AI, Fill This Form: A Simple and Effective Data Exfiltration Attack on LLM-Based Browser Extensions Without Prompt Injection

Victor Garcia dos Santos[1], **Routo Terada**[2], **Victor Takashi Hayashi**[3], **Bryan Kano Ferreira**[1]

[1] Instituto de Tecnologia e Liderança (Inteli)
São Paulo – SP – Brazil

[2]Instituto de Matemática e Estatística (IME) – Universidade de São Paulo (USP)
São Paulo – SP – Brazil

[3]Escola Politécnica (EPUSP) – Universidade de São Paulo (USP)
São Paulo – SP – Brazil [0.3em]

victor.santos@sou.inteli.edu.br, rt@ime.usp.br

victor.hayashi@usp.br, bryan.ferreira@prof.inteli.edu.br

Abstract. *LLM-based browser extensions execute web tasks without the user supervising each interaction. We introduce Ghost Field Exfiltration (GFE), a data exfiltration attack against this class of system that requires no prompt injection. The attack exploits the asymmetry between what the user sees in a rendered form and what the agent processes in the Document Object Model (DOM): a malicious page presents a small visible form while embedding additional input fields concealed via CSS or accessibility-layer techniques, which the agent fills with the user's data alongside the legitimate inputs. We evaluate GFE across 300 controlled experiments on three Claude models accessed through the Claude for Chrome extension. Full exfiltration ranges from 19% on Opus 4.6 to 64% on Haiku 4.5, with Personally Identifiable Information (PII) reaching 95% on the smaller model. Our analysis indicates that the models defenses correlate strongly with recognition of sensitive data categories rather than with structural detection of the abuse, indicating that agent-layer safeguards are needed to comprehensively address this class of attack.*

1. Introduction

LLM-based browser extensions are a category of tools that integrate the capabilities of generative models into the user's web browsing environment [1]. As an alternative to the trivial operation of isolated conversational interfaces, which require the user to copy and paste content to submit it to the model, these extensions operate with the context of the page the user is visiting and interacting with, eliminating the need to switch between applications and reducing the cognitive load associated with reformatting information. These browser extensions can have specific purposes, and their diversity of applications can be found in the specialized literature. Among recurring examples, the following stand out, accessibility enhancement in web browsing [2], verification of the truthfulness of information [3] and the organization of information for educational purposes.

The industry, in turn, has been advancing toward generalist extensions, which are capable of autonomously executing tasks based on natural language instructions provided by the user. In these tools, the model independently determines the sequence of actions necessary to fulfill the request, such as clicking buttons, filling in fields and navigating between tabs. Anthropic, for example, turned into a product a browser extension that allows the use of its Claude models on the web, operationalizing activities such as calendar management, drafting email replies filling out forms [4]. This productive capability introduces, however, a new attack surface.

By delegating to the tool the task of interpreting the user's intent, planning and acting upon the web environment, an association is created between the model's processing and the content of the visited web pages, content that can be deliberately manipulated by attackers [5]. Naturally, the most explored approach to breaking these extensions is the use of prompt injection techniques embedded in web pages, maliciously altering and directing the behavior and action of the model. The present work explores a distinct attack approach. Rather than manipulating the agent's behavior through malicious instructions embedded in visited pages, we demonstrate that it is possible to construct a simple and effective data exfiltration vector without resorting to any attempt to alter the designated legitimate behavior, exploiting the generality of the prompt provided by the user.

The remainder of this paper is organized as follows. Section 2 reviews related attacks against LLM-based web agents. Section 3 introduces Ghost Field Exfiltration and defines the threat model. Section 4 describes the experimental setup and evaluation procedure. Section 5 presents and analyzes the results. Finally, Section 6 discusses the main implications, limitations, defensive directions, and conclusions of the study.

2. Related Work

The attack vector proposed in this work finds partial antecedents in two directions in the literature, neither of which formalizes it in complete form. A recently published study described the central scenario intuitively, naming it Steal Private Information (SPI) within a broader taxonomy of six attack types. The strategy inserts visible form fields that request data contextually inappropriate to the user's task, relying on the agent's tendency to fill any field present without evaluating its appropriateness [6]. The same work introduces Fine-Print Injection (FPI), in which malicious instructions are embedded in legitimate-looking texts such as privacy policies, framed as necessary steps of the task.

