AudAgent: Automated Auditing of Privacy Policy Compliance in AI Agents

Ye Zheng
Rochester Institute of Technology
Rochester, USA

Yidan Hu
Rochester Institute of Technology
Rochester, USA

Abstract

AI agents can autonomously perform tasks and, often without explicit user consent, collect or disclose users' sensitive local data, which raises serious privacy concerns. Although AI agents' privacy policies may describe their intended data practices, there remains limited transparency and accountability about whether runtime behavior matches those policies. To close this gap, we introduce AudAgent, a visual tool that continuously monitors AI agents' data practices in real time and guards compliance with stated privacy policies.

AudAgent consists of four components for automated privacy auditing of AI agents. (i) Policy parsing: an ensemble of LLMs translates natural-language privacy policies into a structured privacy-policy model, where cross-LLM voting guarantees confidence of the parsing results. (ii) Runtime annotation: a lightweight Presidio-based analyzer detects sensitive data and annotates how the data is used based on the context of the AI agent's operations and the privacy-policy model. (iii) Compliance auditing: ontology alignment and automata-based evaluation connect the policy model with runtime annotations, enabling on-the-fly compliance checks between the natural-language policy and observed unordered data practices of AI agents. (iv) User interface: a platform-independent implementation visualizes the real-time execution trace of AI agents along with potential privacy risks detected during auditing, providing user-friendly transparency and accountability.

In addition to common formatted privacy policies, AudAgent also supports user-defined policies for fine-grained control and customization. We evaluate AudAgent on AI agents built upon mainstream agent-building frameworks such as AutoGen, experiments show that AudAgent effectively identifies potential privacy policy violations in real time.

CCS Concepts

• **Security and privacy** → **Privacy protections**; *Information flow control.*

Keywords

privacy policy compliance, AI agents, data auditing

ACM Reference Format:
Ye Zheng and Yidan Hu. 2018. AudAgent: Automated Auditing of Privacy Policy Compliance in AI Agents. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXX.XXXXXX

1 Introduction

AI agents, such as intelligent assistants and workflow automation tools, are increasingly capable of autonomously performing a widening range of personal and professional tasks on behalf of users. To function, these agents often collect, process, and sometimes disclose users' local data [11–13, 16], which can include sensitive information [30], frequently without explicit user consent. Despite their data practices being outlined in the platform's privacy policies, users often lack clear visibility into an AI agent's actual practice of the sensitive data and whether they comply with those privacy policies. This opacity is further compounded by complex third-party integrations, which may involve external APIs and services that can lead to unintended data collection and disclosure. Such challenges highlight the need for effective auditing tools that empower users to verify and ensure that their AI agents adhere to the platform's stated privacy policies or user-defined privacy preferences.

Related work. This paper focuses on users' data privacy in the context of AI agents, relating to two broader aspects. Attacks on AI agents' privacy and security, including techniques for extracting sensitive data via malicious prompts [31, 33, 34, 41] and compromised tools [27, 42, 50]. Defenses, including sandbox-based behavioral testing [43, 54] and rule-based control [19] to constrain agent actions. Technically, this paper also relates to privacy policy analysis [23, 24, 51, 56] and compliance auditing of applications [21, 22, 57]. Section 5 provides detailed discussion and comparison. In summary, prior work on defending users' privacy in AI agents has largely focused on techniques for detecting vulnerabilities and malicious behaviors, while compliance-auditing research has concentrated on mobile applications, web services, and similar domains. For ordinary users of AI agents, however, there is no effective real-time method that lets them easily determine an agent's sensitive data practices or whether those practices comply with the agent's stated privacy policies. This motivates the following question: *How can users easily audit their AI agents' data practices against the privacy policies they care about?*

Challenges. Enabling users' auditing of AI agents' data practices raises three key challenges.

- *(Policy formalization)* Privacy policies are typically written in natural language and must be translated into precise, machine-checkable formats.

- *(Limited visibility)* AI agents frequently interact with local and third-party services, such as APIs and file systems, making it difficult to detect sensitive data practices from a single vantage point.
- *(User effort and efficiency)* Auditing should be automated to avoid expert involvement and should be real-time without significant overhead.
- *(User-friendly visualization)* The tool should offer an intuitive, platform-agnostic visualization, making auditing easily accessible and facilitating broad adoption.

This paper. We propose AudAgent, an automated, user-facing data-auditing tool that tackles these challenges. (i) AudAgent leverages the regular structure of privacy-policy documents to guide LLMs in extracting a formal, machine-checkable policy model that captures data categories and concrete constraints on collection, processing, disclosure, and retention. To improve extraction robustness and reduce errors, AudAgent applies cross-LLM voting for confidence aggregation to reconcile the policy elements. (ii) AudAgent employs a lightweight, Presidio-based [37] analyzer to monitor the agent's inbound and outbound data in real time, combining runtime operation context with the extracted policy model to identify and annotate sensitive data usage. (iii) AudAgent uses ontology alignment and automata-based evaluation to connect the policy model with runtime annotations: the ontology alignment reconciles semantic variations and hierarchical inclusions between policy categories and detected entities, and the automata-based evaluation compiles policy constraints into efficient, on-the-fly checks that can run in parallel. (iv) AudAgent is implemented as a modular, platform-independent tool that visualizes the real-time execution trace of AI agents along with potential privacy risks detected during auditing via a browser interface, providing user-friendly transparency and accountability.

AudAgent is fully automated: users only need to provide the relevant privacy-policy documents. It continuously monitors agent behavior in real time and alerts users to potential privacy risks. Figure 1 illustrates the overall workflow of AudAgent.

Benefits to users and platforms. For users, AudAgent provides real-time privacy auditing of their AI agents and configurable controls to block or warn about specific data-leakage behaviors (for example, via keyword filters or category-based rules in user-defined policies). It also displays alerts to help users manage privacy risks with minimal effort. For AI-agent platforms, AudAgent functions as a diagnostics and accountability tool: it helps evaluate actual agent behavior against stated privacy policies, find discrepancies, and guide adjustments to data-handling practices or policy wording to improve compliance and user trust.

Deployment examples. We implement AudAgent as a modular tool that can be integrated with multiple AI agent building frameworks (orchestration platforms). Section 4 presents a running example using Microsoft's AutoGen [36], a widely used open-source framework for building LLM-based agents with tool support. AudAgent compensates such frameworks with real-time privacy auditing and visualization, addressing their lack of built-in auditing capabilities and improving transparency and user trust for practical deployments.

Specifically, our contributions include the following aspects:

- To our knowledge, AudAgent is the first tool that enables automated auditing of AI agents' data practices against explicit privacy-policy documents. It provides end users visual transparency to verify that agent behavior matches stated privacy expectations.
- AudAgent addresses key challenges in policy formalization, limited visibility, efficient auditing, and user-friendly visualization. It bridges the technical gap between natural-language privacy policies and runtime data practices of AI agents, facilitating practical deployments of AI agents by enhancing user trust and transparency.
- We implement AudAgent as a modular tool that can be agnostically integrated with various AI agent building frameworks. The evaluation on such integrations demonstrates the effectiveness of AudAgent in real-world scenarios.

The main parts of this paper are organized as follows. Section 2 formalizes the problem and provides preliminaries. Section 3 presents the design of AudAgent, detailing its architecture and core components. Section 4 describes application examples and integrations with existing agent platforms.

2 Preliminaries

This section specifies our threat model, reviews background on privacy policies, AI agents, privacy policy compliance, and real-time visualization. It also highlights the key challenges that AudAgent aims to address.

2.1 Assumptions and Threat Model

Given the varying definitions of AI agents and the practical limits of observing data flows, we make the following scoping assumptions:

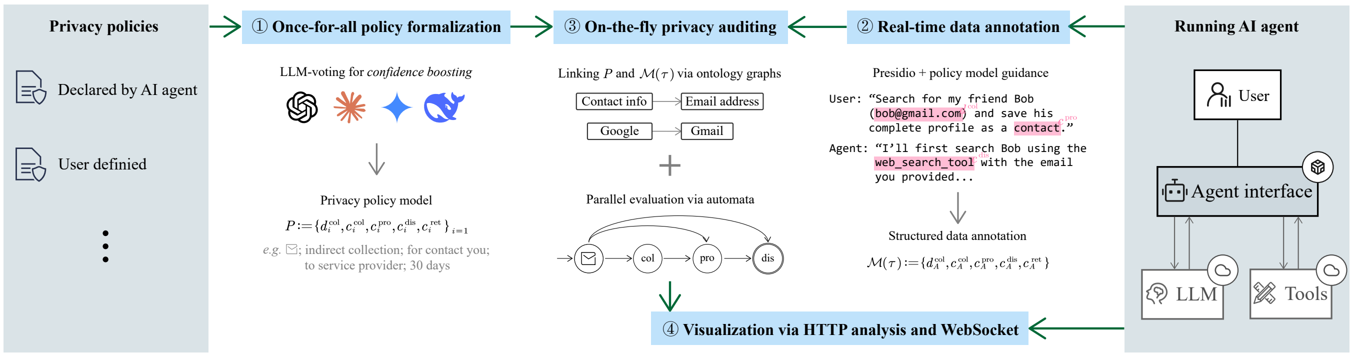
- An AI agent consists of an LLM and a set of tools: the LLM provides core decision-making, while tools interact with context or external systems to extend the agent's capabilities.
- The agent's control logic (orchestrator) executes locally on the user's device, whereas the LLM and third-party tools may be hosted as cloud services.

