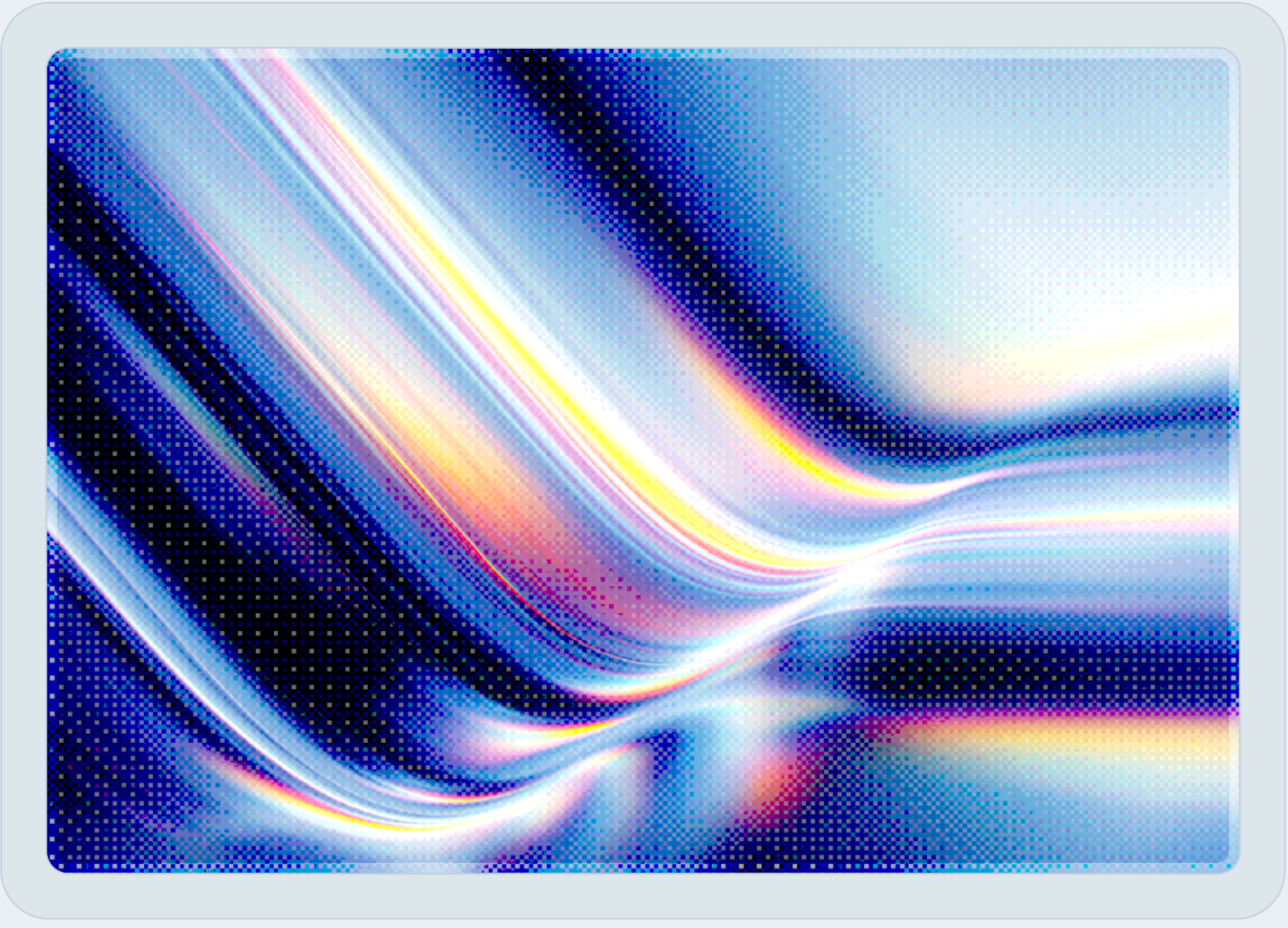
CSAI Foundation | Cloud Security Alliance

