PleaseFix: Zero-Click Browser Agent Hijacking

The Emerging Attack Class Targeting Autonomous Web Agents

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-28

Key Takeaways

- On March 3, 2026, Zenity Labs disclosed **PleaseFix**, a vulnerability family affecting Perplexity Comet and other agentic browsers. The attacks require no user clicks, no malicious code execution on the victim's device, and no social engineering — only a crafted piece of content that the agent processes during routine tasks. [1]
- PleaseFix is not an isolated flaw. It represents a named class of **low-friction agentic exploits** — a category that includes ZombieAgent (OpenAI Deep Research), GeminiJack (Google Gemini), Tainted Memories (OpenAI Atlas), and HashJack (Cato Networks research), all requiring zero user interaction, as well as closely related low-interaction attacks such as CometJacking (Perplexity Comet), which requires one click. Collectively, these disclosures document a pattern rather than a series of unrelated bugs.
- The root cause shared across all of these attacks is a structural trust failure: agentic browsers inherit the user's authenticated session and process untrusted web content — emails, calendar invites, documents, web pages — without a validated boundary distinguishing legitimate user intent from adversarially injected instructions. [2]
- OWASP ranks prompt injection (LLM01:2025) as the top vulnerability class for large language model applications. [3] The PleaseFix class is a direct manifestation of this risk in browser-integrated deployment contexts.
- Organizations deploying or evaluating agentic browsers — including AI-native browsers, computer-use frameworks, and AI-augmented enterprise browsers — should treat indirect prompt injection as a first-class threat requiring immediate policy and technical controls.

Background

From ClickFix to PleaseFix

The name PleaseFix carries deliberate meaning. ClickFix is a social engineering technique in which victims are deceived into executing malicious payloads through dialog boxes or instructions that appear to request routine actions — "click here to fix this issue." The technique depends on a human being present to act on the instruction.

PleaseFix removes the human from this loop entirely. In an agentic browser context, an AI agent is empowered to act on the user's behalf: reading email, managing calendar entries, accessing cloud files, interacting with password managers. When that agent encounters a crafted piece of content containing an adversarial instruction, it may execute the instruction without surfacing it to the user at all. The attack requires no clicks, no downloads, no exploitable memory corruption, and no user awareness. The plea is made directly to the agent.

Agentic Browsers and Their Attack Surface

Agentic browsers represent a new product category distinct from conventional browsers. Where a standard browser renders content for a human to review, an agentic browser renders content on behalf of an AI system that then acts on what it perceives. Products such as Perplexity Comet and OpenAI Atlas connect to users' Gmail, Google Calendar, Google Drive, password managers, and productivity tools via OAuth, then execute multi-step tasks — booking meetings, retrieving files, composing and sending messages — autonomously and at the direction of natural language instructions. [1]

This architecture creates an attack surface for which existing security models provide only partial precedent. Prior automated systems — macro processors, web crawlers, email automation scripts — encountered some forms of content injection risk, but none combined LLM-scale instruction-following, persistent OAuth authorization across multiple enterprise services, and the autonomous action breadth characteristic of current agentic browsers. A conventional browser's security model is built around the assumption that a human reviews content before acting on it. An agentic browser's security model must account for the possibility that adversarially crafted content will be read and acted upon by an autonomous system that may have no mechanism to distinguish malicious intent from legitimate task context.

Zenity Labs researchers Michael Bargury and colleagues described this failure as "an agent trust failure that exposes data, credentials and workflows in ways existing security controls were never designed to see." [1] The statement is technically precise: conventional network security, endpoint detection, and

data loss prevention tools were designed to observe and filter traffic between humans and systems — not between autonomous agents and the services those agents control on the human's behalf.

Security Analysis

The PleaseFix Vulnerability Family

Zenity Labs disclosed PleaseFix on March 3, 2026, with Perplexity Comet as the primary affected system. [1][14] The disclosure identified a subfamily of two distinct exploit paths, labeled **PerplexedBrowser**, both of which exploit the same underlying trust failure through different delivery mechanisms and objectives.

The first path, zero-click file system exfiltration, works through the delivery of a malicious calendar invite or similarly ordinary content item to the victim. When the user later instructs Comet to manage a calendar-related task, the agent processes the malicious content in the course of that operation. The embedded adversarial instructions direct the agent to access local file system paths via `file://` URI scheme, then exfiltrate the retrieved data to an attacker-controlled endpoint. The agent simultaneously delivers the expected result of the legitimate task to the user, so the compromise remains invisible. Perplexity addressed this by implementing a hard boundary blocking autonomous `file://` URI access. [1]

The second path demonstrates credential theft through password manager manipulation. Rather than attacking 1Password's own code, the exploit leverages the trust relationship that Comet holds as a 1Password-integrated agent. The agent's inherited authorization allows an attacker who controls injected instructions to manipulate credential retrieval workflows or execute partial account takeover sequences using capabilities that were legitimately granted for user-directed tasks. Responsible disclosure was coordinated with both Perplexity and 1Password prior to publication. [1][2]

A Pattern Across the Industry

The significance of PleaseFix extends beyond the specific Perplexity Comet vulnerabilities. A review of comparable disclosures from the preceding year reveals that the same structural condition — an autonomous agent with broad OAuth permissions processing untrusted content — has been exploited independently across multiple platforms.

