ShadowPrompt: Zero-Click DOM XSS Enables AI Prompt Injection

Browser Extension Wildcard Origin Trust and Third-Party CAPTCHA Components Create Compound Exploitation Path Against 3 Million Users

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-27

Key Takeaways

- ShadowPrompt, disclosed March 26, 2026, is a zero-click exploit chain combining a wildcard origin allowlist in the Claude Chrome extension with a DOM-based XSS in a third-party CAPTCHA subdomain to inject arbitrary prompts without any user interaction [1].
- The vulnerability affected all 3+ million users of the Claude Chrome extension prior to version 1.0.41 [1]. No user action beyond having the extension installed was required for exploitation.
- Impact included potential theft of Gmail access tokens, Google Drive file access, Claude conversation history exfiltration, and the ability to send emails impersonating victims — all achievable through a single malicious webpage visit [1].
- The attack required no knowledge of the victim's browser state and was invisible to the user. It exploited the structural trust relationship between the extension and the `*.claude.ai` wildcard, illustrating how third-party service integrations on official AI subdomains expand the effective attack surface of AI products.
- Two concurrent arXiv papers — one introducing the TIP (Tree-structured Injection for Payloads) framework for MCP-based prompt injection and one introducing PIDP-Attack for compound RAG poisoning — signal sustained and expanding research focus on AI-specific injection techniques spanning browser interface, tool protocol, and retrieval layers [2][3].
- Security teams should audit all browser extensions that interact with AI platforms for wildcard `postMessage` origin checks, third-party subdomain trust relationships, and `dangerouslySetInnerHTML`-equivalent rendering patterns.

Background

The Claude Chrome extension, published by Anthropic, allows users to interact with the Claude AI assistant directly from the browser, including surfacing context from open web pages and connected Google services. With over 3 million installations as of early 2026 [1], it represents a significant attack surface: a high-privilege browser extension with OAuth access to Google Workspace services running in a trusted execution context alongside every tab the user opens.

Oren Yomtov of Koi Security submitted the ShadowPrompt vulnerability to Anthropic's HackerOne bug bounty program on December 27, 2025 [1]. The disclosure was coordinated over approximately thirteen weeks before public release on March 26, 2026 — a timeline consistent with the coordination overhead of remediating a flaw split across Anthropic's extension codebase and a third-party component maintained by Arkose Labs. The Arkose Labs CAPTCHA component, separately reported to Arkose on February 3, 2026, was patched in full by February 19, 2026 [1]. Anthropic deployed the remediated extension as version 1.0.41, enforcing a strict exact-origin check restricted to `claude.ai` in place of the prior wildcard pattern.

The broader context for this disclosure is one of rapidly accumulating research on AI-specific injection vectors. Prompt injection — the manipulation of an LLM's output by embedding adversarial instructions in its input context — has been ranked LLM01, the highest-ranked vulnerability, in both the 2023 and 2025 editions of the OWASP Top 10 for LLM Applications [4]. ShadowPrompt represents a concrete, weaponized instantiation of indirect prompt injection: an attack path that does not require the victim to type anything into the AI interface, relying instead on compromised environmental data the AI system is trusted to read.

Security Analysis

The Two-Flaw Chain

ShadowPrompt is not a single vulnerability but a compound exploit requiring two independent weaknesses to interact. Neither flaw alone was sufficient to achieve the zero-click AI prompt injection outcome; together they created a zero-click code execution path into the AI interface.

The first flaw resided in the Claude Chrome extension's postMessage event handling. Extensions communicate with web pages through the browser's structured message-passing API, and they must validate the origin of incoming messages to ensure only trusted pages can trigger privileged extension behavior. The Claude extension implemented this validation using a wildcard pattern: any page hosted on a `*.claude.ai` subdomain was granted full messaging trust [1]. This is a recognized antipattern in browser extension security — it trades implementation simplicity for an expanded trust boundary. The intent is to allow legitimate Anthropic subdomains to communicate with the extension, but the wildcard match treats every subdomain — including any that might host third-party code or be vulnerable to injection — as equally trusted as the canonical `claude.ai` application.

The second flaw resided in the CAPTCHA verification component provided by Arkose Labs and served from `a-cdn.claude.ai`, an Arkose-controlled subdomain that matched the extension's wildcard allowlist [1]. This component accepted `postMessage` data from any parent origin without validating `event.origin`, and used the attacker-controlled `stringTable` field from the incoming message to populate React's `dangerouslySetInnerHTML` property without sanitization. The name `dangerouslySetInnerHTML` is React's own explicit signal that direct HTML injection is occurring; its use without sanitization is a textbook DOM XSS pattern.