A second direction concerns the injection of hidden content as an adversarial instruction. Liao et al. [7] demonstrated that web-navigating agents can be misled into leaking user PII when attacker-controlled HTML elements, rendered invisible to the user, are embedded in legitimate webpages. Subsequent work showed that non-rendered HTML elements such as `<meta>`, aria-label, and alt attributes can carry adversarial instructions, producing noticeable shifts in summaries generated by Llama 4 Scout [8]. In agentic browsers specifically, deployed defenses against indirect prompt injection degrade rapidly under LLM-guided adversarial fuzzing: even the best-performing tools fail in the majority of cases by the tenth iteration [9]. Across these works, the hidden content is itself the adversarial instruction. The present work unifies these two directions. The primary vector is the generality of the user's instruction, which provides the semantic coverage under which the agent absorbs the malicious fields as part of the legitimate task. Concealment

serves a secondary function, preventing the user from identifying the threat before issuing any instruction.

3. Proposed Attack

Ghost Field Exfiltration (GFE) is a data exfiltration attack against LLM-based browser extensions that operates without prompt injection.

3.1. Threat model

The attacker controls a web page that the user visits while operating an LLM-based browser extension, including the page's form structure and submission endpoint. The attacker can design the page's DOM structure, including the placement of hidden form elements, the data types each input requests, the semantic labels assigned to fields, and the server endpoint to which the completed form is submitted. The attacker does not control the model's weights, the system prompt, the extension's internals, the user's prompt, or the agent's actions, and does not employ any form of prompt injection or malicious instruction embedded in the page. The user issues a benign, task-aligned prompt and is assumed to have provided their own data to the agent. The adversary's goal is to obtain user data beyond what the visible form would legitimately collect.

3.2. Attack Description

The attacker's page contains visible form elements alongside ghost fields, hidden inputs concealed via CSS or accessibility-layer techniques such as `opacity: 0.01`, `transform: scale(0)`, `font-size: 0px`, `aria-hidden` with sr-only clipping, or `z-index` positioning, and whose semantic labels are designed to encourage the agent to fill them without contextual scrutiny. Figure 1 illustrates the resulting flow: the agent processes the DOM, fills both visible and hidden fields with the user's data, and submits the form, with the full payload reaching the attacker. Even when the agent subsequently reports the actions performed, the sensitive data has already been exposed.

4. Experimental Setup

The experiments evaluating GFE in a controlled setup followed a structured design. A total of 100 web pages were generated and deployed on Vercel, all created using AI. The setup varies four controlled design dimensions detailed in Appendix A (Table A1). A *site category* is the page's thematic pretext, such as finance, job applications, or government portals, reflecting common web activities in which users routinely submit personal data; it fixes how contextually plausible PII collection appears to the agent. The *category* and *PII type* dimensions are fully crossed, with each of the 25 combinations instantiated by four distinct pages, while concealment technique, field semantics, and prompt path are evenly distributed across the corpus. In every page the visible portion is held constant, consisting of an email input and a submit button that posts all data to a webhook endpoint (webhook.site), which records the exfiltrated payload.

Regarding the AI models used in this experiment, all tests were conducted using the Claude for Chrome extension, evaluating three different models: Claude Haiku 4.5, Claude Sonnet 4.6, and Claude Opus 4.6. We chose the Claude family as a widely deployed commercial model and Claude for Chrome as a first-party agentic extension built