These assumptions reflect common agent architectures [7, 36], where a local orchestrator coordinates on-device data and cloud-based components.

Threat model. We assume external/cloud services used by the agent (including LLMs and third-party tools) are only partially trusted. Such services may, intentionally or inadvertently, misuse or disclose sensitive user data beyond the limits stated in their privacy policies. For example, an agent might forward sensitive information to non-essential third-party tools or otherwise behave in ways that exceed its declared data practices. Our goal is to detect these privacy risks and potential policy violations in the agent's data flows.

2.2 Privacy Policies

Privacy policies are legal documents that explain how an organization collects, uses, and discloses personal information from users. According to the General Data Protection Regulation (GDPR) [4], any organization that processes personal data of residents must


Privacy policies
Declared by AI agent
User definied
① Once-for-all policy formalization
LLM-voting for *confidence boosting*
Privacy policy model
$P := \{d_i^{col}, c_i^{col}, c_i^{pro}, c_i^{dis}, c_i^{ret}\}_{i=1}$
e.g. ✉; indirect collection; for contact you; to service provider; 30 days
③ On-the-fly privacy auditing
Linking P and $\mathcal{M}(\tau)$ via ontology graphs
Contact info
Email address
Google
Gmail
Parallel evaluation via automata
col
pro
dis
② Real-time data annotation
Presidio + policy model guidance
Structured data annotation
$\mathcal{M}(\tau) := \{d_A^{col}, c_A^{col}, c_A^{pro}, c_A^{dis}, c_A^{ret}\}$
d^{col}
c^{pro}
c^{dis}
④ Visualization via HTTP analysis and WebSocket
Running AI agent
User
Agent interface
LLM
Tools

```
User: “Search for my friend Bob
      (bob@gmail.com) and save his
      complete profile as a contact.”
Agent: “I’ll first search Bob using the
       web_search_tool with the email
       you provided...
```

Figure 1: Overview of AudAgent, a tool for auditing an AI agent's data practices against specified privacy policies. It comprises four components: (1) Policy formalization (Section 3.1), which performs a one-time extraction of privacy-policy documents into a formal model; (2) Runtime Annotation (Section 3.2), which continuously monitors and annotates the agent's data practices; (3) Compliance Auditing (Section 3.3), which performs on-the-fly checks comparing the policy model with runtime annotations; and (4) User Interface (Section 3.4), which visualizes the agent's real-time execution trace and highlights privacy risks detected during auditing.

provide clear information about how that data is used.[*] Privacy policies typically follow a standard structure [3, 28, 49], formalized by the following components.

Definition 1 (Privacy Policy). *A privacy policy P is a data model defined by a set of tuples $P := \{(d_i^{col}, c_i^{col}, c_i^{pro}, c_i^{dis}, c_i^{ret})\}_{i=1}$,*[†] *where*

- *$d_i^{col} \in D^{col}$ is a data type the first party collected from users together with the conditions c_i^{col} under which collection occurs;*
- *$c_i^{pro} \in C^{pro}$ specifies the purposes or conditions the first party processes the data d_i^{col};*
- *$c_i^{dis} \in C^{dis}$ specifies the third parties to whom the first party discloses the data;*
- *$c_i^{ret} \in C^{ret}$ denotes retention (expiration) periods associated with collected data type d_i^{col}.*

For example, the first four sections of OpenAI's privacy policy [40] map directly to the components defined above. Concretely, the policy states that it collects users' location information (d^{col}) such as IP address or device GPS from using their services (c^{col}), processes this data to provide users more accurate responses (c^{pro}), and may disclose it to hosting providers and customer-support vendors (c^{dis}). The policy also states that, chats data will be retained for up to 30 days (c^{ret}). In addition to this retention policy, for users who opt in to the Zero Data Retention API [39], the service does not retain their data.

Challenge 1: Privacy formalization. Privacy policies, written in natural language and with varied formats and terminologies used across different organizations, lack the necessary precision. This makes it difficult to automatically and accurately extract the relevant components to form a structured data model defined above.

[*]Other well-known regulations, such as the California Consumer Privacy Act (CCPA) [1], have similar requirements. Meanwhile, most organizations publish privacy policies to build trust with their users.

[†]Some privacy policies may include additional components, such as organization identifier, user rights and children's data handling. We focus on the core components for clarity.

2.3 AI Agents

AI agents receive a user's prompt and then plan and execute tasks autonomously. To fulfill the task they often perform multiple rounds of interaction with the user and external tools. In this paper we focus on the data interactions; consequently, an agent's data interactions during a task execution are abstracted by the model below.

Definition 2 (Data interactions of an AI agent). *An AI agent's data interactions during a task execution are modeled by a finite sequence $\langle d_1, d_2, \ldots, d_N \rangle$, where each d_i is a data item and can be classified into one of four categories:*

- user$\rightarrow$agent*: data provided by or collected from the user (e.g. prompts, personal files);*
- user$\leftarrow$agent*: data processed and produced by the agent and returned to the user (e.g. generated responses);*
- agent$\rightarrow$third*: data sent to external services or tools (e.g. tool calls, API requests);*
- agent$\leftarrow$third*: data received from external services or tools (e.g. tool outputs).*

Among these data interactions, the data flow from the user to the agent (user$\rightarrow$agent) and from the agent to third parties (agent$\rightarrow$third) are of particular interest, as they represent the points where users' sensitive information may be out of users' control and potentially violated against privacy policies. The other two categories (user$\leftarrow$agent and agent$\leftarrow$third) are also relevant, as they may imply data processing and retention practices of the agent.

Challenge 2: Limited visibility. AI agents often operate with unstructured prompts and responses, complicating efficient identification of sensitive data within their interactions. Furthermore, these interactions often involve multiple rounds of communication, further hindering the temporal detection of data flows.

2.4 Privacy Policy Compliance

In a typical AI agent usage scenario, a user interacts with an agent A to execute tasks on their behalf, often providing the agent with access to their personal data D.[‡] Though the AI agent has privacy policy P that outlines its data collection and disclosure practices, and users may trust these agents to handle their data, the lack of transparency in which data is exactly collected and disclosed raises concerns about compliance with the stated privacy policies and also users' privacy expectations.

Definition 3 (Privacy Policy Compliance for an AI Agent). *Let* $(d_A^{\text{col}}, c_A^{\text{col}}, c_A^{\text{pro}}, c_A^{\text{dis}}, c_A^{\text{col}})$ *be any data practice during a task execution of an AI agent A, where* d_A^{col}, c_A^{col}, c_A^{pro}, c_A^{dis}, *and* c_A^{col} *are data items collected, along with their collection, processed, disclosed conditions, and retention periods, respectively. We say that A is compliant with a privacy policy P if*

$$P \models \left(d_A^{\text{col}}, c_A^{\text{col}}, c_A^{\text{pro}}, c_A^{\text{dis}}, c_A^{\text{col}}\right)$$

meaning that policy P semantically permits (or entails) the specified data practices.

Auditing the privacy entailment defined in Definition 3 is difficult in practice, even if we assume that the privacy policy model P and d_A^{col} are accurately identified. This difficulty arises from the large number of data items, their syntactic variations, and temporal dependencies that can occur during a task execution.

Challenge 3: User effort and efficiency. There is a gap between the formal privacy policy model and the actual data practices of AI agents. Bridging this gap requires tools that can automatically and efficiently audit data flows without imposing significant burdens on users. Meanwhile, AI agents may produce a large volume of data interactions during task execution, requiring scalable and effective auditing mechanisms, especially for on-the-fly scenarios.

2.5 Real-time Visualization

AI agents often operate in long-running sessions, so monitoring their data practices requires real-time visibility. Post-hoc logs are therefore insufficient; visualizations must present data interactions as they occur, enabling users to instantly identify privacy risks.

Real-time visualization architectures typically comprise three layers: a data-collection layer that captures the agent's interactions with users, context, and external services as they occur; a streaming layer that filters, normalizes, and routes captured events as they occur; and a visualization layer that renders these events to users in real time. Beyond the standard engineering concerns, achieving a user-friendly visualization for AI agents requires addressing additional challenges.