In August 2025, Amanda Rousseau at Straiker STAR Labs disclosed a zero-click attack against an AI agent's Gmail and Google Drive integration. The attack required no jailbreak and no explicit prompt injection trigger. Crafted emails using natural-language phrases such as "take care of" and "handle this on my behalf" were sufficient to shift task ownership to the agent, which then executed destructive file deletion instructions across the victim's Google Drive, including shared folders and team drives. [4]

In October 2025, LayerX Security disclosed **Tainted Memories**, an attack against OpenAI's Atlas browser. The attack used a cross-site request forgery vulnerability to inject malicious instructions into ChatGPT's long-term memory store. Once poisoned, those instructions persisted and executed across all devices and sessions where the victim's account was active. LayerX researchers demonstrated that this pathway could lead to remote code execution. [5] A separate LayerX finding, **CometJacking**, used crafted URL query parameters to achieve a one-click hijack of Perplexity Comet, enabling email and calendar data exfiltration. [6]

In November 2025, Cato Networks disclosed **HashJack**, an attack exploiting the tendency of browser agents to process URL fragment identifiers — the portion of a URL following the `#` symbol — even though conventional security tools typically ignore fragments as client-side-only data. By embedding adversarial instructions in URL fragments, attackers could deliver payloads through links that appeared clean to network inspection tools. [7]

Radware researcher Zvika Babo reported **ZombieAgent** to OpenAI in September 2025 and Radware publicly disclosed the technique in January 2026, demonstrating server-side prompt injection paths against OpenAI's Deep Research agent, including one that required no user interaction. [8] Noma Security's **GeminiJack** disclosure showed that shared Google Docs containing hidden instructions would be silently retrieved and executed by Google Gemini when employees used the assistant to search their workspace, resulting in leakage of Gmail, Calendar, and Docs data. [9]

This breadth of disclosure within a single year — across Perplexity, OpenAI, and Google products — argues strongly that the pattern reflects a structural gap in how agentic browser security is designed, not a collection of platform-specific implementation errors.

Technical Root Causes

Two foundational problems underlie the entire low-friction agentic exploit class.

The first is the absence of an instruction provenance mechanism. When a browser agent processes a page, an email, or a document, the current generation of systems does not reliably distinguish between instructions issued by the authenticated user and instructions embedded by a third party in content that the user's system fetched. The agent's instruction-following capability is, in the general case,

indiscriminate with respect to the source of the instruction. This differs meaningfully from conventional application injection vulnerabilities: where SQL injection exploits a parser's failure to distinguish data from code, indirect prompt injection exploits an AI agent's failure to distinguish user intent from adversarial instructions embedded in externally sourced content.

Researchers at Brave noted a further dimension of this problem: adversarial instructions can be rendered visually imperceptible to human reviewers through techniques such as using extremely low-contrast text (faint light-blue on yellow) that the human eye cannot resolve but that AI visual processing extracts as readable content. [10] This means that even if users were to review all content before agent processing — which the value proposition of autonomous agents explicitly forecloses — they could not reliably detect injected instructions.

The second root cause is excessive inherited permission. Agentic browsers are authorized by users to take broad action on their behalf, typically through OAuth scopes that grant read and write access to email, calendar, file storage, and connected applications. Those permissions are granted for the purpose of executing the user's instructions, but they are held in a way that is structurally accessible to any instruction the agent processes. An injected instruction that arrives via a calendar invite inherits the same capabilities as a legitimate user instruction. There is no runtime mechanism in current deployments that restricts agent capabilities to the scope of the specific task that a verified human instruction initiated.

The ArXiv paper "The Hidden Dangers of Browsing AI Agents" identified a related technical flaw in the browser-use Python library (CVE-2025-47241): URL parsing that fails to handle Basic Authentication credentials embedded in URLs allows attackers to bypass configured domain allowlists and perform server-side request forgery attacks against internal services. The same paper noted that positioning adversarial content at the end of prompts — where some LLM architectures show elevated responsiveness to instructions positioned near context boundaries, as documented in [11] — substantially increases injection success rates in tested configurations. [11]

Recommendations

Immediate Actions

Organizations that have deployed or are piloting agentic browsers should take several immediate steps to reduce exposure.