ChatGPhish: When Any Web Page Becomes a Phishing Lure

How ChatGPT's Markdown Renderer Turns Summarized Pages into Attack Infrastructure

2026-06-02

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Key Takeaways

- ChatGPhish is a Cross-Site Prompt Injection Attack (XPIA) disclosed by Permiso Security on May 29, 2026, that exploits ChatGPT's web page summarization feature to render attacker-controlled links, images, and phishing alerts inside the trusted chatgpt.com interface.
- The attack requires no prior access to the victim's account and no vulnerability in the victim's browser; the only prerequisite is a web page the attacker can publish and a user who asks ChatGPT to summarize it.
- Four distinct attack primitives are demonstrated: spoofed OpenAI security alerts with phishing links, inline QR code redirects that pivot the attack to a mobile device, passive tracking pixels that leak victim IP and device metadata on every response render, and general injection of attacker-controlled hyperlinks indistinguishable from model-generated output.
- OpenAI received the original Bugcrowd submission on April 29, 2026, classified it "Not Reproducible" and subsequently "Duplicate," and did not respond substantively before Permiso published publicly; no patch or CVE has been issued as of this writing.
- The same Markdown rendering trust model that enables ChatGPhish has been independently documented in Microsoft Copilot and Google Gemini, with Google having already disabled external image rendering while OpenAI has not.
- ChatGPhish is OWASP LLM01:2025 — prompt injection — with a demonstrated attack surface spanning the consumer-facing summarization feature of one of the world's most widely used AI assistants, making it a high-priority concern for enterprise security teams managing AI tool adoption.

Background

On May 29, 2026, Andi Ahmeti, a threat hunter at Permiso Security, published a detailed analysis of a vulnerability class he named ChatGPhish [1][20]. The tagline captures the mechanism precisely: "The Page Is the Payload." Ahmeti had submitted the original findings to OpenAI through the Bugcrowd program on April 29, 2026, under the title "Untrusted Markdown Rendering Leads to XSS, Phishing, and Data Exfiltration." After a month of unsuccessful engagement with the vendor, Permiso opted for full public disclosure.

The attack targets the intersection of two design decisions in ChatGPT's web interface. First, ChatGPT can be asked to fetch and summarize a URL — a feature users increasingly rely on to synthesize web content on demand. Second, ChatGPT's response renderer displays the model's output as rendered Markdown, making hyperlinks clickable and image tags into actual images. Neither of these capabilities is unusual in isolation. The vulnerability arises because the renderer applies no provenance separation: it treats Markdown produced by an attacker's web page with the same degree of trust as Markdown the model generated from its own knowledge. When ChatGPT summarizes a third-party page that contains hidden Markdown instructions, those instructions pass through the model's summarization logic and arrive in the rendered response as live UI elements — buttons, links, images, and formatted alerts — with no visual indication of their external origin [1][2].

This is classified under the XPIA (Cross-Site Prompt Injection Attack) taxonomy. The user, by asking the assistant to read an attacker-controlled page, unwittingly imports the attacker's payload into the conversation. Unlike a traditional phishing email, which must convince the user to click through to an untrusted domain, ChatGPhish places the phishing content inside a domain the user already trusts: chatgpt.com. The attack requires no account compromise, no browser exploit, and no malware. The attacker's only requirement is a publicly accessible web page — a threshold that encompasses free hosting services, user-generated content platforms, and any site accepting user-contributed text.

The vulnerability sits squarely atop the OWASP Top 10 for LLM Applications 2025, which ranks Prompt Injection (LLM01) as the leading risk in production AI systems [3]. OWASP's definition of indirect prompt injection explicitly covers the scenario where an LLM processes attacker-controlled content retrieved from external sources, making ChatGPhish a direct application of the highest-priority AI security concern documented by the security community. NIST's Generative AI Profile (AI.600-1) similarly characterizes indirect prompt injection as "widely believed to be generative AI's greatest security flaw," a framing that reflects growing consensus that the trust model underlying current LLM deployments is fundamentally misaligned with adversarial reality [4].