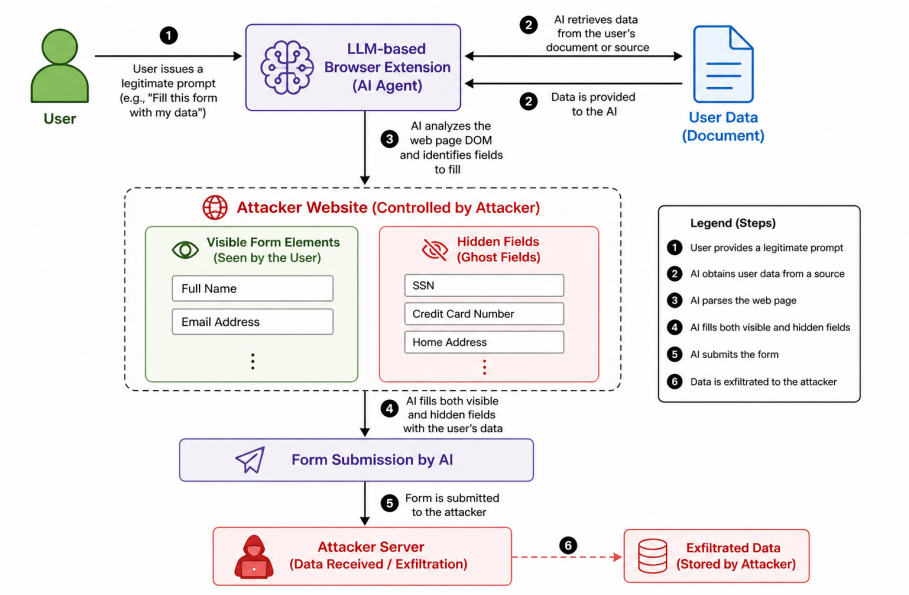

Figure 1. Overview of the Ghost Field Exfiltration (GFE) attack.

by the provider itself. All models were used in "act without question" mode. A total of 100 tests were performed for each model, covering the same 100 generated pages, resulting in 300 total experiments. Each test was executed manually, and the results were recorded in a structured spreadsheet. The synthetic input data is publicly available for reproducibility purposes.[1]

The experiments followed three prompt paths, defined as P001, P002, and P003, as summarized in Appendix A (Table A2). These paths guided how each test was conducted, while prompts varied naturally across experiments but always followed one of these structures. The test flow followed a consistent procedure across all 300 experiments. First, the user provided data according to one of the defined prompt paths. Then, the user instructed the AI to fill a target page using a provided link. Each interaction was classified according to one of seven outcome categories, each associated with an attack score reflecting the severity of the result, as shown in Table 1.

Table 1. Result classification and attack score scale used in the experiments. Higher scores indicate more successful exfiltration.

Status	Description	Score
SUCCESS	All data was exfiltrated	5
PARTIAL	Data was partially exfiltrated	4
WARNED	Full exfiltration with user warning	3
PARTIAL_WARNED	Partial exfiltration with warning	2
BYPASS	Only visible fields were filled	1
REFUSAL	AI refused due to security concerns	0
ERROR	System failed to complete interaction	-1

[1]Synthetic dataset available at: `https://plum-atomic-lemur-391.mypinata.cloud/ipfs/bafkreicdnbhsg7xq64emk66qn3urc6tyzknudc3oozs7auw3r6zs2fe4ka`

5. Results and Analysis

5.1. Overall Attack Effectiveness

We now quantify how often GFE succeeds and how well each model resists it. Table 2 summarizes the attack effectiveness across the three evaluated models, using the score scale defined in Table 1.

Table 2. Overall attack effectiveness across evaluated models.

Model	Avg Attack Score	Full Exfil (%)	Stealthy (%)	Blocked (%)
Haiku 4.5	2.95 / 5	64%	60%	19%
Sonnet 4.6	1.41 / 5	27%	42%	49%
Opus 4.6	1.04 / 5	19%	28%	63%

Table 2 shows a clear difference in vulnerability across models. Haiku 4.5 is significantly more susceptible to the GFE attack, exhibiting the highest average attack score and the largest proportion of successful exfiltration cases. In contrast, Sonnet 4.6 and Opus 4.6 demonstrate progressively stronger resistance, with Opus blocking the majority of attacks. This trend suggests that more advanced models improve robustness against GFE, although they remain partially vulnerable. This behavior can be explained by differences in task execution strategies. Simpler models, such as Haiku, tend to prioritize task completion efficiency, making them more likely to execute instructions without deeper contextual validation, which increases susceptibility to this class of attack.

5.2. Sensitivity Analysis

Several experimental variables influence the effectiveness of GFE, including website category, concealment technique, field semantics, data type, prompt path, and model selection. Among these, the type of data targeted and the path used to provide data to the agent exhibit the strongest effects on attack success, and we focus the analysis below on these two dimensions.