The resulting attack chain is straightforward to execute and requires no victim interaction. An attacker hosts a page anywhere on the internet. That page embeds the Arkose CAPTCHA component from `a-cdn.claude.ai` in a hidden iframe. The attacker's page sends a crafted `postMessage` to the iframe containing an HTML payload in the `stringTable` field. Because the Arkose component does not validate `event.origin`, it renders the payload as raw HTML. The injected JavaScript, now executing in the context of a trusted `*.claude.ai` subdomain, sends a `postMessage` to the Claude extension — which accepts it without further scrutiny because the origin matches the wildcard [1]. The extension then executes the attacker's embedded prompt against the full scope of its Google Workspace OAuth grants.

Third-Party Subdomain Trust as an Expanded Attack Surface

The architectural lesson of ShadowPrompt extends beyond any single vulnerability. When an AI product integrates third-party services — CAPTCHA providers, analytics platforms, CDN-hosted components — on first-party subdomains, it absorbs the security posture of those third parties into its own trust boundary. The wildcard pattern implies an implicit assumption that any `*.claude.ai` subdomain is under Anthropic's security governance — an assumption ShadowPrompt demonstrates to be unwarranted. In practice, `a-cdn.claude.ai` was Arkose Labs' code, maintained on Arkose's release cycle, with Arkose's security assumptions.

This pattern is not specific to Anthropic or to Claude. Any AI platform that delegates subdomain hosting to SaaS providers while issuing broad origin allowlists to its client-side components faces a structurally similar risk. Prior security research has documented analogous trust boundary violations in MCP server ecosystems, where tool descriptions authored by third parties execute with the same privilege as first-party instructions [5][6][7]. ShadowPrompt demonstrates the same structural problem at the browser layer: the AI interface treats the source of instructions as a proxy for their trustworthiness, and that proxy is forgeable whenever wildcard matching substitutes for cryptographic attestation or strict origin verification.

Concurrent Research: Escalating Injection Techniques Across AI Architectures

The same week as the ShadowPrompt disclosure, two independent research teams published work that together paint a comprehensive picture of the injection threat landscape across AI system architectures.

Shen et al. from Fudan University introduced TIP (Tree-structured Injection for Payloads), a black-box attack framework targeting MCP-enabled agents [2]. TIP frames malicious payload generation as a tree-structured search problem, using a large language model as the attacker to iteratively optimize for natural-sounding payloads that evade four categories of defense: instruction prevention, sandwich prevention, perplexity filtering, and fine-tuned classifier detection. In undefended settings, TIP achieved attack success rates above 95% against models including Llama and Qwen variants. Against all four defenses simultaneously, TIP retained effectiveness above 50% [2]. The attack vector requires only that an MCP-enabled agent query a compromised tool server — no user interaction, mirroring the zero-click nature of ShadowPrompt at a different layer of the stack.

Wang et al. introduced PIDP-Attack, a compound attack on Retrieval-Augmented Generation systems that integrates prompt injection with database poisoning [3]. Prior RAG poisoning attacks, including the well-documented PoisonedRAG technique, required the attacker to know the victim's query in advance — a significant operational constraint. PIDP-Attack eliminates this requirement by appending a lightweight malicious suffix to queries at inference time while simultaneously seeding the retrieval corpus with poisoned passages crafted to appear credible. Evaluated across eight LLMs and three benchmark datasets, using only two poisoned documents, PIDP-Attack achieved average attack success rates of 98.125% — outperforming PoisonedRAG by four to sixteen percentage points [3]. Neither the prompt injection component nor the database poisoning component was effective alone; the compound nature of the attack is essential to its efficacy.

Taken together, ShadowPrompt, TIP, and PIDP-Attack illustrate a consistent structural principle: AI systems that extend trust to ambient environmental inputs — web content, tool responses, retrieval corpora — without cryptographic attestation of their provenance are vulnerable to injection at each layer where that trust relationship is present.

Scope of Impact

The practical impact of ShadowPrompt exploitation would have varied depending on each victim's extension grant scope. For users who had authorized the Claude extension's full Google Workspace integration, exploitation could yield Gmail access tokens enabling read and send access to the victim's email, Google Drive file enumeration and exfiltration, Claude conversation history export, and the ability

to issue authenticated requests to any Google service the extension was granted access to [1]. All of this was achievable through a single page load on a malicious website, with no alerts, permission prompts, or other user-visible indicators.