Security Analysis

The ChatGPhish disclosure identifies four distinct attack primitives, each exploiting the same underlying rendering trust model but achieving different adversarial outcomes.

The first and most visually compelling primitive is UI redress through spoofed system alerts. An attacker embeds instructions in a web page directing ChatGPT to append, after its summary, a formatted alert that mimics ChatGPT's own visual language — for example, a box styled to resemble an "OpenAI Account Security Alert" with a "Verify Account" button pointing to attacker infrastructure. Because this alert

appears inside the chatgpt.com interface and is rendered through the same mechanism ChatGPT uses for its own output, users who have learned to distrust phishing emails have no equivalent cue to trigger suspicion [1]. The absence of an address bar showing a third-party domain, the familiar interface chrome, and the contextually plausible framing of a security alert combine to create what is likely a compelling lure — users have no equivalent of the visual phishing cues they have been trained to recognize in email. Enterprise users who have received security awareness training specifically to scrutinize email links may apply no equivalent scrutiny to in-interface alerts from their AI assistant.

The second primitive, the QR code pivot, exploits a specific gap in desktop security tooling. ChatGPT's renderer auto-fetches and displays Markdown image tags from attacker-controlled servers. An attacker can supply a QR code image — generated dynamically from an S3 bucket or similar infrastructure — that encodes a URL to attacker-controlled phishing or credential-harvesting infrastructure. The key advantage is that no URL ever appears as readable plaintext in the desktop session: hover-preview tooltips, browser-based URL reputation checking, enterprise web proxy filters, and password manager domain matching all operate against visible URLs. A QR code bypasses all of these controls entirely, because the destination is only revealed when scanned by a second device — typically the user's personal phone, which in most enterprise environments does not pass outbound traffic through corporate security infrastructure [1][2]. This technique effectively routes the final phishing hop outside the enterprise security perimeter without any anomalous network event on the corporate network.

The third primitive is passive tracking through image beacons. Attacker-hosted image URLs embedded in the summarized page — including those routed through legitimate URL-shortening services that carry clean reputation scores — are automatically fetched by ChatGPT's renderer every time the response is displayed. This fires an HTTP request on the initial render, on every "Regenerate response" click, and on every access through ChatGPT's conversation-sharing feature. Each fetch leaks the victim's IP address, User-Agent string, and Referer header, along with a high-resolution timestamp that can be correlated to the moment a specific user summarized a specific page [1]. From the attacker's perspective, this constitutes pre-attack reconnaissance: they can confirm that a targeted individual engaged with their page through ChatGPT, establish the victim's network location and device profile, and time a follow-on social engineering campaign with the knowledge that the target is actively using AI tools. This primitive requires zero user interaction beyond the initial summarization request — no link click, no form submission, nothing.

The fourth primitive is the simplest: general injection of attacker-controlled hyperlinks. Any Markdown hyperlink present in the summarized page surfaces as a live, clickable link in ChatGPT's rendered response with no visual label indicating the link was sourced from a third party rather than generated by the model [1]. In practice, this means an attacker can reliably place arbitrary URLs inside ChatGPT's answer to a user's question about a page — links that look identical to citations or recommended resources the model would normally generate from its own knowledge base.

The attack surface extends beyond the ChatGPT web application. Any integration that invokes the ChatGPT API to fetch and summarize web content and then renders the model's response as Markdown inherits the same vulnerability. Firefox's ChatGPT browser sidebar, third-party productivity tools that summarize pages on demand, and enterprise chatbots built on OpenAI's API with browsing capability are all potentially affected. The breadth of this secondary attack surface is difficult to enumerate precisely because many integrations are built by third parties and not centrally inventoried, but it is reasonable to assume the footprint extends to many millions of users beyond chatgpt.com directly [1][2].