Challenge 4: User-friendly visualization. To be broadly usable, the real-time visualization must be both agent-agnostic and platform-independent. Agent-agnostic deployment ensures compatibility with diverse agent frameworks by avoiding coupling to specific architectures. Platform-independent integration enables the system to run across operating systems and runtime environments, allowing users to use with minimal setup regardless of their platform.

This paper thus aims for an automated and user-friendly tool that effectively addresses the challenges of privacy formalization, limited visibility, efficient auditing, and user-friendly visualization. AudAgent is designed to enable users to easily audit their AI agents' data practices against the privacy policies they care about, ensuring transparency and accountability while minimizing user effort and resource overhead.

3 AudAgent

This section describes the design of AudAgent. To address the three challenges identified above, AudAgent incorporates the following key design elements:

- *(Voting-based policy formalization with LLMs)* AudAgent employs multiple LLMs to independently interpret the natural-language privacy policy documents and then aggregates their outputs through equivalence checking and majority voting. This voting approach is once-and-for-all, producing quantifiable improvement in confidence of the final policy model.
- *(Model-Guided Data Annotation)* AudAgent designs data annotation methods guided by the privacy policy model, tailored separately for the collection, processing, disclosure, and retention stages. These methods complement the lightweight Presidio [37] analyzer, enabling low-overhead annotation of sensitive data during AI agent execution.
- *(Privacy auditing via ontology graph and automata)* AudAgent employs ontology alignment to reconcile semantic variations and hierarchical inclusions between policy categories and detected entities, and uses automata-based evaluation to compile policy constraints into efficient, on-the-fly checks that can run in parallel.
- *(Visualization via HTTP Analysis and WebSocket)* AudAgent integrates a real-time data collection layer using HTTP analysis to capture the AI agent's data interactions, a streaming layer using WebSocket to transmit annotated data and audit results, and a visualization layer based on `React.js`, which together ensure agent-agnostic and platform-independent.

Figure 1 illustrates the overall workflow of AudAgent, consisting of the above four components.

3.1 Voting-Based Policy Formalization

Privacy policy formalization is the first step in AudAgent, which transforms a natural-language privacy policy document into a structured, machine-checkable model as defined in Definition 1. Manual formalization (or analysis) requires expert knowledge and is labor-intensive, while statistical and symbolic extraction methods [14, 48] may struggle with the complexity and variability of natural language. LLM-based approaches have achieved state-of-the-art performance in privacy policy analysis [23, 48, 51, 52], owing to their superior natural-language understanding capabilities. Nonetheless, relying on a single LLM can lead to inaccuracies due to biases or occasional errors, and also does not provide any *confidence* in the correctness of the output. To enhance the accuracy and confidence of the result, we design a majority voting approach that leverages multiple LLMs, ensuring a comprehensive understanding of the document. In this process, one challenge is to identify semantically equivalent elements across different LLM outputs.

[‡]If the user is asked to provide sensitive data in the prompt to agent A, we treat that data as collected data. Such passive data collection behavior is often explicitly stated in the privacy policy of AI agents, e.g. AutoGPT [18].

Algorithm 1: LLM voting for privacy policy formalization

```
Require: Privacy policy document doc, LLMs $\{L_1, L_2, \ldots, L_M\}$
         and their prompt, voting threshold $\tau$
Ensure: Privacy policy model $P$ as per Definition 1
  ▷ Interpretations from LLMs
1 for $m \leftarrow 1$ to $M$ do
2     $P_m \leftarrow L_m(\mathtt{doc}, \mathtt{prompt})$;
  ▷ Vote aggregation and deduplication
3 $V^{\sim} \leftarrow$ votes from equivalence class as Formula (1);
4 $P^{\sim} \leftarrow \cup_{m=1}^{M} P_m \,/\!\sim$;                ▷ Modulo equivalence
  ▷ Final model by thresholding
5 $P \leftarrow \{e \in P^{\sim} : v(e) \geq \tau\}$;
6 return $P$;
```

3.1.1 LLM Voting. AudAgent's LLM voting approach for privacy policy formalization simulates a multi-participant decision-making process, with each LLM contributing its own interpretation of the policy. The final policy model is derived from the consensus among the LLMs, enhancing both accuracy and confidence in the result.

Specifically, given a privacy policy document and M different LLMs $\{L_1, L_2, \ldots, L_M\}$, each LLM is prompted to analyze the document,[§] and produce its interpretation P_m following the template in Definition 1. Elements in different P_i tend to be semantically equivalent, so we perform semantic equivalence checks to deduplicate them and count the number of votes for each semantically distinct element. Without loss of generality, we call two elements $e_i^{\text{col}} := (d_i^{\text{col}}, c_i^{\text{col}}, c_i^{\text{pro}}, c_i^{\text{dis}}, c_i^{\text{ret}})$ and $e_j^{\text{col}} := (d_j^{\text{col}}, c_j^{\text{col}}, c_j^{\text{pro}}, c_j^{\text{dis}}, c_j^{\text{ret}})$ in P_m semantically equivalent if they are interpreted from the same text span in the document, denoted by $e_i^{\text{col}} \sim e_j^{\text{col}}$. Then, the equivalence class of e_i^{col} can be grouped as a set

$$[e_i^{\text{col}}] := \left\{ e_j^{\text{col}} \in \cup_{m=1}^{M} P_m : e_j^{\text{col}} \sim e_i^{\text{col}} \right\}. \tag{1}$$

Each element in the equivalence class $[e_i^{\text{col}}]$ receives one vote from some L_m, and the total votes for the class is $v(e_i^{\text{col}}) = |[e_i^{\text{col}}]|$, i.e. the number of elements in the class.

In such a way, we find all equivalence classes in $\cup_{m=1}^{M} P_m$ and deduplicate them. Assuming the deduplicated privacy policy model is $P^{\sim}$, with each element having received a certain number of votes, the final policy model is formed by including only those elements in $P^{\sim}$ that receive at least τ votes, where τ is a pre-defined voting threshold. Algorithm 1 summarizes the LLM voting process for privacy policy formalization.

3.1.2 Benefits: Accuracy and Confidence. (i) The LLM voting approach ensures that the final privacy policy model P is generally at least as accurate as one individual LLM. (ii) It also provides a measure of confidence in the correctness of P based on the number of votes each element receives.

Theorem 1 (Confidence boost from LLM voting). *Assume the ideal privacy policy model is P^*. Given M independent LLMs each with probability $\alpha > 0.5$ judging an element $e \in P^*$ or $e \notin P^*$ correctly, when there are m LLMs votes for $e \in P^*$, the probability that $e \in P^*$, i.e. actually being in the ideal policy model, is*

$$\Pr[e \in P^*] = \left(1 + \left(\frac{1-\alpha}{\alpha}\right)^{2m-M}\right)^{-1}.$$

Proof. (Sketch) The proof follows from the Bayes' theorem with each LLM having the same prior. Appendix A.1 provides details. ◻

As shown in the theorem, the confidence $\Pr[e \in P^*]$ increases with $2m - M$, the margin by which the votes for $e \in P^*$ exceed the votes for $e \notin P^*$. This means that the more LLMs agree on the inclusion of an element, the higher the confidence that the element is indeed part of the ideal policy model.

Example 1. *Suppose there are $M = 5$ LLMs, each with an independent probability $\alpha = 0.8$ of correctly judging whether $e \in P^*$. If $m = 4$ LLMs vote for $e \in P^*$, then the probability that $e \in P^*$ is approximately 0.94. This demonstrates the high confidence achieved through strong consensus among the LLMs, i.e. from 0.8 to 0.94.*

Although the exact value of α is unknown in practice, Theorem 1 provides a quantitative improvement of confidence through LLM voting compared to relying on a single LLM.

3.1.3 User-Defined Privacy Policies. In addition to organizational privacy policies, AudAgent allows users to define their own privacy policies. Users can specify which data types they are comfortable sharing and under what circumstances using natural-language documents. This flexibility empowers users to take control of their privacy while still benefiting from the functionality of AI agents.

3.2 Model-Guided Data Annotation

Data annotation is the second step in AudAgent, which identifies the data being used by the AI agent during its execution and labels it with necessary metadata. This metadata is crucial for correlating the agent's data practices with the formalized privacy policy model. Two challenges arise in this step: (i) efficiency and overhead, as data annotation needs to be performed continuously during the agent's execution; (ii) specificity and accuracy, as the annotation serves for a particular privacy policy model and needs to capture relevant data types, purposes, and retention periods.

AudAgent addresses these challenges by combining a lightweight, off-the-shelf data analyzer with model-guided annotation methods tailored to privacy policy models.

3.2.1 The Presidio Analyzer [37]. Presidio is a lightweight, open-source data analyzer developed by Microsoft for detecting and anonymizing sensitive data in text. It extends the spaCy [10] NLP library by integrating regex, named-entity recognition models, and other logic to identify personally identifiable information (PII) and other sensitive data types in unstructured text. The average latency of Presidio is less than 100ms for moderate-length text [46], making it suitable for real-time AI agent execution.

