CSAI

cloud security alliance®

CSAI Foundation

Cloud Security Alliance AI Safety Initiative

Promptware: When Prompt Injection Becomes C2

Promptware-Powered Command and Control and the Operational Attack Framework Against AI Agents

Unofficial AI-assisted Research

2026-04-06

© 2026 Cloud Security Alliance. Unofficial AI-assisted Research.

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

- **Prompt injection, once analyzed primarily as an isolated input-validation failure, has been demonstrated as an operational attack class.** Researchers have documented full kill chains — initial access, persistence, lateral movement, and data exfiltration — executed entirely through crafted text inputs to AI agents.
- **"Promptware" names this evolution.** Coined in a January 2026 academic paper co-authored by Bruce Schneier, the term describes prompt-based attacks that traverse multiple stages of a structured kill chain, including command-and-control infrastructure implemented entirely through AI context manipulation [1][2].
- **Researchers have demonstrated working C2 channels over AI systems.** Exploits against ChatGPT, Google Jules, Microsoft Copilot, and Grok have shown that production AI platforms can be turned into command relays, beaconing agents, and data exfiltration channels without any malware or compromised infrastructure [3][4][5].
- **Multi-agent pipelines amplify the attack surface.** Indirect prompt injection propagating through RAG databases, MCP tool outputs, and agent-to-agent communication protocols can spread attacker instructions across an entire enterprise AI deployment from a single poisoned document [6][7].
- **No single control has been demonstrated to reliably prevent this class of attacks across all documented variants.** Defense-in-depth combining least privilege, sandboxing, instruction-data separation, and human oversight gates reflects the converging guidance from OWASP [8], MITRE ATLAS [19], and other frameworks reviewed in this note.

Background

Prompt injection, as a vulnerability class, was first formally described in September 2022 [26] and quickly recognized as a fundamental challenge in AI system design. In its original conception, it described a relatively contained problem: an attacker might trick a chatbot into revealing its system prompt, ignoring user safety constraints, or producing harmful output. OWASP classified it as LLM01 — the top-

ranked vulnerability in its Top 10 for Large Language Model Applications — in both its 2023 and 2025 editions [8]. At the time, the concern was largely about individual sessions and individual models. The attack surface was bounded.

That boundary no longer holds. The deployment of AI agents — systems that couple language models with persistent memory, tool use, file system access, web browsing, code execution, and the ability to invoke other agents — has fundamentally changed the threat calculus. An injection that once affected a single conversation can now propagate across workflows, persist across sessions, and direct real-world actions: sending emails, modifying code repositories, exfiltrating credentials, and making API calls on behalf of the targeted user or organization. The attack surface has grown from a model's context window to the full operational footprint of an enterprise AI deployment.

This shift prompted a formal reassessment of how the security community should categorize and reason about these attacks. A January 2026 paper by researchers Oleg Brodt, Elad Feldman, Bruce Schneier, and Ben Nassi proposed the term **promptware** to describe this evolved class of threat [1]. Analyzing 36 documented attacks, they found that at least 21 traverse four or more stages of a structured kill chain — demonstrating that prompt-based attacks now exhibit the same operational complexity as traditional malware campaigns. Lawfare, in its coverage of the paper, characterized the findings as evidence that "the AI ecosystem has developed an attack surface comparable in sophistication to what took decades to emerge in traditional software" [9][23].

The coinage of "promptware" reflects a necessary conceptual shift. Treating each prompt injection as an isolated input-validation failure may cause practitioners to underestimate the degree to which these techniques have been combined, chained, and structured into coherent attack campaigns. The remainder of this note examines the current state of that development.

Security Analysis

The Promptware Kill Chain

The promptware concept formalizes what security researchers have been observing in practice: that prompt-based attacks now map onto the same kill-chain taxonomy used to analyze traditional intrusion campaigns. Initial access is achieved through indirect prompt injection — instructions embedded in content that an AI agent is designed to retrieve and process. A poisoned email, a malicious PDF, a compromised web page, or a tainted entry in a shared RAG database can each serve as the delivery mechanism, triggering the attack the moment an agent processes the content in the normal course of its operation.