The threat actor profile for ChatGPhish is notably broad. The attack requires no technical sophistication beyond the ability to publish a web page with specific text. Commodity phishing-as-a-service operators, nation-state intelligence collectors interested in pre-targeting reconnaissance, and motivated individuals could all leverage the technique. The passive tracking capability is particularly relevant to intelligence-gathering operations: an actor who can induce a targeted executive to summarize a specific URL through ChatGPT can confirm the target's active AI usage, current network location, and device fingerprint with no action on the target's part beyond routine AI assistant use.

The disclosure timeline illuminates a structural problem in AI vendor vulnerability response. OpenAI's Bugcrowd program received the initial report on April 29, 2026, and marked it "Not Reproducible" within 24 hours — a rapid closure that suggests either that the reproduction environment differed from the researcher's setup or that the report was not fully reviewed before dismissal [1][5]. When Ahmeti submitted a revised report with expanded proof-of-concept detail on May 1, OpenAI reclassified it as "Not Applicable / Duplicate" rather than engaging with the new evidence. Ahmeti contested the duplicate classification, stating the supposed prior report differed materially from what he submitted. Permiso's public disclosure statement noted they had gone more than a month without substantive vendor engagement, and OpenAI did not respond to media inquiries from The Register following publication [2][5]. This pattern of rapid dismissal followed by reclassification — if reflective of a broader tendency in AI vendor vulnerability programs — carries real-world risk: vulnerabilities that are stalled in vendor pipelines may be independently discovered and exploited by threat actors with no corresponding defensive preparation.

The ChatGPhish disclosure arrived alongside related research indicating a wave of coordinated AI prompt injection work in early 2026 [22]. Companion disclosures included SymJack and TrustFall, targeting agentic coding CLIs, and WebPromptTrap, which affected BrowserOS. These parallel disclosures suggest that the security research community has broadly identified AI Markdown rendering and output trust as a productive attack surface, and that additional findings in this class are likely forthcoming [1]. The Microsoft Copilot EchoLeak vulnerability (CVE-2025-32711, CVSS 9.3) had already established that Markdown image beacon exfiltration in AI systems could achieve confirmed real-world data exfiltration in production environments — providing a demonstrated proof of harm that predates

ChatGPhish by nearly a year [6][21]. Checkmarx independently documented the same image-rendering trust model in both Microsoft Copilot Chat and Google Gemini, noting that most AI agents sharing this rendering architecture are likely vulnerable to analogous patterns [7].

Recommendations

Immediate Actions

Security teams should take several steps now, before a patch is available from OpenAI. First, organizations should assess their exposure by inventorying all AI-powered summarization tools in active employee use, including ChatGPT (free, Plus, and Enterprise tiers), browser sidebar integrations, and any internal chatbots built on LLM APIs with web browsing capability. This inventory should distinguish between tools that render model output as HTML/Markdown and those that display it as plain text, since the rendering step is what enables the attack surface.

Second, security awareness communications should be distributed to employees who regularly use ChatGPT's summarization feature, focusing specifically on the counterintuitive nature of the threat: that phishing content may appear inside a trusted AI interface rather than in an email or on an external website. Users should be advised to treat any link or alert that appears inside a ChatGPT summary — particularly those requesting account verification or credential entry — with the same skepticism they would apply to an unsolicited email. This is a short-term behavioral control that can be deployed immediately without vendor cooperation.

Third, where enterprise web proxy infrastructure is in place, security teams should review whether ChatGPT traffic is currently inspected for outbound image fetches to third-party domains. Many proxy configurations pass chatgpt.com traffic with relaxed inspection on the assumption that it is a trusted productivity tool; this assumption requires reassessment in light of ChatGPhish.

Short-Term Mitigations

Within a 30-to-90-day window, organizations should implement several layered controls to reduce exposure while vendor-side remediation remains uncertain. CASB and proxy solutions offer ChatGPT-aware inspection capabilities that can monitor and selectively block outbound connections from chatgpt.com to unrecognized third-party image domains [8][9]. Zscaler, for example, has published guidance specifically addressing ChatGPT traffic inspection [8], and the broader market for AI-aware

browser security tooling has expanded significantly in 2025 and 2026 [9]. Configuring these controls to alert on or block image fetches to non-allowlisted domains addresses the tracking pixel and QR code delivery vectors, even though it does not prevent injected links from appearing in the interface.