Input and output. Presidio takes text strings as input and outputs a list of detected entities, each including the entity type, start and end positions, and a confidence score. It supports optional parameters, such as predefined entity types and language models, to customize the detection process. An example of Presidio's input and output is shown below.

[§] Due to the structured nature of privacy policies, prompt engineering is simpler to help LLMs focus on relevant sections and extract necessary elements.

```
input: "My name is John Doe and my email is
        john.doe@example.com"
output: entity_type: "PERSON",
        start: 11, end: 19, score: 1;
        entity_type: "EMAIL_ADDRESS",
        start: 36, end: 56, score: 1;
```

An online demonstration of Presidio is available at [35].

Integration with AI Agents. Data interactions between AI agents and users are often text-based, such as user prompts and documents.[¶] AI agents may also interact with third-party tools, such as Gmail or Calendar, which are also involved in text-based data exchanges. Presidio can be integrated into the AI agent's workflow to analyze PII and sensitive data in these text interactions in real-time. Nonetheless, Presidio alone is insufficient for comprehensive data annotation, as it does not capture the context-specific information restricted by privacy policies, such as conditions and retention periods.

AudAgent complements Presidio with model-guided annotation methods that leverage the formalized privacy policy model to provide context-aware data annotation, which is described in the next subsection.

3.2.2 *Data Annotation Guided by Privacy Policy Models.*

AudAgent designs data annotation methods guided by the privacy policy model, tailored separately for the collection, processing, disclosure, and retention. When Presidio detects an entity in the text, AudAgent annotates it with additional metadata based on the structure of the privacy policy model.

Recall that the privacy policy model P comprises four components: $d^{\mathrm{col}}, c^{\mathrm{col}}, c^{\mathrm{pro}}, c^{\mathrm{dis}}, c^{\mathrm{ret}}$, each specifies a data type along with its associated conditions for collection, processing, disclosure, and retention, respectively. To audit compliance of the AI agent's data practices with the privacy policy, AudAgent must annotate detected entities with the pertinent conditions.

Collection condition c^{col}. A privacy policy specifies the conditions under which certain data types are collected. It typically includes direct collection, where users explicitly provide data to the agent, and indirect collection, where users provide data through interactions with third-party services.[‖] Based on these two modes of data collection, AudAgent annotates a sensitive data type d^{col} that users directly send to the agent, e.g. within user prompts or files, with the condition $c^{\mathrm{col}} = \text{direct}$. Similarly, if d^{col} is sent to the agent through third-party tools, e.g. in API requests or tool calls, AudAgent annotates it with $c^{\mathrm{col}} = \text{indirect}$. These annotations are finished instantly as they are explicit according to the interactions.

Purpose c^{pro}. Data use purposes are often too broadly stated in privacy policies, expressed in vague terms such as "to provide and improve our services", which apply to *all* services of the organization beyond an AI agent's functionalities. This makes identification of data use purposes from the agent's interactions unrealistic. Thus, AudAgent focuses on a more specific task relevant/irrelevant classification of data use purposes. We say collected data type d^{col} is relevant to the AI agent's task ($c^{\mathrm{pro}} = \text{relevant}$) if it directly supports the task at hand, i.e. appears in following interactions, such as user prompts or tool calls. Otherwise, it is considered irrelevant ($c^{\mathrm{pro}} = \text{irrelevant}$). A detected irrelevant data use purpose may indicate potential over-collection or misuse of data by the AI agent, or it may be used for other purposes not explicit in the agent's interactions.

Disclosure condition c^{dis}. A privacy policy specifies which third parties the organization may share certain data types with, such as service providers, affiliates, or legal authorities. Thus, when the agent sends a detected sensitive data type d^{dis} to a third-party tool, AudAgent annotates it with the disclosure condition c^{dis} set to the name of that tool or service.

Retention condition c^{ret}. The actual retention period for each collected data type d^{col} can be recorded from the collection time and the following appearance times in the execution trace of the AI agent. Specifically, when d^{col} is first collected at time t_0, AudAgent sets its retention period to $c^{\mathrm{ret}} = 0$. (i) If d^{col} is not collected again, each time d^{col} appears at time t_i in the agent's interactions, AudAgent updates its retention period to $c^{\mathrm{ret}} = t_i - t_0$. (ii) If d^{col} is collected again at time t_j, AudAgent resets its retention period to $c^{\mathrm{ret}} = 0$, and continues to update it as in (i).

Annotation output. AudAgent annotates each detected sensitive data instance with the relevant metadata as described above. If an instance passes the consistency checks, i.e. not stopped by policy violations, it is annotated with the following metadata:

- the collected data type d^{col} (from Presidio);
- the collection condition c^{col} (direct or indirect);
- the relevance of processing c^{pro} (relevant or irrelevant);
- the disclosure condition c^{dis} (name of third party);
- the retention period c^{ret} (time duration).

Along with these five fields, AudAgent also includes the position of d^{col} and the confidence score from Presidio.

Example 2. *Consider an AI agent that collects a user's email address through a prompt, uses it to send a welcome message by requesting Gmail API with the email address and content. The content does not contain other sensitive data, and this task is completed within 3 seconds. Presidio detects the email address as a sensitive data instance annotated* $d^{\mathrm{col}} = \text{EMAIL\_ADDRESS}$. *AudAgent then annotates it with the following metadata:* $c^{\mathrm{col}} = \text{direct}$, $c^{\mathrm{pur}} = \text{relevant}$, $c^{\mathrm{dis}} = \text{Gmail}$, $c^{\mathrm{ret}} = 3\text{s}$.[\**]

3.2.3 *Soundness.*

The soundness of a data annotation mechanism is essential for reliable privacy auditing.

Definition 4 (Soundness of data annotation). *Denote* $P = \{d_i^{\mathrm{col}}, c_i^{\mathrm{col}}, c_i^{\mathrm{pro}}, c_i^{\mathrm{dis}}, c_i^{\mathrm{ret}}\}_{i=1}$ *as a privacy policy model. An annotation mechanism* $\mathcal{M}$ *is sound w.r.t* P *if for every data practice trace* τ *and every sensitive data type* $d^{\mathrm{col}} \in D^{\mathrm{col}}$ *that appears in* τ *there* exists *an annotation* $\left(d^{\mathrm{col}}, c^{\mathrm{col}}, c^{\mathrm{pro}}, c^{\mathrm{dis}}, c^{\mathrm{ret}}\right) \in \mathcal{M}(\tau)$ *such that correctly reflects the possible conditions in* P.

Soundness means that all sensitive data instance described by the policy is detected when it appears and all required condition

[¶] While AI agents may also handle non-text data (e.g. images, audio), they can be converted to text through OCR or speech-to-text techniques. Presidio also supports image input; we focus on text for simplicity.

[‖] Some privacy policies may also include negative conditions, e.g. "we do not collect your email", which is reflected in the policy model but not in the data annotation. Some policies may also specify more fine-grained conditions, e.g. "we collect your email when you sign up". AudAgent can be extended to handle such cases by recording fine-grained dependencies in the annotation process.

[**] Note that if the email address is not re-collected but appeared again in later interactions, the retention period would be increased from 3s.

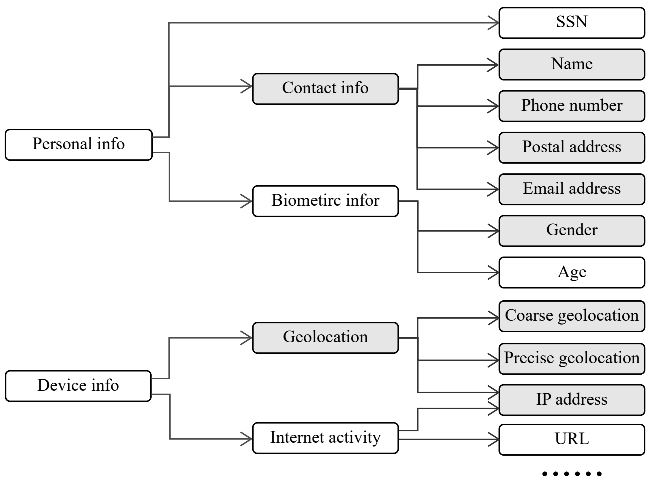


Figure 2: Example ontology graph of data types. Some items can share the same parent node, e.g. "IP address".

annotations can be correctly produced. It specifies the minimum *ability* of $\mathcal{M}$ to reliably give a "compliance" answer. A sound annotation mechanism $\mathcal{M}$ may have stronger ability beyond correctly annotating D^{col} and their conditions in P, or it may produce false positives, e.g. annotating non-sensitive data. Nonetheless, $\mathcal{M}$ must be able to produce the correct annotations for D^{col} in P.[††]

A sound annotation mechanism always gives a correct "compliance" answer, i.e. when the annotated data passes the compliance checks, it is guaranteed no policy violations occur. However, it may give incorrect "non-compliance" answers, as it can not guarantee that annotation beyond P is correct. The following theorem states the soundness of AudAgent's data annotation mechanism under specified assumptions.