First, audit the OAuth scopes that have been granted to any agentic browser or AI assistant product. Permissions to read and write email, manage calendar events, access file storage, or interact with password managers should be reviewed against the specific workflows for which the product was deployed. Any granted scope that is not required for a currently active, authorized use case should be revoked. Minimum-privilege OAuth configurations should be enforced at the identity provider level, not relied upon to be handled by the agent product vendor.

Second, classify agentic browser activity as a distinct data access category in data loss prevention policies. Traffic from agentic browsers to external endpoints may not be distinguishable from user-initiated traffic under conventional DLP rules. Organizations should ensure that API calls made autonomously by AI agents — especially calls to data retrieval, file access, or messaging APIs — are logged with sufficient detail to reconstruct what content was accessed and what actions were taken.

Third, check whether vendor patches are current for any deployed agentic browser. Perplexity addressed the specific PleaseFix exploit paths prior to Zenity Labs' public disclosure. However, the disclosure also implicates a class of products — any agentic browser with broad OAuth authorization and autonomous processing of user-fetched content — that may have analogous unpatched exposures. Vendors should be queried for their current security posture regarding indirect prompt injection.

Short-Term Mitigations

In the near term, organizations should establish governance controls that reduce the blast radius of a successful agent hijack. Agent OAuth grants should be scoped to the minimum permission set required for the authorized task and should be reviewed and reauthorized on a defined cycle — not granted as persistent, broad authorizations. This corresponds to the principle of just-in-time, just-enough access applied to autonomous systems. [12]

Confirmation workflows — in which an agent must surface a human-readable summary of the specific actions it is about to take before executing any destructive or data-exfiltration-class operation — should be required for any agentic task that involves file deletion, outbound data transfer, credential access, or modification of shared resources. For operations in these categories, the risk reduction from requiring human confirmation is likely to outweigh the efficiency cost in most organizational contexts, given that a hijacked operation could cause irreversible harm.

Organizations using AI-integrated productivity environments (Google Workspace, Microsoft 365) should review whether AI assistant features that access shared documents, email threads, or calendar entries are operating with scopes that allow exfiltration of that data to external endpoints. The GeminiJack disclosure demonstrates that shared document environments are a viable delivery vector for indirect prompt injection payloads even when no browser agent product is explicitly deployed. [9]

Strategic Considerations

The structural nature of these vulnerabilities suggests that patch cycles alone will be insufficient to resolve the low-friction agentic exploit class. The structural condition that enables these attacks is deeply embedded in the current agentic browser architecture: these systems are designed to process untrusted content and act on it with broad authorization. Until that architecture changes — through instruction sandboxing, runtime privilege isolation, or equivalent controls — patch cycles will address symptoms rather than the underlying exposure.

Instruction provenance verification — the ability to cryptographically or probabilistically distinguish user-issued instructions from injected instructions embedded in third-party content — is an active research area. Current approaches include architectural sandboxing (processing untrusted content in a context that does not have access to agent tool invocation), instruction source tagging (labeling instruction tokens with their origin before processing), and dual-model review architectures (a separate model reviews proposed actions for anomalies before execution). To the authors' knowledge, none of these approaches has yet achieved robust production deployment at scale, but each represents a direction that security teams should monitor and vendors should be pressed to adopt.

Security evaluations of agentic browser products should explicitly include indirect prompt injection test cases covering each authorized integration scope. Red team exercises that deliver injection payloads through calendar invites, email threads, shared documents, and URL fragments — reflecting the documented real-world delivery vectors for this attack class — should be part of standard procurement security review for any agentic browser deployment. [15]

CSA Resource Alignment

This research note connects to several foundational CSA frameworks and publications that provide structural context for the PleaseFix vulnerability class and the broader low-friction agentic exploit category.

MAESTRO (Multi-Agent Environment, Security, Threat, Risk, and Outcome) provides the primary threat modeling vocabulary for this analysis. PleaseFix and related disclosures map principally to Layer 3 (Agent Frameworks) — where the prompt-processing architecture fails to enforce instruction provenance — and Layer 2 (Data Operations) — where untrusted content enters the agent's context without adequate trust boundary enforcement. Layer 6 (Security and Compliance) captures the gap in monitoring and detection capabilities noted throughout this analysis, as conventional DLP and network inspection tools are not designed to observe autonomous agent actions.

OWASP Top 10 for LLM Applications (LLM01:2025 – Prompt Injection) classifies indirect prompt injection specifically within its treatment of LLM01, distinguishing direct injection (manipulating the system prompt via user input) from indirect injection (embedding malicious instructions in external content processed by the agent). The PleaseFix class exemplifies the indirect injection pattern in a high-impact operational context where broad OAuth authorization amplifies blast radius.

