From Conventional Web Privacy to Agentic Disclosure: How Tool Schemas May Invite LLM Oversharing

Shahriar Shayesteh **Shomir Wilson**
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA, USA
{sxs7285, shomir}@psu.edu

Abstract

LLM agents increasingly act on behalf of users by selecting tools and constructing API requests to external services. This creates a new privacy risk in agentic systems: disclosure is no longer limited to what users directly enter into a form, but can instead be generated by the agent at runtime. In conventional web settings, disclosure is largely bounded by the user-facing interface, and what is appropriate to share varies across service contexts. In tool-using agents, however, disclosure is generated at runtime when user intent is translated into tool-call arguments for a particular receiving service, making context-sensitive disclosure boundaries harder to preserve. In this position paper, we argue that the runtime tool call is the key unit of privacy analysis in agentic systems. Our contribution is diagnostic rather than behavioral: instead of measuring realized leakage, we analyze interface conditions that may make agent oversharing more plausible. In particular, schemas that expose generic, weakly constrained free-text fields leave part of disclosure under agent discretion. In a case study of 2,344 tool specifications from the OpenAI GPT ecosystem, we find that 36.9% expose at least one such channel, creating conditions for within-context over-disclosure, cross-context leakage, and what we call contextual flattening. We conclude by outlining a research agenda for NLP that moves beyond output-only evaluation toward argument-level analysis of what tool schemas allow agents to send to third-party services.

1 Introduction

Large language model (LLM) agents increasingly act on behalf of users by planning tasks and invoking external tools. This shifts privacy risk to runtime disclosure: rather than focusing only on what web services collect through users' direct interactions with websites, we must also examine what agents disclose when they translate user requests into tool-call arguments for downstream services.

Existing work has shown that language models and tool-using agents can reveal sensitive or task-irrelevant information through outputs, prompts, and runtime behavior (Mireshghallah et al., 2023; Li et al., 2025; Roh et al., 2026; Zharmagambetov et al., 2025). Related work has also shown that user intent can sometimes be reconstructed from metadata alone (Yao et al., 2025). Those results are important, but they primarily evaluate whether agents leak or what can be inferred from agent-generated traces. Our contribution is diagnostic rather than behavioral: we ask how tool schemas create interface conditions that make agent over-disclosure more plausible, prior to any runtime measurement.

Agents access third-party services through tools whose schemas specify the interface and the parameters to fill. When those parameters are generic free-text fields such as `query`, `notes`, or `context`, part of the disclosure decision is left to the agent at runtime. This creates a design-side privacy risk that agent-centered behavioral evaluation alone cannot detect.

What counts as appropriate disclosure also varies across receiving service contexts. Because directly modeling all such contexts is difficult, we use sector as a practical proxy for context. We use contextual flattening to describe failures in which service-specific disclosure boundaries are not preserved at runtime, so that agents may treat distinct service settings as if the same user background were appropriate everywhere.

To ground this argument, we analyze 2,344 real-world tool specifications from the OpenAI GPT ecosystem and find that 36.9% expose at least one weakly constrained free-text parameter. Our claim is not that such schemas by themselves cause leakage, but that they create interface conditions that leave disclosure partly under agent discretion, mak-

Proceedings of the Seventh Workshop on Privacy in Natural Language Processing, pages 1–6

ing oversharing more plausible and harder to bound. We therefore argue that privacy in tool-using agents should be studied at the runtime tool call, not only in terms of task success but also in terms of what is disclosed to external services.

2 From Conventional Web Privacy to Agentic Disclosure

In the conventional web, disclosure is largely bounded by the user-facing interface: users provide information through forms, clicks, and browsing activity, and services receive that directly supplied input. Privacy research in that setting asks what services receive and what they do with it. Prior large-scale measurement work on web forms suggests that personal-information collection practices vary systematically across website categories and form functionalities (Cui et al., 2024). This variation is also normatively meaningful because many sectors operate under domain-specific legal frameworks, such as HIPAA, GLBA, and FERPA. It therefore provides a useful baseline for evaluating disclosure in agentic systems.

Agentic systems change the information-flow structure of the conventional web. The user delegates a natural-language goal to an LLM agent, and the agent interprets that goal, selects tools, and constructs the arguments transmitted in tool calls. This is a language generation problem: the agent draws on the full conversational context—current query, prior turns, stored or inferred attributes—to generate tool-call content. The user remains the source of the request, but the immediate sender to downstream services is now the agent acting as the user's delegate. What reaches an external service may therefore exceed what the user directly addressed to it.

This shift changes the relevant unit of privacy analysis. In conventional settings, auditing what a service receives is often close to auditing what the user directly submitted. In agentic settings, that equivalence breaks down: services may receive information assembled across turns or inferred from conversational context. The key privacy event is therefore the runtime tool call, where agent generation and interface constraints jointly determine what gets disclosed. The next question, then, is how to evaluate whether agent-generated tool calls preserve or violate context-specific disclosure boundaries.