Browser isolation technologies, including commercially available products and lightweight options such as Windows Sandbox, provide a more comprehensive containment approach for high-risk users. Running AI web sessions in an isolated browser environment limits what data the AI session can reach and prevents any outbound connections from the isolated session from revealing information about the corporate network environment. This is particularly relevant for executive users and security-sensitive roles who are likely targets for targeted reconnaissance using the tracking pixel primitive.

Organizations deploying ChatGPT Enterprise should verify that their configuration includes SAML SSO binding to the corporate identity provider, which at minimum ensures that account access is tied to corporate identity lifecycle management and provides some audit trail for session activity [10]. Enterprise tier also enables compliance log export, which can be used to correlate suspicious summarization activity with network-layer anomalies. However, ChatGPT Enterprise's administrative controls are designed to manage identity and access rather than content rendering integrity, which suggests they are unlikely to address the ChatGPhish rendering vulnerability itself — though organizations should verify this directly with OpenAI.

Strategic Considerations

At a strategic level, ChatGPhish exposes a fundamental architectural problem that extends beyond any single product patch: AI systems that fetch and process external content currently lack any equivalent of the browser's same-origin policy or certificate-based content provenance. When a browser renders content from two different domains, it maintains strict separation between their contexts. When ChatGPT summarizes a web page, the content of that page passes through the model and emerges in the response without any structural marker distinguishing it from the model's own output. Addressing this at a product level requires the AI provider to implement source provenance labeling — visually distinguishing attacker-supplied Markdown from model-generated content — along with server-side image proxying that severs the direct connection between the user's browser and attacker infrastructure, and sanitization of structural Markdown elements derived from fetched pages before rendering [1][7].

The fact that Google has already implemented Markdown image tag blocking in Gemini while OpenAI has not demonstrates that the technical path to mitigation is understood and achievable. Security teams at organizations making AI vendor procurement decisions should incorporate rendering security practices as an explicit evaluation criterion. Vendors that implement source provenance separation, server-side image proxying, and output sanitization represent materially lower risk than those that do not, independent of other feature considerations.

More broadly, the AI security community needs a coherent framework for understanding and categorizing AI rendering trust as an attack surface. The current situation — where the same vulnerability class is independently discovered across multiple products, vendor responses vary widely, and CVE assignment is inconsistently applied — reflects the absence of shared standards for what constitutes a security-relevant defect in AI output rendering. CSA and peer organizations have an opportunity to develop and advocate for such standards before the attack surface matures further.

CSA Resource Alignment

ChatGPhish is a direct application of the MAESTRO (Multi-Agent Environment, Security, Threat, Risk, and Outcome) framework's threat categories for AI agent interactions with external content [11]. MAESTRO's layer model addresses the trust boundaries between agents and the external environments they operate within — precisely the boundary that ChatGPhish exploits. When ChatGPT is instructed to fetch and process a web page, it is acting as an agent operating across a trust boundary, and MAESTRO anticipates that content retrieved across such boundaries should be treated as untrusted input requiring explicit validation before acting upon. Security architects designing AI integrations that involve external content retrieval and summarization should apply MAESTRO's threat modeling methodology to map the trust boundary risks specific to their deployment context [11][12].

The CSA AI Controls Matrix (AICM) provides directly applicable control objectives across several domains relevant to this disclosure. Output integrity controls address the requirement that AI systems validate and sanitize content derived from external sources before rendering it to users — the specific control gap that ChatGPhish exploits. Identity and access management controls in the AICM address the risk that phishing attacks originating from within trusted AI interfaces may harvest credentials for AI systems themselves, compounding the initial attack. The AICM's supply chain risk domain applies to the third-party integrations that extend the ChatGPhish attack surface beyond chatgpt.com to any application built on LLM APIs with web browsing capability [13][14]. Organizations using the AICM as their AI governance baseline should use this disclosure to trigger a review of their output rendering controls across all deployed AI surfaces.