The **CSA Agentic AI Red Teaming Guide** (2025, lead author Ken Huang) identifies "Goal and Instruction Manipulation" and "Agent Authorization and Control Hijacking" as two of its twelve critical threat categories. Both categories apply directly to the attack patterns analyzed here. The Guide's 629 security test cases and curated frameworks (AgentDojo, Agent-SafetyBench) provide a structured baseline for organizations developing evaluation protocols for their agentic browser deployments. [13]

CSA Agentic Identity Survey (2025) found that only 18% of organizations reported high confidence in their identity and access management capabilities as applied to AI agents, and that 82% could not ensure those capabilities were sufficient to manage agent identities at production scale. [12] The PleaseFix disclosures confirm the operational consequences of this gap: agents operating with broad, persistent OAuth authorizations and without runtime access scoping are structurally exposed to the full blast radius of any successful injection attack.

The **AICM (AI Controls Matrix)** – CSA's AI Controls Matrix, a superset of the Cloud Controls Matrix designed for AI system governance – provides control domains applicable to agentic browser deployments across identity, authorization, monitoring, and data governance dimensions. Organizations performing security assessment of agentic browser products should use the AICM as a structured control reference, particularly in the domains of AI runtime behavior monitoring and authorization lifecycle management.

References

[1] Zenity Labs, "Zenity Labs Discloses PleaseFix Vulnerability Family in Perplexity Comet and Other Agentic Browsers," Zenity Labs Newsroom, March 3, 2026. https://zenity.io/company-overview/newsroom/company-news/zenity-labs-discloses-pleasefix-perplexedagent-vulnerability

[2] BusinessWire, "Zenity Labs Discloses PleaseFix Vulnerability Family in Perplexity Comet and Other Agentic Browsers," March 3, 2026. https://www.businesswire.com/news/home/20260303909038/en/Zenity-Labs-Discloses-PleaseFix-Vulnerability-Family-in-Perplexity-Comet-and-Other-Agentic-Browsers

[3] OWASP, "LLM01:2025 – Prompt Injection," OWASP Top 10 for LLM Applications 2025. https://genai.owasp.org/llmrisk/llm01-prompt-injection/

[4] Amanda Rousseau / Straiker STAR Labs, "The Silent Exfiltration: Zero-Click Agentic AI Hack That Can Leak Your Google Drive with One Email," Straiker Blog, August 2025. https://www.straiker.ai/blog/the-silent-exfiltration-zero-click-agentic-ai-hack-that-can-leak-your-google-drive-with-one-email

[5] LayerX Security, "LayerX Identifies Vulnerability in New ChatGPT Atlas Browser: Tainted Memories," LayerX Blog, October 2025. https://layerxsecurity.com/blog/layerx-identifies-vulnerability-in-new-chatgpt-atlas-browser/

[6] LayerX Security, "CometJacking: How One Click Can Turn Perplexity's Comet AI Browser Against You," LayerX Blog, October 2025. https://layerxsecurity.com/blog/cometjacking-how-one-click-can-turn-perplexitys-comet-ai-browser-against-you/

[7] Cato Networks, "Cato CTRL: HashJack – First Known Indirect Prompt Injection via URL Fragments," Cato Networks Research, November 2025. https://www.catonetworks.com/blog/cato-ctrl-hashjack-first-known-indirect-prompt-injection/

[8] Zvika Babo / Radware, "Radware Discloses ZombieAgent Technique to Compromise AI Agents," Security Boulevard, January 2026. https://securityboulevard.com/2026/01/radware-discloses-zombieagent-technique-to-compromise-ai-agents/

[9] Noma Security, "GeminiJack: Google Gemini Zero-Click Vulnerability," Noma Security Blog, 2025. https://noma.security/blog/geminijack-google-gemini-zero-click-vulnerability/

[10] Brave Security Team, "Unseeable Prompt Injections in AI Browsers," Brave Blog, 2025. https://brave.com/blog/unseeable-prompt-injections/

[11] Arxiv / Research Team, "The Hidden Dangers of Browsing AI Agents" (CVE-2025-47241), arXiv preprint, 2025. https://arxiv.org/html/2505.13076v1

[12] Cloud Security Alliance, "Securing Autonomous AI Agents: 2025 CSA Agentic Identity Survey," CSA Research, 2025. https://cloudsecurityalliance.org/artifacts/securing-autonomous-ai-agents

[13] Ken Huang et al., "Agentic AI Red Teaming Guide," Cloud Security Alliance, 2025.

[14] Help Net Security, "Agentic Browser Vulnerability: PerplexedBrowser," March 4, 2026. https://www.helpnetsecurity.com/2026/03/04/agentic-browser-vulnerability-perplexedbrowser/

[15] Palo Alto Networks Unit 42, "Indirect Prompt Injection Observed in the Wild," Unit 42 Blog, March 2026. https://unit42.paloaltonetworks.com/ai-agent-prompt-injection/