Theorem 2 (Soundness of data annotation in AudAgent). *Data annotation given by AudAgent is sound w.r.t privacy policy model P, as per Definition 4, under the following assumptions:*

- *(Form of the privacy policy model) P restricts sensitive data type D^{col} with their collection conditions $C^{\mathrm{col}} = \{\text{direct}, \text{indirect}\}$, purpose relevance $C^{\mathrm{pur}} = \{\text{relevant}, \text{irrelevant}\}$, disclosure conditions C^{dis} as names of third parties.*
- *(Accuracy of annotation) AudAgent detects all sensitive data types D^{col} and their conditions in P.*

Proof. The first assumption ensures that all conditions in P can be correctly reflected by AudAgent's annotation mechanism. The second assumption ensures that all sensitive data types in P are detected and annotated when they appear in the data practice trace. Thus, for every sensitive data type $d^{\mathrm{col}} \in D^{\mathrm{col}}$ that appears in the trace, there exists an annotation $(d^{\mathrm{col}}, c^{\mathrm{col}}, c^{\mathrm{pro}}, c^{\mathrm{dis}}, c^{\mathrm{ret}}) \in \mathcal{M}(\tau)$ that correctly reflects the possible conditions in P. □

[††]Completeness requires that *all* annotations produced by $\mathcal{M}$ for D^{col} are correct w.r.t. P. Pursuing completeness in isolation can lead to poor coverage: for example, a mechanism that never produces any annotations is complete but practically useless. Soundness and completeness are often at odds; we slightly favor soundness for annotation coverage.

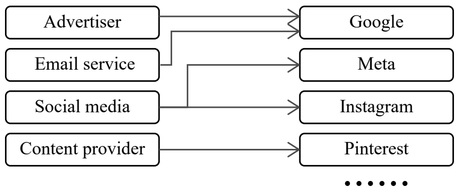


Figure 3: Example ontology graph of entities. Some entities can be classified into multiple categories, e.g. Google as an advertiser and an email-service provider.

Likelihood of assumptions. (i) The first assumption is a simplification that captures the core structure of most real-world privacy policies. Real policies can include additional complexities (e.g. finer-grained conditions and ontology of entities), which can be incorporated into AudAgent's annotation mechanism with further engineering. (ii) The second assumption reflects the design objective of AudAgent: the combined pipeline (Presidio + model-guided annotation) can detect and annotate the sensitive data types and their associated conditions specified in P as in the first assumption.

3.3 Privacy Auditing via Ontology Graph and Automata

So far we have a formalized privacy policy model and an annotated trace of the AI agent's data practices. The final step in AudAgent is to perform real-time auditing by checking the annotated instances against the policy model. Two practical challenges remain: (i) granularity mismatches between policy terms and detected annotations (for example, a policy document may refer to "contact information" while the data annotation reports "email address"); and (ii) requirements for efficient, on-the-fly compliance checks.

3.3.1 Ontology graph of Data Types and Entities. Privacy policy documents often refer to data types and entities at a higher level of abstraction while missing specific subtypes, leading to granularity mismatches with the data annotation results. To reconcile such mismatches, AudAgent leverages ontology graphs for data types and entities. Each ontology graph encodes hierarchical "is-a" relationships, enabling the auditor to map a concrete annotated type (e.g. "email address") to a broader policy term (e.g. "contact information"). The following definition formalizes the ontology graph for data types; the ontology graph for entities is defined similarly.

Definition 5 (Ontology graph of data types). *An ontology graph of data types O^{dat} is a directed acyclic graph where:*

- *Each node represents a data type (e.g. "contact information", "email address", "phone number").*
- *Each directed edge $A \to B$ indicates that B is a subtype of A (e.g. "email address" is a subtype of "contact information").*

This ontology graph allows AudAgent to map specific data types identified in the data annotation in an AI agent's execution trace to broader categories defined in the privacy policy model. When auditing, if an annotated data type matches or is a subtype of a data type in the policy model, it is considered compliant w.r.t. the privacy policy model.

Ontology graph construction. AudAgent constructs its ontology graph for the given privacy policy document via an *internal+external* approach. Internally, it incorporates hierarchical relationships from the privacy policy document itself as interpreted by the LLMs during the policy formalization step (Section 3.1). Externally, it complements the internal ontology graph with terms and relationships from CCPA [1], as many privacy policies are designed to comply with terms defined in CCPA. The combined ontology graph is then pruned to retain only nodes and edges relevant to the data types present in the privacy policy model.

Figure 2 shows an ontology graph for data types derived from internal and external sources. Similarly, AudAgent builds an ontology graph of third-party entities (e.g. classifying Google as an advertising provider and an email-service provider) by leveraging external knowledge in the Ghostery Tracker Database [29].[‡‡] Figure 3 shows an example ontology graph for entities.

Together, the data-type and entity ontology graphs allow AudAgent to reconcile granularity mismatches during auditing. Both graphs are hardcoded so they introduce no runtime overhead during auditing.

3.3.2 *Automaton for On-the-Fly Privacy Auditing.*

So far, annotated data instances produced during the AI agent's execution have been reconciled with the privacy policy model. The final step is an on-the-fly compliance check: upon arrival of each annotated instance as data practices unfold, AudAgent needs to check whether it satisfies the collection, purpose, disclosure, and retention constraints. Among these, the retention constraints complicate the auditing as they require tracking how long data are stored, reused, or refreshed over time.

To efficiently handle retention constraints and enable real-time auditing, AudAgent compiles the privacy policy model P into a set of simple automata. Each automaton corresponds to a specific data type d^{col} in P and tracks its state based on the annotated instances observed during the AI agent's execution. The automaton's states represent the current status of d^{col} with respect to its $c^{\text{col}}, c^{\text{pur}}, c^{\text{dis}}$ and c^{ret} conditions. The transitions between states are triggered by the arrival of annotations, and the automaton checks compliance with the policy model at each transition.

Definition 6 (Auditing automaton). *Given a data type $d^{\text{col}} \in D^{\text{col}}$ in the privacy policy model P, its corresponding auditing automaton $\mathcal{A}_{d^{\text{col}}}$ is defined as a tuple $(d^{\text{col}}, Q, \Sigma, \delta, F)$ where:*

- *d^{col} is the initial state representing the data type being tracked.*
- *$Q := \{d^{\text{col}}, \text{col}, \text{pur}, \text{dis}\}$ is a set of states representing the status of d^{col} w.r.t. its conditions in P.*
- *Σ is the input alphabet, consisting of conditions in $C^{\text{col}}, C^{\text{pur}}, C^{\text{dis}}$, and C^{ret} from P.*
- *$\delta_{d^{\text{col}}} : Q \times \Sigma \to Q$ is the state transition function.*
- *$F \subseteq Q$ is the set of accepting states indicating compliance with the policy model.*

The privacy policy model P implies a set of automata $\{\mathcal{A}_{d^{\text{col}}} : d^{\text{col}} \in D^{\text{col}}\}$, meaning compliance of d^{col} with the policy is equivalent to the automaton $\mathcal{A}_{d^{\text{col}}}$ being in an accepting state. Due to the retention constraints, (i) the automaton tracks the retention period c^{ret} at each transition, (ii) the automaton set may accept d^{col} without entering the col state if c^{ret} is within the allowed retention period in P.

Example 3. *Consider the privacy policy of AutoGPT [47], which states that it collects users' email addresses directly for providing products and services, and may disclose them using Google Workspace API. It says user data are retained as long as necessary; nonetheless we assume for this example that 30 days under OpenAI's data retention policy [40] for demonstration. Therefore, the corresponding policy model P for data type d^{col} =* EMAIL_ADDRESS *includes: c^{col} =* direct, *c^{pro} =* relevant, *c^{dis} =* Google Workspace API, *and c^{ret} =* 30 days. *The corresponding automaton $\mathcal{A}_{d^{\text{col}}}$ is illustrated in Figure 4.*

Acceptance and rejection. The auditing automaton is evaluated with the sequence of annotated instances produced during the AI agent's execution to check compliance with the privacy policy model. (i) It accepts an annotated instance if it ends in an accepting state, indicating compliance with the policy. (ii) It rejects an annotated instance if it sticks in a non-accepting state at any point, indicating a violation from the data type and conditions in the policy model. (iii) In addition to rejection by automata, AudAgent also rejects an annotated instance if it cannot trigger any automaton in the automaton set, indicating a violation from data types not covered by the policy model.[§§]

Using the example in Figure 4, if there is an annotated email address instance collected directly, the automaton transitions to state col, indicating compliance with the collection condition. Meanwhile, a timer starts to track the retention period. If then, the instance is used for subsequent interactions relevant to the AI agent's task and within the retention period, the automaton transitions to the purpose state pur, indicating compliance with the purpose condition. Now, depending on whether disclosure happens within the retention period, we have two cases: (i) If disclosure does not happen, the automaton without disclosure constraints remains in the accepting state pur. (ii) If disclosure happens, e.g. to Google Workspace API within the retention period, the automaton with disclosure constraints transitions to accepting state dis, indicating compliance with the disclosure condition. An annotated email address instance may be processed or used again after 20 days in later interactions (without re-collection), the automaton transitions to the purpose state pur again, as the retention period is satisfied. In other cases, there exists violations caused by data type d^{col} = EMAIL_ADDRESS.

