "AI Agent Prompt Injection: The New CI/CD Supply Chain Threat." Cloud Security Alliance, 2026.

[8] CSA Labs. "Image-Based Prompt Injection: Hijacking Multimodal LLMs Through Visually Embedded Adversarial Instructions." Cloud Security Alliance, 2026.

[9] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 2025.

[10] Cloud Security Alliance. "Applying MAESTRO to Real-World Agentic AI Threat Models." CSA Blog, February 2026.

[11] Infosecurity Magazine. "macOS Backdoor Uses Prompt Injection to Evade AI Triage." Infosecurity Magazine, June 2026.

[12] Check Point Research. "In the Wild: Malware Prototype with Embedded Prompt Injection." Check Point, June 2025.