The affected population of 3+ million Chrome extension users represents a large and high-value target set. Threat intelligence reporting suggests browser extensions with OAuth access to productivity platforms have emerged as a consistent target class for both nation-state actors and financially motivated threat groups, because a single extension compromise can provide persistent, authenticated access to email and document stores that would otherwise require phishing or credential theft to reach.

Recommendations

Immediate Actions

Security teams operating environments where the Claude Chrome extension is deployed should verify that installed extension versions are at or above v1.0.41 [1]. In managed browser deployments, this can be confirmed through Chrome extension inventory tooling or by querying the extension manifest version directly. Organizations with strict change management requirements should evaluate whether AI browser extensions can be deployed from an internal extension store with version locking rather than auto-updating from the Chrome Web Store, accepting that this requires active patch management.

For any browser extension that interacts with AI platforms — whether from Anthropic or any other vendor — security teams should request or conduct a review of the extension's `postMessage` event handlers, focusing specifically on whether origin validation uses exact string matching against an allowlist or wildcard pattern matching. Extensions that rely on `*.provider.ai` or similar wildcard origin checks should be treated as potentially vulnerable to variants of the ShadowPrompt attack pattern until a code review confirms otherwise.

Short-Term Mitigations

Organizations that cannot immediately validate extension versions should consider whether the Claude extension's Google Workspace OAuth grants can be scoped more narrowly pending confirmation. Google Workspace administrators can review and revoke third-party OAuth grants through the Admin Console. Where the extension's full functionality is not required, revoking broad grants in favor of narrower scopes reduces the potential impact of any future extension-layer compromise.

Security teams should add browser extension postMessage handling to their application security review checklist. Specifically, reviews should flag any use of `dangerouslySetInnerHTML`, `innerHTML`, or equivalent DOM injection patterns in code that processes externally sourced `postMessage` data without prior sanitization. This class of DOM XSS is straightforward to detect in static analysis and should be treated as a blocking finding in security code review for any component deployed on a subdomain that receives trust from first-party platform extensions.

For AI platforms deploying third-party components on first-party subdomains, a pragmatic short-term control is to enforce a Content Security Policy on those subdomains that prohibits inline script execution. A strict CSP would not have prevented the `dangerouslySetInnerHTML` rendering in this case — because the React component itself was served from the subdomain rather than injected — but CSP layers provide defense-in-depth and can constrain the capabilities of injected scripts even when initial DOM XSS cannot be prevented.

Strategic Considerations

The ShadowPrompt disclosure, combined with the TIP and PIDP-Attack research, points to a systemic gap in how AI product security is modeled. Threat models for AI browser extensions have historically focused on the extension-to-browser boundary and the extension-to-AI-service boundary. ShadowPrompt demonstrates the necessity of modeling the extension-to-third-party-subdomain boundary as a first-class attack surface, because any third-party code executing on a subdomain that the extension trusts inherits that trust relationship.

AI platform providers should conduct subdomain audits specifically examining which subdomains are operated by third-party service providers, whether those subdomains are within the scope of origin allowlists in any companion extension or application, and whether those third-party providers have undergone security assessments equivalent to those applied to first-party code. The principle of least privilege should apply to subdomain trust just as it applies to OAuth scopes and IAM permissions.

At the architectural level, the convergence of AI capabilities in the browser creates a category of high-value targets that will attract sustained attacker attention. Browser extensions with AI features and productivity integrations combine broad DOM access, OAuth-authenticated API calls, and natural language processing into a single component — an unusually powerful combination that amplifies the impact of any injection vulnerability. Companion desktop applications face analogous risks: separately documented vulnerabilities in Claude Desktop Extensions via MCP tool chaining created remote code execution pathways for affected users [8], reinforcing that the broader AI client ecosystem warrants the

same security scrutiny applied to traditional enterprise applications. Security architectures should account for this asymmetry when defining acceptable residual risk, particularly for extensions and applications deployed at enterprise scale.

For organizations building AI products that integrate with third-party services, the TIP and PIDP-Attack findings suggest that individual defenses tested in isolation — including instruction prevention, perplexity filtering, and classifier-based detection — may achieve only partial effectiveness against compound or adaptive attacks. The research supports a defense-in-depth strategy combining input validation, output monitoring, and privilege scoping rather than reliance on any single control.