Persistence is achieved by instructing the agent to modify something it will read again — its own long-term memory, a configuration file it loads on startup, or an entry in the vector database it consults for every query. Researcher Johann Rehberger demonstrated this concretely against ChatGPT: a single indirect injection, delivered via a malicious website viewed by a user, planted persistent instructions in ChatGPT's memory store [10]. On every subsequent session, the model applied attacker-authored behavioral modifications without user notification. The payload survived logout, browser restarts, and standard session cleanup.

Lateral movement in multi-agent architectures follows the agent-to-agent communication pathways that make these systems useful. When one agent passes outputs to another as trusted inputs — a common design pattern in LangChain pipelines, AutoGen frameworks, and custom agentic workflows — a compromised agent becomes a vector for infecting its peers. Researchers have documented propagation patterns in which poisoned documents can successfully traverse multiple agent layers. Controlled evaluations of prompt injection techniques against state-of-the-art defenses have reported success rates exceeding 85% when adaptive strategies are employed, with variation across architectures and attack configurations [18].

Actions on objective run the gamut of whatever capabilities the compromised agent has been granted. Documented real-world impacts include: exfiltration of OneDrive files and SharePoint documents via Microsoft 365 Copilot [11]; theft of private repository source code and cryptographic keys through a compromised GitHub MCP integration [7]; unauthorized data exports from enterprise service management platforms through peer-agent trust exploitation, as surveyed in the literature [12]; and persistent remote code execution on developer workstations through coding agent compromise [13].

C2 Infrastructure Through AI Context

Perhaps the most operationally significant development in this threat class is the construction of command-and-control infrastructure that uses AI platforms as the communication layer. Traditional C2 architectures require attacker-controlled servers, domain registration, and network infrastructure that can be detected through threat intelligence, firewall rules, and network monitoring. Promptware-native C2 requires none of these.

Rehberger's ZombAI research, published in January 2025, demonstrated the first full implementation of this concept [3]. By exploiting ChatGPT's memory persistence feature, an attacker could establish a beaconing agent without any dedicated infrastructure. On each session, the compromised ChatGPT instance would check a GitHub Issues page — a trusted, allowlisted domain — for new attacker commands. Exfiltrated data traveled outbound through Azure Blob Storage, also on Microsoft's allowlist. The entire C2 channel was invisible to standard network security controls because every connection it made appeared to be legitimate, expected traffic to trusted services.

In May 2025, the same researcher applied this approach to Google Jules, a production AI coding agent [4]. Jules was found to have unrestricted outbound internet connectivity. A prompt injection caused the agent to download and execute attacker instructions, harvest environment variables and credentials, and maintain persistent remote control over the developer's local machine — all through the agent's normal-appearing coding assistance workflow. As Rehberger reported [4], Google assessed the finding as an "abuse risk" and closed the disclosure without public acknowledgment of remediation.

Check Point Research published a formal analysis of this architectural pattern in February 2026 [5]. Their research demonstrated that AI assistants with web-browsing capabilities — specifically Grok and Microsoft Copilot — can be deployed as C2 relays with no API keys, no authenticated accounts, and no dedicated attacker infrastructure. Their proof-of-concept C++ implementation used Windows' WebView2 component to automate the attack without visible user interaction. Reconnaissance data was encoded into URL parameters; the AI's "summary" of an attacker-controlled page carried the return commands. The researchers noted that high-entropy encoding of the data was sufficient to bypass content safety filters, and warned of an emerging "AI-Driven malware" category in which AI reasoning capabilities are used at runtime to make decisions about sandbox detection, target prioritization, and exfiltration strategy.

Vectra AI researchers formalized a related concept they termed "prompt control" in March 2026 [14]. Their analysis identified three distinct patterns: a heartbeat mechanism in which agents are instructed to read attacker-controlled files at regular intervals; persistent context poisoning through a single initial injection that plants attacker instructions into files the agent reads routinely; and covert data routing through which exfiltrated information is blended into normal agent summarization output and routed to attacker-controlled destinations such as Telegram accounts. As Cardiet and Paredes write: "An attacker no longer needs a persistent channel if they can shape what the agent sees, remembers, and prioritizes. Control becomes indirect, continuous, embedded in normal operation" [14]. This structural insight is significant: the attack does not require an ongoing active connection — it requires only that attacker intent be embedded in the agent's operating context.