3 The Threat: Contextual Flattening

We frame the privacy risk in tool-using agents through Contextual Integrity (CI), which treats privacy as appropriate information flow within a given context (Nissenbaum, 2004; Barth et al., 2006). In CI terms, the key questions are who is sending the information, what information is being shared, and under what conditions that sharing is appropriate. In the conventional web, these flows are relatively bounded by the user-facing interface.

In agentic systems, all three dimensions of the information flow can shift. The immediate sender is no longer only the user, but the agent acting on the user's behalf; the information transmitted is no longer limited to directly entered form content, but may be assembled from conversational context; and the transmission principle becomes less fixed when tool schemas expose generic open-text fields. Structured fields constrain disclosure through their type and label, whereas free-text fields leave that boundary partly to the agent's runtime inference. The result is a design-level privacy risk: the interface no longer sharply encodes what belongs in the disclosure and what should be left out.

This creates two privacy failures: **within-context over-disclosure**, where an agent shares more information than a receiving service needs within the same task context; and **cross-context leakage**, where information introduced in one task is unnecessarily carried into another service context where it does not belong. For example, a user might tell an agent, "I have chest tightness. Find urgent care, then a ride there." To complete the first task, the agent may draw on the current request together with prior conversational context or stored memory, such as the user's ZIP code, home address, or medication history. The agent calls the urgent-care locator with `specialty=cardiology` and the user's ZIP code—both appropriate inputs to the locator's function. If the locator schema also exposes a generic free-text field such as `notes`, however, the agent may additionally include unnecessary health details such as a stored medication like sertraline. That is *within-context over-disclosure*: the receiving service gets more information than is needed for the immediate task, violating data minimization. The locator does not need the medication to return nearby cardiology facilities.

If the agent then carries the cardiology context into a later ride-booking request—for example, by passing `ride_notes=going to a`

`cardiology appointment`—that is *cross-context leakage*. The cardiology attribute was appropriate at the locator, whose function requires specialty as input, but is not appropriate at the ride-booking service, whose function requires trip logistics such as origin and destination, not medical specialty. The same attribute, appropriate in one service context, now flows into another with a different purpose, violating purpose limitation. In both cases, the consequence is not only excess disclosure, but an expansion of which downstream services can retain and repurpose sensitive information beyond its original context.

We refer to the broader pattern underlying these failures as contextual flattening: distinctions among service-specific disclosure boundaries are no longer preserved when tool interfaces leave too much of the information flow to runtime generation. Put differently, the agent treats distinct service contexts as if they could all receive the same user background. As a result, privacy violations are no longer just a matter of excessive disclosure, but of eroding the context-specific norms that ordinarily govern appropriate information flow.

4 A Schema Analysis of Agentic Leakage

Contextual flattening is not only a failure of agent reasoning (Mireshghallah et al., 2023); it is also shaped by tool design. Tool schemas define the action space within which agent reasoning operates: they specify which fields are available, how narrowly those fields are typed, and whether the agent has a residual channel for adding conversational context. When a schema exposes only constrained fields, such as a date, postcode, or identifier slot, there is relatively little room to add unrelated user context. By contrast, generic free-text parameters with broad descriptions—such as `query`, `notes`, or `additional_context`—leave part of the disclosure decision to the agent. Our case study therefore examines the interface-level conditions under which oversharing becomes plausible.

We analyze 2,344 unique third-party Action/API specifications attached to GPTs in the OpenAI GPT ecosystem, derived from the large-scale crawl by Wu et al. (2025), who collected over 119,000 GPTs and 4,592 Actions in May 2024.[1] To operationalize when disclosure is left partly under agent discretion, we use a simple structural heuristic over the flattened tool catalog: for each parameter, we take the base name after any nesting prefix and flag it when it is an unconstrained string field whose name matches a case-insensitive regex over generic open-text identifiers.[2] We constructed this identifier list by starting from common OpenAPI-style parameter names that invite natural-language input (e.g., `query`, `message`, `notes`, `prompt`) and expanding it through inspection of the flattened catalog to include semantically similar variants. We treat a string field as unconstrained when the schema does not limit its possible values through an enum, format constraint, or other schema-level validation rule—in practice, fields that accept arbitrary natural-language text rather than a fixed category, identifier, date, or similarly bounded value.

Schema Type	Count	% of All Tools
Structured-only	1,480	63.1%
Free-text only	204	8.7%
Mixed (free-text + structured)	660	28.2%

Table 1: Schema distribution across 2,344 unique third-party Action/API specifications attached to GPTs in the Wu et al. (2025) corpus. In total, 864 tools (36.9%) expose at least one generic, weakly bounded free-text parameter under our heuristic.