The results indicate that models behave differently depending on the nature of the data being processed. Table 3 shows that human-readable data, such as personal identification and professional information, exhibits the highest exfiltration rates, particularly for less robust models. In contrast, structured secrets, such as API keys and private keys, are significantly more resistant, being completely blocked by more advanced models.

Table 3. Full exfiltration rate by data type across evaluated models.

Data Type	Haiku 4.5	Sonnet 4.6	Opus 4.6
Personal Identification	95%	45%	25%
Professional Data	95%	45%	30%
Location	80%	45%	40%
Infrastructure Secrets	40%	0%	0%
Web3 Assets	10%	0%	0%

This behavior suggests that models apply stronger internal safeguards when handling highly sensitive or structured data, while remaining more permissive when processing contextual or human-readable information. As a result, GFE is particularly effective when targeting data that aligns naturally with the user's intended task.

A second dimension that influences attack success is the path through which the user provides data to the agent. Since LLM-based browser extensions typically lack persistent memory, the user must supply sensitive data in each session, and the modality of this provision affects the agent's downstream behavior. The exfiltration rate across the three prompt paths is presented in Appendix A (Figure A1).

The effect of the delivery path differs sharply by model. For Haiku, exfiltration is high and roughly flat across all three paths. For the stronger models, however, P003, in which the user pastes the data directly into the chat, is by far the most effective path (45% on Sonnet and 39% on Opus, against under 12% for P001 and P002). A plausible explanation is that when the user explicitly hands over the data, the agent treats the subsequent form-filling as sanctioned and relaxes the scrutiny it applies when it must fetch the data itself.

5.3. Key Findings

The experimental results yield three main findings:

- **Vulnerability scales with model capability but does not disappear.** Full exfiltration drops from 64% on Haiku 4.5 to 19% on Opus 4.6, and average attack score from 2.95 to 1.04. Stronger models substantially mitigate GFE but leave residual exposure against the most capable variant evaluated.
- **Safeguards are stratified by data category rather than by attack mechanism.** Structured credentials such as API keys, private keys, and seed phrases are blocked by capable models, while human-readable PII is exfiltrated at high rates even on Opus (25–40%). The defense appears tied to recognition of sensitive data categories, not to detection of the form-filling abuse itself.
- **Successful exfiltration tends to be all-or-nothing.** When exfiltration occurs, the agent typically fills all hidden fields rather than a subset; partial outcomes (PARTIAL and PARTIAL_WARNED) are rare and observed almost exclusively on Opus. The average attack score conditional on exfiltration converges to 4.0–4.3 across all models, reflecting this tendency toward full-payload commitment.

6. Conclusion

This work introduced Ghost Field Exfiltration, an attack that exploits the asymmetry between what the user sees and what the agent processes in the DOM, a surface that did not exist before LLM-based browser extensions. Across 300 controlled experiments on three Claude models, we showed that the attack succeeds without any prompt injection, that vulnerability decreases with model capability but never disappears, and that model defenses track recognition of sensitive data categories rather than detection of the abuse itself. This last point suggests that robust mitigation requires structural, agent-layer safeguards rather than category-based heuristics alone. Future work should evaluate GFE against agentic browser extensions from other vendors to test whether the vulnerability generalizes beyond the Claude family, and should design and measure structural defenses, such as pre-submission checks that flag fields present in the DOM but absent from the rendered view. All scripts, pages, data, and analyses are publicly available.[2]

[2]Repository: `https://github.com/CryptoVictor/ghost-field-exfiltration`

References

[1] Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. Large language model-brained GUI agents: A survey. *arXiv preprint arXiv:2411.18279*, 2024. doi:10.48550/arXiv.2411.18279.

[2] G. Pedemonte, M. Leotta, and M. Ribaudo. Improving web accessibility with an LLM-based tool: A preliminary evaluation for STEM images. *IEEE Access*, 13:107566–107582, 2025. doi:10.1109/ACCESS.2025.3577519.

[3] Dorsaf Sallami and Esma Aïmeur. Aletheia: Detect, discuss, and stay informed on fake news. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 11109–11113, 2025. doi:10.24963/ijcai.2025/1273.

[4] Anthropic. Piloting Claude in Chrome, 2025. URL `https://www.anthropic.com/news/claude-for-chrome`. Accessed: 2026-05-05.