Production Vulnerabilities and CVE Landscape

The research described above is corroborated by a growing body of disclosed vulnerabilities in production AI systems. CVE-2025-32711, named EchoLeak, affected Microsoft 365 Copilot with a CVSS score of 9.3 (Critical, per Microsoft CNA assessment; NVD independently scores the same CVE at 7.5 High) [11]. The attack required no user interaction: a crafted email containing hidden prompt injection instructions caused Copilot to exfiltrate the victim's chat logs, OneDrive files, SharePoint documents, and Teams messages without visible indication. The attack chained bypasses of Microsoft's cross-

prompt injection attack (XPIA) classifier, link redaction, and the Teams proxy allowlist — demonstrating that even multi-layered platform defenses can be defeated through technique composition. Microsoft patched the vulnerability server-side following disclosure.

CVE-2025-53773, disclosed by Rehberger against GitHub Copilot, demonstrated a path from repository-embedded prompt injection to remote code execution [15]. Malicious instructions embedded in public repository code comments modified the target developer's VS Code settings to enable autonomous code execution, bypassing the approval step that is Copilot's primary user-facing safety control. CVE-2025-59944 demonstrated a related configuration file substitution technique against the Cursor agentic IDE. These disclosures collectively indicate that the IDE and developer toolchain layer — where AI agents have been granted extensive filesystem and execution access in the name of productivity — presents a significant attack surface that security researchers are actively investigating.

Palo Alto Unit 42 documented the first detected real-world malicious indirect prompt injection against a production AI deployment in December 2025 [16]. The target was an AI-based product advertisement review system. The attacker employed multiple simultaneous IDPI techniques with high-severity intent. Unit 42's broader telemetry across enterprise AI deployments documented 22 distinct in-the-wild indirect prompt injection techniques, ranging from SEO poisoning of content retrieved by browsing agents to unauthorized payment transaction initiation. This represents the transition of promptware from proof-of-concept research into active threat actor tradecraft.

RAG Poisoning and Supply Chain Vectors

Retrieval-augmented generation architectures introduce a supply chain dimension to the promptware threat. When an AI agent's knowledge base — its vector store, its embedded documentation, its indexed email archive — is treated as a trusted input channel, poisoning that knowledge base becomes equivalent to poisoning the agent's instruction stream. The PoisonedRAG research, presented at USENIX Security 2025, demonstrated that as few as five carefully crafted documents injected into a RAG database can manipulate AI responses 90% of the time [17]. The poisoned documents are semantically meaningful and unlikely to be flagged by content scanning tools; they appear legitimate until retrieved in the context of a targeted query.

MCP (Model Context Protocol) servers, which have become a standard integration layer between AI agents and enterprise tooling, introduce additional supply chain risk. Researchers have documented "tool shadowing" attacks in which malicious MCP servers exploit namespace collisions to intercept tool calls intended for legitimate servers [18]. When multiple MCP servers are active simultaneously — a common configuration in integrated development environments and enterprise AI deployments — an attacker who can introduce a malicious server into the namespace can invisibly redirect tool invocations, log sensitive

parameters, and modify outputs before they reach the agent. In tested configurations, neither the user nor the legitimate tool detected the interception — a behavior that standard monitoring is not currently designed to identify [18][25].

Recommendations

Immediate Actions

Security teams should treat any AI agent with tool-use capability as an endpoint requiring the same access controls applied to human users and service accounts. The principle of least privilege must be applied to agent permissions at the time of deployment, not retrofitted after an incident. An agent that needs to read calendar events should not have permission to send emails; an agent that needs to query a database should not have credentials for schema modification. For organizations that have already deployed agentic AI, reviewing current agent permission grants against the minimum required for their stated function is among the most impactful near-term actions, given that overpermissioned agents expand the blast radius of any successful injection.

For agents that process external content — web pages, uploaded documents, emails, third-party API responses — organizations should implement explicit trust labeling at the architectural boundary. Untrusted content should be processed in an isolated context that limits its ability to override system-level instructions. OWASP's mitigation guidance is direct on this point: "Separate and clearly denote untrusted content to limit its influence on user prompts" [8][24]. Implementing this separation requires awareness during agent design, not just at the model layer.