Soundness and completeness. Soundness means that if the automaton accepts an annotated instance, it complies with P. Completeness means that if an annotated instance complies with P, the automaton accepts it.

Theorem 3 (Soundness and completeness of AudAgent's auditing automata). *Given a privacy policy model P having the assumed forms in Theorem 2, and its corresponding auditing automaton set $\{\mathcal{A}_{d^{\text{col}}} : d^{\text{col}} \in D^{\text{col}}\}$, with accurate ontology graphs for data types and entities, AudAgent's privacy auditing by the automaton set is sound and complete w.r.t. P, i.e.*

[‡‡] AudAgent can also use other ontology graphs. For example, DuckDuckGo's Tracker Radar [2] provides a larger set of third-party services and their categories.

[§§] In fact, this is also sticking in a non-accepting state, as it sticks before entering any initial state of the automata.

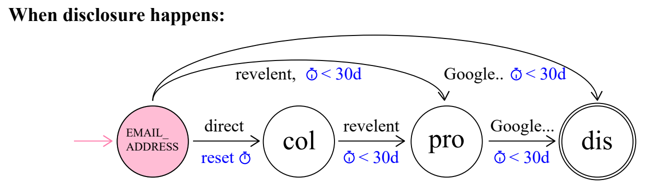


Figure 4: An example auditing automata accepting the dis **state for data type d^{col} =** EMAIL_ADDRESS**. The initial state is the data-type state (pink circle); accepting states are double-circled. Transitions record collection, purpose, disclosure, and retention constraints for compliance. Retention constraints are tracked by a timer (blue clock icon).**

- *(Soundness) If an annotated instance is accepted by its corresponding $\mathcal{A}_{d^{\text{col}}}$ depending on with or without disclosure constraints, it complies with P.*
- *(Completeness) If an annotated instance complies with P, it is accepted by its corresponding $\mathcal{A}_{d^{\text{col}}}$ depending on with or without disclosure practices.*

Proof. (Sketch) The proof follows from the construction of the automaton set from P and the definition of acceptance and rejection. Each component of P is directly mapped to states and transitions in the automata, and we prove that all possible data interaction patterns are covered by the automata, ensuring that acceptance implies compliance (soundness) and compliance implies acceptance (completeness). Appendix A.2 provides details. □

So far, the auditing problem is reduced to evaluating the automaton set with the annotated instances. The next paragraph describes how AudAgent efficiently evaluates the automaton set for on-the-fly auditing.

3.3.3 On-the-fly Auditing by Parallel Evaluation. One benefit of using automata is their mature efficient evaluation algorithms [32], such as parallelization [38, 45]. They can be adapted to AudAgent, significantly reducing the overhead compared to non-automata-based auditing.

AudAgent evaluates all annotated instances against the automaton set in parallel. Specifically, it maintains a vector of current states of all annotated instances $\{d_1^{\text{col}}, \ldots, d_n^{\text{col}}\}$ for their corresponding auditing automata, initialized as stat = [null, …, null]. These states will trigger transitions in the automata as the AI agent's execution unfolds, i.e. an annotation's data type d_j^{col} or its condition as input will trigger a state transition $\sigma \in \Sigma$ as per Definition 6, AudAgent updates the corresponding state vector in stat based on the transition function of each data type $\delta_{d_j^{\text{col}}}$:

$$\text{stat}[j] \leftarrow \delta_{d_j^{\text{col}}}(\text{stat}[j], \sigma).$$

The transition function $\delta_{d_j^{\text{col}}}$ can be statically stored as a three-dimensional lookup table (with dimensions for automata, current state, input symbol), enabling $\Theta(1)$ time complexity for each state update with multi-threading.

If one data practice on d_j^{col} is allowed, the corresponding update should be successful, meaning that the input symbol σ matches the constraints for d_j^{col} in the automaton $\mathcal{A}_{d_j^{\text{col}}}$. If not, the update fails and sticks in a non-accepting state, indicating a violation. When an automaton $\mathcal{A}_{d_j^{\text{col}}}$ reaches an accepting state, it indicates compliance with the policy model for that data type. Algorithm 2 summarizes on-the-fly auditing by parallel evaluation of all annotated instances against the automaton set.

Algorithm 2: On-the-fly auditing by parallel evaluation of all annotated instances against automata

```
Require: arriving annotation
    $\{(d_i^{\text{col}}, c_i^{\text{col}}, c_i^{\text{pro}}, c_i^{\text{dis}}, c_i^{\text{ret}}) : i = 1, \ldots, n\}$, automaton set
    $\{\mathcal{A}_{d_i^{\text{col}}} : d_i^{\text{col}} \in D^{\text{col}}\}$ from privacy policy model $P$
Ensure: auditing result: compliance or violation
stat ← [null, …, null] ;                              ▷ states of $\{d_i^{\text{col}}\}$
while AI agent is running do
    if annotation $\{d_j^{\text{col}}\}_{j \in J}$ arrive then
        ▷ initialize via ontology graph
        stat[$j$] ← $O^{\text{dat}}(d_j^{\text{col}})$;
    if annotation $\{c_j^{\text{col}}\}_{j \in J^{\text{col}}}$ arrive then
        stat[$j$] ← $\delta_{d_j^{\text{col}}}(\text{stat}[j], c_j^{\text{col}})$;
        if exist $j \in J^{\text{col}}$ that stat[$j$] ≠ col then
            return violation;
        $t_{d_j^{\text{col}}} \leftarrow 0$ ;                  ▷ renew retention timer
    if annotation $\{c_j^{\text{pur}}\}_{j \in J^{\text{pur}}}$ arrive then
        update $t_{d_j^{\text{col}}}$;
        stat[$j$] ← $\delta_{d_j^{\text{col}}}(\text{stat}[j], c_j^{\text{pur}}, t_{d_j^{\text{col}}} < c_j^{\text{pur}})$;
        if exist $j \in J^{\text{pur}}$ that stat[$j$] ≠ pur then
            return violation;
    if annotation $\{c_j^{\text{dis}}\}_{j \in J^{\text{dis}}}$ arrive then
        update $t_{d_j^{\text{col}}}$;
        stat[$j$] ← $\delta_{d_j^{\text{col}}}(\text{stat}[j], O^{\text{ent}}(c_j^{\text{dis}}), t_{d_j^{\text{col}}} < c_j^{\text{dis}})$;
        if exist $j \in J^{\text{dis}}$ that stat[$j$] ≠ dis then
            return violation;
save $\{t_{d_j^{\text{col}}}\}_{j \in J}$ for next auditing;
return compliance;
```

Complexity analysis. Algorithm 2 has $\Theta(1)$ time complexity per annotated instance. The computation comes from updating automaton states by the transition function $\delta_{d_j^{\text{col}}}$. Since AudAgent uses simple finite state automata with 4 states and 4 input symbols, the transition function can be evaluated in negligible time. Moreover, all annotated instances are independent and can be evaluated in parallel by searching a lookup table for the automaton set, described in previous paragraphs. This parallelization reduces the time complexity to $\Theta(1)$ for a set of annotated instances.

The space complexity of Algorithm 2 is $\max(n, |D^{\text{col}}|)$, where n is the number of annotated instances being tracked and $|D^{\text{col}}|$ is the number of data types in the privacy policy model. These two values are typically small in practice, causing negligible space overhead.

3.4 Visualization via HTTP Analysis and WebSocket

This subsection outlines the system architecture of AudAgent for real-time, user-friendly visualization of privacy auditing. The architecture comprises three layers and has two key design innovations: (i) Agent-agnostic auditing: By inspecting known LLM HTTP endpoints, AudAgent reconstructs the agent's control flow from observed requests and responses; (ii) Platform-independent delivery: A browser-based frontend combined with WebSocket streaming provides low-latency, cross-platform updates from the auditing backend. The three layers are described below.

Data collection layer. AudAgent's data-collection layer captures HTTP requests and responses between the AI agent, its tools, and underlying LLMs,[¶¶] and extracts auditable data practices from this traffic. When users submit prompts, the agent forwards them to the LLM via a known API endpoint (e.g., OpenAI at `api.openai.com`),[***] and receives model responses; tool use are observed similarly. By analyzing this traffic, AudAgent asynchronously recovers user prompts, tool calls, and model responses for annotation and auditing.