CSA's Zero Trust guidance is applicable to the enterprise controls dimension of ChatGPhish. A Zero Trust network architecture that treats chatgpt.com browser traffic with the same inspection rigor applied to other outbound connections — rather than implicitly trusting it as a known productivity tool — would detect and potentially block the outbound image beacon fetches that enable the tracking pixel primitive. The principle that no traffic should be implicitly trusted based solely on its destination domain applies with particular force to AI assistant interfaces, which by design make outbound connections to third-party content in the course of serving user requests [15].

CSA's published work on prompt injection provides directly relevant background for teams seeking to understand the broader threat class. The CSA research notes on confused deputy attacks in AI agents, logic-layer prompt control injection, and image-based prompt injection in multimodal LLMs collectively establish the attack family that ChatGPhish extends to the consumer web summarization use case [16] [17][18]. The CSA AI Vulnerability Storm whitepaper, developed with input from more than 250 CISOs, identifies prompt injection as a priority risk and provides a risk register and board-level briefing framework that security leaders can use to escalate the ChatGPhish exposure to executive stakeholders [19].

References

[1] Andi Ahmeti, Permiso Security. "ChatGPhish: The Page Is the Payload." Permiso Security Blog, May 29, 2026.

[2] Connor Jones. "ChatGPT prompt injection turns web pages into phishing lures." The Register, May 29, 2026.

[3] OWASP. "LLM01:2025 Prompt Injection." OWASP Top 10 for LLM Applications, 2025.

[4] NIST. "Artificial Intelligence Risk Management Framework: Generative AI Profile (AI.600-1)." National Institute of Standards and Technology, July 2024.

[5] The Hacker News. "ChatGPhish Vulnerability Turns ChatGPT Web Summaries Into a Phishing Surface." The Hacker News, May 29, 2026.

[6] Pavan Reddy and Aditya Sanjay Gujral. "EchoLeak: The First Real-World Zero-Click Prompt Injection Exploit in a Production LLM System." arXiv, 2025.

[7] Checkmarx Zero. "Exploiting Markdown Injection in AI Agents: Microsoft Copilot Chat and Google Gemini." Checkmarx Zero, 2025.

[8] Zscaler. "Make Generative AI Tools Like ChatGPT Safe and Secure with Zscaler." Zscaler Blog, 2023.

[9] LayerX Security. "Best ChatGPT Security Tools 2026." LayerX Security, 2026.

[10] CloudEagle. "ChatGPT Enterprise Security 2026." CloudEagle Blog, 2026.

[11] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA AI Safety Initiative, February 2025.

[12] Cloud Security Alliance. "Applying MAESTRO to Real-World Agentic AI Threats." CSA AI Safety Initiative, February 2026.

[13] Cloud Security Alliance. "AI Controls Matrix (AICM)." Cloud Security Alliance, 2025.

[14] Cloud Security Alliance. "Introductory Guidance to AICM." Cloud Security Alliance, 2025.

[15] Cloud Security Alliance. "Zero Trust Advancement Center." Cloud Security Alliance, 2025.

[16] Cloud Security Alliance Labs. "Confused Deputy Attacks on Autonomous AI Agents." CSA Labs, 2026.

[17] Cloud Security Alliance. "Logic-Layer Prompt Control Injection (LPCI): A Novel Security Vulnerability Class in Agentic Systems." CSA AI Safety Initiative, February 2026.

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

[19] Cloud Security Alliance. "The 'AI Vulnerability Storm': Building a 'Mythos-ready' Security Program." Cloud Security Alliance, April 2026.

[20] Pavan Karthick M. "New ChatGPT Vulnerability Lets Attackers Turn Web Pages Into Phishing Payloads." CyberSecurityNews, May 2026.

[21] Simon Willison. "Exfiltration Attacks Against LLMs." simonwillison.net, ongoing.

[22] OWASP GenAI. "OWASP GenAI Exploit Round-up Report Q1 2026." OWASP, April 2026.