This heuristic is structural rather than behavioral: it identifies schemas that leave a generic open-text channel weakly bounded at the interface level, not realized leakage by any particular agent or model.[3] Because semantically sensitive but domain-specific parameters (e.g., `patient_notes`, `account_summary`, `legal_context`) are not captured by generic name matching, the 36.9% figure should be read as a conservative structural indicator of how often tool schemas expose generic open-text channels for agent-generated disclosure, not as an exhaustive measure of risky schemas, a vulnerability rate, or an estimate of how often leakage will occur in practice.

Table 1 summarizes the distribution. A substantial minority of tools—**36.9% (n=864)**—expose at

[1]The code is available at https://github.com/shahriarshayesteh/agentic-disclosure.git

[2]The identifier list included: `q`, `query`, `search`, `message`, `content`, `text`, `description`, `notes`, `context`, `instructions`, `input`, `prompt`, `body`, `comment`, `details`, `user_query`, `user_message`, `user_input`, `user_request`, `request`, `question`, `info`, `information`, `reason`, `purpose`, `subject`, and `task`.

[3]The heuristic does not capture domain-specific risky parameters or distinguish benign open-text use from harmful disclosure; see Limitations for full discussion.

least one generic, weakly bounded free-text parameter, either as free-text-only schemas (8.7%) or as mixed schemas that combine structured slots with an open-text field (28.2%). Not all free-text parameters are equally privacy-relevant—for example, some search-style query fields may be functionally necessary—but the mixed cases are especially concerning because they partially constrain the task while still leaving one residual channel through which an agent can add unnecessary user context. These schemas can support correct task completion while still creating an opening for within-context over-disclosure: the agent fills the required fields appropriately but includes more information than the task requires.

The same design also makes cross-context leakage plausible. Common open-text fields such as `query`, `q`, `prompt`, `context`, and `notes` are generic rather than context-specific: their names and descriptions indicate that the tool expects user-provided text, but not what should be excluded. In multi-step workflows where an agent retains earlier conversational state, this absence of schema-level constraints creates a channel through which information from one task can carry over into another tool call.

A small number of tools go further by explicitly encouraging richer disclosure. One notable example in our case study is Zapier AI Actions for GPT, used by 231 GPTs. Its `instructions` parameter carries the description: *"Provide as much detail as possible, even if other fields are present."* We do not treat this example as representative of the entire ecosystem. Rather, we use it to illustrate that some schemas do more than permit broad disclosure: they actively nudge agents toward richer input than the task requires.

This analysis is intentionally coarse. It identifies schema conditions that make privacy failure more plausible, not realized leakage rates for any particular agent, prompt, or model. We therefore do not claim that schema analysis demonstrates behavioral leakage. Our claim is that current tool ecosystems contain interface conditions that make both failure modes plausible. The problem is not only what agents choose to send—it is also what tool interfaces leave weakly constrained, or explicitly encourage agents to send.

5 A Research Agenda for NLP

If privacy risk in agentic systems is shaped at runtime, evaluation cannot stop at task success or final responses. It must examine the arguments agents send in tool calls: which information is included, which service receives it, and whether that disclosure is justified by the task context. In Contextual Integrity terms, this treats each tool call as an information flow between the agent and the receiving service. We highlight three directions for evaluating these disclosures.

Cross-context benchmarks. Existing agent benchmarks primarily measure task completion, output quality, or general safety, but they do not test whether information from one task reappears in later tool calls serving a different purpose. This matters because cross-context leakage can occur in intermediate channels, not only in final outputs. Recent work has begun to highlight such leakage risks in agentic systems (Yagoubi et al., 2026). A concrete next step is to build multi-step evaluations that measure the *carryover rate* of prior-context information into later tool calls and distinguish appropriate carryover from contextually inappropriate reuse. Because inappropriate disclosure is often semantic rather than lexical, evaluating it will require methods beyond string overlap, such as LLM-based judges that assess appropriateness under contextual-integrity norms.

Argument-level disclosure metrics. Current evaluation often treats tool-call arguments as a means to an end. Privacy evaluation, however, should treat them as the object of analysis: what information is transmitted, whether it was necessary for task completion, and how excess disclosure varies across schema types such as structured-only, free-text-only, and mixed tools. A useful next step is to define metrics such as the *irrelevant-information rate* per call and compare them across schema conditions, extending recent privacy evaluations grounded in data minimization (Zharmagambetov et al., 2025).

Structural leakage conditions. Our analysis also points to the need for controlled comparisons of the structural factors most likely to shape disclosure, especially schema type and tool-description permissiveness, while accounting for moderators such as memory settings, prompting strategy, and model family. One useful goal is to estimate how much more an agent discloses under free-text or

mixed schemas compared to structured-only interfaces, and whether that excess is more sensitive. Such estimates would provide an empirical basis for safer tool-interface design.