CSA Resource Alignment

ShadowPrompt and the concurrent injection research map directly to several dimensions of the CSA's AI security framework work.

The MAESTRO threat modeling framework for agentic AI identifies browser-layer and tool-layer injection as distinct threat categories within its attack surface taxonomy. ShadowPrompt instantiates a MAESTRO-class attack at the browser interface layer, where the AI extension's implicit trust model is the exploitable assumption. The TIP attack on MCP tool responses and the PIDP-Attack on RAG retrieval corpora correspond to MAESTRO's tool abuse and knowledge poisoning threat categories respectively. Prior security research has documented fundamental vulnerabilities in MCP protocol implementations and broad threat scenarios across the tool integration lifecycle [6][7]. Organizations using MAESTRO for agentic AI threat modeling should add `postMessage` origin validation and third-party subdomain trust to their browser interface threat scenarios.

The AI Organizational Responsibilities framework, which defines security accountability boundaries across AI provider, platform, and consumer layers, is directly implicated by the third-party component integration pattern exposed in ShadowPrompt. The framework's guidance on supply chain accountability — specifically, which party bears responsibility for security assessment of components integrated into AI products — applies to the Anthropic-Arkose Labs relationship. Platform providers that integrate third-party services into first-party trust boundaries bear accountability for the security posture of those services, regardless of contractual arrangements.

The CSA's prior work on securing LLM-backed systems, which identifies prompt injection as the highest-ranked vulnerability in the OWASP Top 10 for LLM Applications and describes injection attacks exploiting weaknesses analogous to XSS and SQL injection, provides the foundational analysis within which

ShadowPrompt should be interpreted [5]. ShadowPrompt is a concrete demonstrated instance of the "XSS of the AI agent era" framing — indirect prompt injection delivered via a web vulnerability that enables AI-mediated data exfiltration without any direct model manipulation by the attacker.

The AICM (AI Controls Matrix), as a superset of the Cloud Controls Matrix, offers controls relevant to extension security governance (application security testing requirements), third-party integration assessments (supply chain security controls), and data access governance (OAuth scope management and periodic access reviews). Security teams using AICM as a controls framework should map the ShadowPrompt attack chain to the application security, supply chain, and identity and access management control families when assessing their AI product security posture.

References

[1] Ravie Lakshmanan. "Claude Extension Flaw Enabled Zero-Click XSS Prompt Injection via Any Website." *The Hacker News*, March 26, 2026. (Research by Oren Yomtov, Koi Security.) https://thehackernews.com/2026/03/claude-extension-flaw-enabled-zero.html

[2] Yulin Shen, Xudong Pan, Geng Hong, Min Yang. "Invisible Threats from Model Context Protocol: Generating Stealthy Injection Payload via Tree-based Adaptive Search." arXiv:2603.24203 [cs.CR], submitted March 25, 2026. https://arxiv.org/abs/2603.24203

[3] Haozhen Wang, Haoyue Liu, Jionghao Zhu, Zhichao Wang, Yongxin Guo, Xiaoying Tang. "PIDP-Attack: Combining Prompt Injection with Database Poisoning Attacks on Retrieval-Augmented Generation Systems." arXiv:2603.25164 [cs.CR], submitted March 26, 2026. https://arxiv.org/abs/2603.25164

[4] OWASP Foundation. "OWASP Top 10 for Large Language Model Applications." OWASP, 2025. https://owasp.org/www-project-top-10-for-large-language-model-applications/

[5] Cloud Security Alliance. "Securing LLM-Backed Systems: Essential Authorization Practices." CSA, August 23, 2024. https://cloudsecurityalliance.org/artifacts/securing-llm-backed-systems-essential-authorization-practices

[6] Narek Maloyan, Dmitry Namiot. "Breaking the Protocol: Security Analysis of the Model Context Protocol Specification and Prompt Injection Vulnerabilities in Tool-Integrated LLM Agents." arXiv:2601.17549 [cs.CR], January 2026. https://arxiv.org/abs/2601.17549v1

[7] Xinyi Hou, Yanjie Zhao, Shenao Wang, Haoyu Wang. "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions." arXiv:2503.23278 [cs.CR], March 2025, updated October 2025. https://arxiv.org/abs/2503.23278

[8] Roy Paz, LayerX Security. "Claude Desktop Extensions Exposes Over 10,000 Users to Remote Code Execution Vulnerability." LayerX Security Blog, February 2026. https://layerxsecurity.com/blog/claude-desktop-extensions-rce/