The extracted artifacts feed two subprocesses: (i) reconstruction of the agent's control flow and (ii) privacy auditing as described in above sections. Outputs of these subprocesses are sent to the streaming layer via a message queue.

Streaming layer. The streaming layer performs real-time communication between the auditing backend and the React-based visualization frontend using WebSocket connections. It processes reconstructed control flows and auditing results, serializes and formats them into messages, and delivers them over WebSocket to the frontend. This layer operates independently of the AI agent's execution process and continuously streams updates so users can monitor data events and privacy compliance in real time without interrupting the agent's operation.

Visualization layer. The visualization layer provides a platform-independent web interface for monitoring the AI agent's data practices and privacy compliance. Implemented in `React.js` with a component-based architecture, the frontend consumes control-flow and auditing messages streamed over WebSocket. Features include data flow diagrams showing how data moves through the agent and its tools, real-time highlighting of detected potential policy violations, and a timeline of data events. The design takes inspiration from existing visualization tools [25] but introduces new components and real-time privacy auditing. By using web browsers as the primary visualization platform, AudAgent achieves wide compatibility across operating systems and devices, while it remains easy to extend to mobile or desktop deployments by React Native.

4 Application Examples

This section shows an example of how AudAgent visualizes sensitive data usage during an AI agent's task execution to improve user-side transparency. It also illustrates how AudAgent can aid in refining privacy policy documents by identifying potential violations.

Setup. We run AudAgent together with Microsoft's AutoGen [36], a popular open-source framework for constructing AI agents that combine LLMs with external tools. Using AutoGen, we build an agent composed of an Anthropic Claude LLM [15] and several searching tools. The AutoGen orchestrator executes locally on the user's machine, while the Claude model and the third-party search services are cloud-hosted and treated as partially trusted in our threat model.

We consider the following privacy policy and user task:

- Privacy policy: Anthropic's privacy policy [17], combined with a user-defined rule that forbids disclosing any personal email address to third-party tools.
- Task: The user prompts the agent with partial information about a friend, Bob, and asks the agent to search for Bob online and save a more complete contact profile.

Given this prompt, the Claude LLM generates reasoning steps and plans tool calls to search for Bob online. Throughout the agent's execution, AudAgent monitors the data practices of the Claude LLM and the third-party tools, and also audits them against the specified privacy policy.

Visualization of privacy risks. Figure 5 shows AudAgent's web frontend during the agent's execution. (i) The left panel displays the agent's real-time execution trace as a directed graph. Each node represents the user, the LLM, or a third-party tool, and edges denote request/response interactions. In this example, the user prompts partial information about Bob to the agent orchestrator, which forwards it to the Claude LLM. The LLM reasons about next steps, constructs a tool call to a third-party search tool, and submits that call (including Bob's information) via the orchestrator. (ii) The right panel presents detailed data practices for each interaction alongside AudAgent's privacy-auditing results. Data practices flagged as potential privacy risks are highlighted in red; safe practices are shown in green. Here, AudAgent detects a privacy violation on the edge from the agent interface to the "web_search_tool": the outbound message contains Bob's personal email address, which violates the user-defined rule forbidding disclosure of personal emails. The corresponding edge in the left panel is likewise highlighted with a red box to indicate the risk.

5 Related Work

This paper focuses on auditing AI agents' data practices against privacy policies, which relates most closely to AI agents' privacy and security, privacy policy analysis, and compliance auditing.

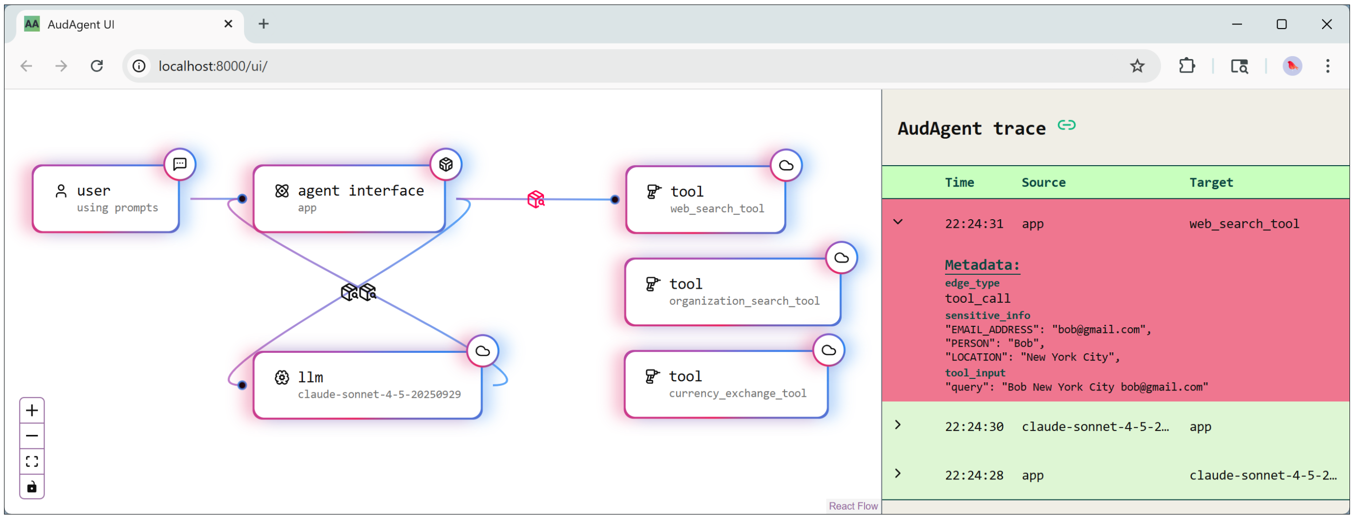
5.1 AI Agents' Privacy and Security

Privacy and security of AI agents are major concerns due to their autonomous decision-making, extensive data handling, and susceptibility to manipulation. Beyond sophisticated attack methods studied in academic research, several startups now offer tooling to safeguard agents' behavior and data practices [5, 20]. Below, we review representative research and industry efforts on this topic.

5.1.1 Attacks. LLMs at the core of AI agents can be exploited via malicious prompts to extract sensitive data [31, 33, 34, 41]. At the

[¶¶]Local LLMs typically expose HTTP endpoints on well-known ports; e.g. Ollama [9] listens on 11434 by default.

[***]Request forwarding is handled by the agent orchestrator (e.g. AutoGen [36] or LangChain [7]), which in our local setting runs on the same host so requests can be inspected at the application layer before transmission.



AudAgent trace

	Time	Source	Target
⌄	22:24:31	app	web_search_tool
›	22:24:30	claude-sonnet-4-5-2…	app
›	22:24:28	app	claude-sonnet-4-5-2…

```
Metadata:
edge_type
tool_call
sensitive_info
"EMAIL_ADDRESS": "bob@gmail.com",
"PERSON": "Bob",
"LOCATION": "New York City",
tool_input
"query": "Bob New York City bob@gmail.com"
```

Figure 5: The React.js web frontend of AudAgent, visualizing an AI agent's data practices and privacy auditing results in real time. The left panel shows the agent's execution trace as a directed graph, with nodes representing the user, LLM, and third-party tools, and edges representing request/response interactions. The right panel details the data practices for each interaction and highlights any potential privacy risks detected by AudAgent.

system level, compromised third-party tools integrated with AI agents can also cause risky behaviors and data leaks [27, 42, 50]. In addition to these attacks, inherent problems such as hallucination, misalignment, and resource-allocation failures can produce incorrect or misleading behavior that threatens data integrity and privacy; see a recent survey for further discussion [26].

AudAgent targets user data privacy by monitoring and auditing AI agents' data practices in real time, helping to detect potential privacy risks arising from both adversarial attacks and systemic failures.

5.1.2 Defenses. Defenses for AI agents' privacy and security can be grouped by focus: input-level and action-level. (i) Input-level defenses employ testing and filtering techniques to prevent malicious or sensitive content from being processed by the agent [44, 53]. For example, Maatphor [44] targets prompt-injection attacks via variant analysis to detect multiple malicious prompt variants, and FuzzLLM [53] uses fuzz testing to generate adversarial prompts that reveal LLM vulnerabilities. (ii) Action-level defenses constrain agent behavior to prevent undesired or risky actions. ToolEmu [43] and AgentDojo [54] use sandboxed or emulated environments to evaluate tool use and identify potential security risks. PrivacyAsst [55] forges user prompts with generative models as a runtime technique to protect sensitive inputs by rewriting or obfuscating prompts. Balunovic et al. [19] propose a rule-based runtime control that restricts agent actions according to predefined security rules; this approach is then reflected in their commercial tool from startup Invariant Labs [6], which offers a rule-based guardrails layer for agents built on the MCP framework [8].