6 Conclusion

The shift from the conventional web to agentic systems changes where privacy risk must be evaluated. When agents construct API requests on behalf of users, the critical observation point is no longer only the privacy policy or the data collection form, but the runtime tool call through which user information is transmitted. We have argued that privacy evaluation in such systems must examine not only whether agents complete tasks, but whether they preserve context-sensitive disclosure boundaries. Our schema analysis shows that this concern is grounded in current tool ecosystems: a substantial minority of real tool schemas expose generic, weakly bounded free-text channels that leave part of disclosure under agent discretion. The privacy problem, in short, is not only about what agents choose to leak—it is also about what tool interfaces invite them to send. For NLP, the key next step is to study what agents disclose across contexts and under what interface conditions.

Limitations

This paper has four main limitations. First, industrial sector is only a practical proxy for the richer notion of "context" in Contextual Integrity and does not capture the full complexity of privacy expectations, which vary across jurisdictions, relationships, and social settings. Sector is useful for large-scale structural analysis, but finer-grained operationalizations of context would better reflect the norms CI is intended to capture. Second, our schema analysis is static: we classify tool interfaces from their specifications without observing agent behavior at runtime. Schema design creates structural conditions for over-disclosure, but whether agents exploit those conditions depends on model family, system prompt, memory configuration, and task context. Third, our heuristic for disclosure-prone free-text fields is intentionally coarse. It may miss risky domain-specific parameters and does not by itself distinguish benign open-text use from harmful disclosure. Our findings therefore establish a necessary precondition, not a sufficient cause, of the privacy failures we describe. Fourth, our case study reflects one platform ecosystem and one collection snapshot—the OpenAI GPT ecosystem as captured by Wu et al. (2025). Broader analyses across agent platforms, schema standards, and time periods are needed to assess generality.

Ethical Considerations

Our analysis uses the publicly released dataset of Wu et al. (2025), collected via automated crawling of publicly indexed GPT stores. It does not contain personally identifiable information, and we do not execute or interact with any tools. The findings identify structural privacy risks in deployed agentic systems. We believe transparency about these risks is necessary for the research community, tool developers, and policymakers to design more privacy-protective systems.

Acknowledgments

This manuscript is based upon work supported by the National Science Foundation under Grant Nos. 2105736 and 2237574.

References

Adam Barth, Anupam Datta, John C. Mitchell, and Helen Nissenbaum. 2006. Privacy and contextual integrity: Framework and applications. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, SP '06, page 184–198, USA. IEEE Computer Society.

Hao Cui, Rahmadi Trimananda, and Athina Markopoulou. 2024. Understanding privacy norms through web forms. *Proc. Priv. Enhancing Technol.*, 2025:5–22.

Haoran Li, Wenbin Hu, Huihao Jing, Yulin Chen, Qi Hu, Sirui Han, Tianshu Chu, Peizhao Hu, and Yangqiu Song. 2025. PrivaCI-bench: Evaluating privacy with contextual integrity and legal compliance. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10544–10559, Vienna, Austria. Association for Computational Linguistics.

Niloofar Mireshghallah, Hyunwoo Kim, Xuhui Zhou, Yulia Tsvetkov, Maarten Sap, Reza Shokri, and Yejin Choi. 2023. Can LLMs keep a secret? testing privacy implications of language models via contextual integrity theory. *arXiv preprint arXiv:2310.17884*.

Helen Nissenbaum. 2004. Privacy as contextual integrity. *Washington Law Review*, 79:119–157.

Jaechul Roh, Eugene Bagdasarian, Hamed Haddadi, and Ali Shahin Shamsabadi. 2026. Spillage: Agentic oversharing on the web. *arXiv preprint arXiv:2602.13516*.

Yuhao Wu, Evin Jaff, Ke Yang, Ning Zhang, and Umar Iqbal. 2025. An in-depth investigation of data collection in LLM app ecosystems. In *Proceedings of the 2025 ACM Internet Measurement Conference*, IMC '25, page 150–170, New York, NY, USA. Association for Computing Machinery.

Faouzi El Yagoubi, Ranwa Al Mallah, and Godwin Badu-Marfo. 2026. Agentleak: A full-stack benchmark for privacy leakage in multi-agent llm systems. *arXiv preprint arXiv:2602.11510*.

Yunhao Yao, Zhiqiang Wang, Haoran Cheng, Yihang Cheng, Haohua Du, and Xiang-Yang Li. 2025. Intentminer: Intent inversion attack via tool call analysis in the model context protocol. *arXiv preprint arXiv:2512.14166*.

Arman Zharmagambetov, Chuan Guo, Ivan Evtimov, Maya Pavlova, Ruslan Salakhutdinov, and Kamalika Chaudhuri. 2025. AgentDAM: Privacy leakage evaluation for autonomous web agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.