Human approval gates for irreversible or high-impact agent actions — credential use, external data transmission, file modification, email sending — provide a backstop when injection succeeds. A single injected instruction that cannot execute its intended action without a human approving an anomalous request is substantially less dangerous than one that can complete its objective without user awareness. This control should be applied proportionally to action impact; requiring approval for every low-stakes agent operation creates alert fatigue and user friction that erodes its effectiveness.

Short-Term Mitigations

Organizations should conduct an inventory of their AI deployments with specific attention to agent memory persistence mechanisms and outbound connectivity. Long-term memory features in AI assistants — particularly those that allow agents to store and retrieve user-provided information across

sessions — are the most readily exploitable persistence mechanism demonstrated in current research. Where these features are not required for business function, they should be disabled. Where they are required, access to memory should be logged, and anomalous memory entries should trigger review.

MCP deployments should be audited for namespace hygiene and server provenance. Organizations should maintain allowlists of approved MCP servers and block unapproved additions at the configuration management layer. Tool outputs from MCP servers should be treated as untrusted external data rather than trusted system instructions, particularly in agentic workflows where those outputs flow into subsequent agent steps.

Network egress from AI agent runtime environments should be scoped. Agents that serve internal purposes — document summarization, code review, data analysis — typically have no legitimate need for arbitrary outbound internet connectivity. Restricting egress to specifically approved destinations significantly raises the bar for the C2 channel pattern demonstrated in ZombAI and Jules Zombie Agent, though defenders should recognize that trusted-platform abuse may shift attacker focus to approved-but-abusable destinations such as cloud storage services the organization already permits [3][4].

Strategic Considerations

The MITRE ATLAS framework catalogs prompt injection as technique AML.T0051 under the Initial Access tactic, and provides a structured vocabulary for reasoning about the broader AI adversarial threat landscape [19]. The promptware kill chain additionally maps to several further ATLAS techniques — including AML.T0043 (craft adversarial data) and relevant supply chain and exfiltration tactics — providing a broader coverage picture when ATLAS is used for threat modeling. Organizations developing AI security programs should map their agent architectures against ATLAS to identify coverage gaps and prioritize controls. Google's December 2025 publication of a formal taxonomy distinguishing "task injection" from prompt injection signals that the threat modeling vocabulary for agentic AI is still maturing [20]; security teams should track this evolving taxonomy and update their threat models as it stabilizes.

The convergence of promptware with traditional malware tradecraft — particularly the use of AI platforms as C2 infrastructure — suggests that network and endpoint security tools will need to be updated to reason about AI-specific behavioral signals. An agent that periodically fetches from an unexpected domain, that begins inserting data into outbound summaries in encoded form, or that requests permissions it did not previously require may be exhibiting post-compromise behavior. SIEM and UEBA tools that ingest AI agent telemetry should be updated with detection logic for these patterns.

Red team exercises should include promptware scenarios. Testing AI agent deployments against realistic indirect injection payloads — malicious documents processed by RAG pipelines, poisoned tool outputs, crafted emails directed at AI-assisted email triage — will reveal practical exposure before threat actors discover it in production. The Trail of Bits research on argument injection and the Rehberger work on coding agent RCE provide published methodologies that can inform adversarial testing [13][21].

CSA Resource Alignment

This research note connects to several active workstreams within the Cloud Security Alliance AI Safety Initiative.

MAESTRO (Multi-Agent Environment and System Threat, Risk, and Opportunity framework) provides the primary threat modeling foundation for the multi-agent propagation patterns described in this note. The lateral movement and trust exploitation patterns documented in cross-agent injection attacks map directly to MAESTRO's Layer 3 and Layer 4 threat categories, covering agent orchestration and inter-agent communication vulnerabilities.

The CSA Agentic AI Security Initiative is developing guidance specifically for the deployment and operation of AI agents in enterprise contexts. The permission scoping and trust boundary recommendations in this note align with the Initiative's emerging least-privilege guidance for agentic systems. The CSA's Agentic NIST AI RMF Profile, available through CSA Labs [22], provides an actionable mapping of NIST AI 100-1 controls to agentic AI deployments and should be consulted alongside this research note.