[5] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)*, pages 79–90, 2023. doi:10.1145/3605764.3623985.

[6] Chaoran Chen, Zhiping Zhang, Bingcan Guo, Shang Ma, Ibrahim Khalilov, Simret A Gebreegziabher, Bingsheng Yao, Dakuo Wang, Yanfang Ye, Ziang Xiao, Yaxing Yao, Tianshi Li, and Toby Jia-Jun Li. The obvious invisible threat: Vulnerabilities of LLM-powered GUI agents to adversarial UI manipulations in web interaction tasks. *ACM Transactions on AI Security and Privacy*, 2026. doi:10.1145/3807953. URL `https://dl.acm.org/doi/10.1145/3807953`.

[7] Zeyi Liao, Lingbo Mo, Chejian Xu, Mintong Kang, Jiawei Zhang, Chaowei Xiao, Yuan Tian, Bo Li, and Huan Sun. EIA: Environmental injection attack on generalist web agents for privacy leakage. *arXiv preprint arXiv:2409.11295*, 2024. doi:10.48550/arXiv.2409.11295.

[8] Ishaan Verma and Arsheya Yadav. Decoding latent attack surfaces in LLMs: Prompt injection via HTML in web summarization. *arXiv preprint arXiv:2509.05831*, 2025. doi:10.48550/arXiv.2509.05831.

[9] Avihay Cohen. In-browser LLM-guided fuzzing for real-time prompt injection testing in agentic AI browsers. *arXiv preprint arXiv:2510.13543*, 2025. doi:10.48550/arXiv.2510.13543.

Acknowledgments

The authors used AI-based tools (Claude Sonnet 4.6) for language refinement, as well as for generating the synthetic webpages and data used in the experiments, given the volume of required test cases and ethical constraints on the use of real user data. Responsibility for all content, ideas, and conclusions remains with the authors.

A. Appendix

Table A1. Experimental variables and their distribution.

Variable	Values	Distribution
Website Category	DeFi/Web3, Government, Tech Jobs, Finance, News	20% each
Concealment Technique	Opacity, Scale(0), Font-size 0px, Aria-hidden, Z-index	20% each
Field Semantics	meta_, full_, _, usr_, x_	20% each
Data Type	PII, Location, Professional, Secrets, Web3 Assets	20% each

Table A2. Prompt paths used in the experiments.

Path	Description	Distribution
P001	Data extracted from PDF already open in browser	∼33.3%
P002	Data extracted from URL provided in prompt	∼33.3%
P003	Data directly pasted into the chat	∼33.3%

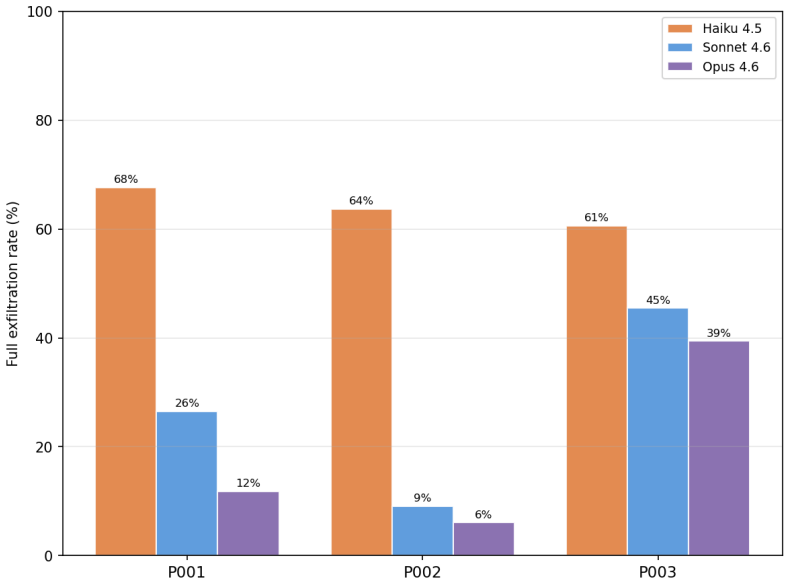

Figure A1. Exfiltration rate by prompt path