CCM and AICM: The AI Controls Matrix (AICM), a superset of the Cloud Controls Matrix that incorporates AI-specific security requirements, provides a framework for mapping the technical controls described in the Recommendations section to organizational governance structures. The trust boundary controls, data handling requirements, and monitoring provisions in AICM are directly applicable to the promptware threat class.

STAR for AI and the catastrophic risk annex under development provide the risk assessment scaffolding for organizations seeking to evaluate their promptware exposure within a broader AI risk management program. The C2-via-AI patterns documented in this note represent a category of systemic risk — where a compromised AI deployment could serve as infrastructure for attacks on other organizations — that warrants consideration at the program level, not merely the deployment level.

References

[1] Brodt, O., Feldman, E., Schneier, B., Nassi, B. "The Promptware Kill Chain: How Prompt Injections Gradually Evolved Into a Multistep Malware Delivery Mechanism." arXiv:2601.09625, January 2026.

[2] Nassi, B., et al. "GenAI-Powered Applications are Vulnerable to PromptWares." arXiv:2408.05061, August 2024.

[3] Rehberger, J. "SPAIware and ChatGPT Command and Control via Prompt Injection – ZombAI." Embrace the Red, January 2025.

[4] Rehberger, J. "Jules Zombie Agent: From Prompt Injection to Remote Control." Embrace the Red, 2025.

[5] Check Point Research. "AI in the Middle: Turning Web-Based AI Services into C2 Proxies – The Future of AI-Driven Attacks." Check Point Research, February 2026.

[6] Medium / InstaTunnel. "Multi-Agent Infection Chains: The Viral Prompt and the Dawn of the AI Worm." Medium, 2026.

[7] Straiker. "Agent Hijacking: How Prompt Injection Leads to Full AI System Compromise." Straiker Security, 2025.

[8] OWASP. "LLM01:2025 – Prompt Injection." OWASP Top 10 for Large Language Model Applications, 2025.

[9] Lawfare. "The Promptware Kill Chain." Lawfare Media, February 2026.

[10] Rehberger, J. "ChatGPT Memory Hacking via Prompt Injection." Embrace the Red, 2024.

[11] Hackers Research / arXiv. "EchoLeak: CVE-2025-32711 Zero-Click Vulnerability in Microsoft 365 Copilot." arXiv:2509.10540, 2025. See also: Hack The Box deep-dive.

[12] MDPI Information. "Comprehensive Review of Prompt Injection Vulnerabilities in Large Language Models." MDPI Information, 2025.

[13] Trail of Bits. "Prompt Injection to RCE in AI Agents." Trail of Bits Blog, October 2025.

[14] Cardiet, L., Paredes, M. "Prompt Control: How Context Becomes the Command and Control Layer for AI Agents." Vectra AI, March 2026.

[15] Rehberger, J. "GitHub Copilot Remote Code Execution via Prompt Injection (CVE-2025-53773)." Embrace the Red, 2025.

[16] Palo Alto Unit 42. "Real-World Case of Malicious Indirect Prompt Injection." Unit 42 Threat Intelligence, December 2025. See also: Unit 42 blog.

[17] Zhong, Z., et al. "PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models." USENIX Security Symposium 2025, pp. 3827–3844. arXiv:2402.07867.

[18] Levi, Y., et al. "From Prompt Injections to C2: Attacking Agentic Coding Assistants." arXiv:2601.17548v1, January 2026.

[19] MITRE. "MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems." MITRE Corporation, 2025.

[20] Google Bug Hunters. "Task Injection: Exploiting the Agency of Autonomous AI Agents." Google Security, December 2025.

[21] Rehberger, J. "Google Jules Invisible Prompt Injection." Embrace the Red, 2025.

[22] CSA Labs. "Agentic NIST AI RMF Profile v1." Cloud Security Alliance, 2025.

[23] Schneier, B. "The Promptware Kill Chain." Schneier on Security, February 2026.

[24] OWASP. "AI Agent Security Cheat Sheet." OWASP CheatSheet Series, 2025.

[25] Rahali, A., et al. "From Prompt Injections to Protocol Exploits: Security Implications in Multi-Agent LLM Systems." ICT Express, Vol. 12, Issue 2, April 2026.

[26] Willison, S. "Prompt injection attacks against GPT-3." Simon Willison's Weblog, September 12, 2022.