None of these defenses address the problem of auditing AI agents' data practices against privacy policies. Fuzzing and sandboxing can find pre-deployment vulnerabilities, but they cannot protect runtime privacy or security for end users given uncertain inputs and the agents' dynamic contexts. Prompt-forging can reduce leakage but risks altering the user's original intent. Rule-based controls require authors to write precise rules in a specific format, which is burdensome for non-expert users; see [6] for their documentation. In contrast, AudAgent provides an automated, user-friendly auditing tool that continuously monitors AI agents' data practices in real time.

5.1.3 Privacy and Security Solutions from Startups. In addition to academic research, several startups provide system-level solutions to improve AI agents' privacy and security. For example, Guardrails AI [5] has a toolkit to block prohibited words and NSFW content, validate semantic logic, and enforce format constraints. Boomi AgentStudio [20] provides design-time governance and runtime monitoring for agent development; its website also highlights rule-based guardrails and real-time monitoring features.

5.2 Privacy Policy Analysis and Compliance Auditing

5.2.1 LLM-based Privacy Policy Analysis. In the pre-LLM era, privacy policy analysis relied on manual formalization and on statistical or symbolic extraction methods [14, 48]. Manual formalization is labor-intensive and requires expert knowledge, while statistical and symbolic techniques often struggle with the complexity and variability of natural language. Recent LLM-based methods achieve state-of-the-art results in privacy policy analysis [23, 48, 51, 52] due to their strong natural-language understanding.

Unlike prior LLM-based approaches that typically rely on a single model for extraction, AudAgent queries multiple LLMs and uses cross-LLM voting to aggregate confidence and reconcile extracted policy elements.

5.2.2 Compliance Auditing of Privacy Policies. Prior work on compliance auditing of privacy policies has largely targeted traditional applications such as mobile apps and web services [21, 22, 57]. For example, MAPS [57] analyzes Android apps and uses classifiers for policy formalization; PurPliance [22] focuses on data-usage purposes in mobile apps and primarily adopts rule-based formalization; ExtPrivA [21] targets browser extensions and provides in-browser privacy disclosures for other extensions.

Prior work targets conventional platforms rather than AI agents, which exhibit distinct architectures and data-handling behaviors. AudAgent instead proposes new techniques in automated policy formalization, runtime annotation, automata-based evaluation, and an intuitive visualization to audit AI agents' data practices against privacy policies in real time.

6 Conclusions

In this paper we present AudAgent, a visual tool for local, real-time auditing of AI agents' data practices against specified privacy policies. AudAgent comprises four components: policy parsing, runtime annotation, compliance auditing, and a user interface. It combines LLM-based automated policy formalization, a lightweight runtime analyzer for annotating data practices, ontology alignment and automata-based evaluation for efficient compliance checks, and a platform-independent visualization delivered over WebSockets. By continuously monitoring agent behavior, AudAgent improves transparency and accountability and enables timely detection of policy violations. Evaluations on agents implemented with mainstream frameworks show that AudAgent reliably identifies potential privacy-policy violations in real time, providing practical benefits to both end users and AI agent developers.

References

[1] 2018. *California Consumer Privacy Act (CCPA).* https://oag.ca.gov/privacy/ccpa

[2] 2025. *duckduckgo/tracker-radar.* https://github.com/duckduckgo/tracker-radar original-date: 2020-02-18T21:59:17Z.

[3] 2025. P3P. https://en.wikipedia.org/w/index.php?title=P3P&oldid=1308137243 Page Version ID: 1308137243.

[4] 2016. *General Data Protection Regulation (GDPR) - Legal Text.* https://gdpr-info.eu/

[5] 2025. *Guardrails AI.* https://www.guardrailsai.com/

[6] 2025. *Invariant Labs.* https://invariantlabs.ai/

[7] 2025. *LangChain.* https://www.langchain.com

[8] 2025. *Model Context Protocol: Introduction.* https://modelcontextprotocol.io/docs/getting-started/intro

[9] 2025. *Ollama.* https://ollama.com

[10] 2025. *spaCy · Industrial-strength Natural Language Processing in Python.* https://spacy.io/

[11] 2025. Use agent mode in VS Code. https://code.visualstudio.com/docs/copilot/chat/chat-agent-mode

[12] 2025. What are AI agents? https://github.com/resources/articles/ai/what-are-ai-agents

[13] 2025. What are AI agents? Definition, examples, and types. https://cloud.google.com/discover/what-are-ai-agents

[14] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie. 2019. PolicyLint: Investigating Internal Privacy Policy Contradictions on Google Play. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, Nadia Heninger and Patrick Traynor (Eds.). USENIX Association, 585–602. https://www.usenix.org/conference/usenixsecurity19/presentation/andow

[15] Anthropic. 2025. *Claude Developer Platform | Claude.* https://www.claude.com/platform/api

[16] Anthropic. 2025. Customers \ Anthropic. https://www.anthropic.com/customers

[17] Anthropic. 2025. *Privacy Policy.* https://www.anthropic.com/legal/privacy

[18] AutoGPT Team. 2025. *Platform Privacy Policy.* https://agpt.co/legal/platform-privacy-policy

[19] Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2024. AI Agents with Formal Security Guarantees. https://openreview.net/forum?id=c6jNHPksiZ

[20] Boomi. 2025. *Boomi Agentstudio | AI Agent Management.* https://boomi.com/platform/agentstudio/

[21] Duc Bui, Brian Tang, and Kang G. Shin. 2023. Detection of Inconsistencies in Privacy Practices of Browser Extensions. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023.* IEEE, 2780–2798. doi:10.1109/SP46215.2023.10179338

[22] Duc Bui, Yuan Yao, Kang G. Shin, Jong-Min Choi, and Junbum Shin. 2021. Consistency Analysis of Data-Usage Purposes in Mobile Apps. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi (Eds.). ACM, 2824–2843. doi:10.1145/3460120.3484536

[23] Jake Chanenson, Madison Pickering, and Noah Apthrope. 2025. Automating Governing Knowledge Commons and Contextual Integrity (GKC-CI) Privacy Policy Annotations with Large Language Models. *Proc. Priv. Enhancing Technol.* 2025, 2 (2025), 280–308. doi:10.56553/POPETS-2025-0062

[24] Hao Cui, Rahmadi Trimananda, Athina Markopoulou, and Scott Jordan. 2023. PoliGraph: Automated Privacy Policy Analysis using Knowledge Graphs. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, Joseph A. Calandrino and Carmela Troncoso (Eds.). USENIX Association, 1037–1054. https://www.usenix.org/conference/usenixsecurity23/presentation/cui

[25] CyberArk. 2025. *agentwatch.* https://github.com/cyberark/agentwatch original-date: 2025-03-26T10:01:05Z.

[26] Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. 2025. AI Agents Under Threat: A Survey of Key Security Challenges and Future Pathways. *Comput. Surveys* 57, 7 (July 2025), 1–36. doi:10.1145/3716628

[27] Embrace The Red. 2023. *Indirect Prompt Injection via YouTube Transcripts · Embrace The Red.* https://embracethered.com/blog/posts/2023/chatgpt-plugin-youtube-indirect-prompt-injection/

[28] CIPT FIP, CIPM and Masha Komnenic CIPP/E. 2025. Privacy Policy Template. https://termly.io/resources/templates/privacy-policy-template/

[29] Ghostery. 2025. *ghostery/trackerdb: Ghostery Tracker Database.* https://github.com/ghostery/trackerdb

[30] Yifeng He, Ethan Wang, Yuyang Rong, Zifei Cheng, and Hao Chen. 2025. Security of AI Agents. In *IEEE/ACM International Workshop on Responsible AI Engineering, RAIE@ICSE 2025, Ottawa, ON, Canada, April 29, 2025.* IEEE, 45–52. doi:10.1109/RAIE66699.2025.00013

[31] Fengqing Jiang, Zhangchen Xu, Luyao Niu, Boxin Wang, Jinyuan Jia, Bo Li, and Radha Poovendran. 2023. Identifying and Mitigating Vulnerabilities in LLM-Integrated Applications. *CoRR* abs/2311.16153 (2023). doi:10.48550/ARXIV.2311.16153 arXiv:2311.16153

[32] David Lee and Mihalis Yannakakis. 1996. Principles and methods of testing finite state machines-a survey. *Proc. IEEE* 84, 8 (1996), 1090–1123. doi:10.1109/5.533956

[33] Patrick Levi and Christoph P. Neumann. 2024. Vocabulary Attack to Hijack Large Language Model Applications. *CoRR* abs/2404.02637 (2024). doi:10.48550/ARXIV.2404.02637 arXiv:2404.02637

[34] Tong Liu, Zizhuang Deng, Guozhu Meng, Yuekang Li, and Kai Chen. 2024. Demystifying RCE Vulnerabilities in LLM-Integrated Apps. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie (Eds.). ACM, 1716–1730. doi:10.1145/3658644.3690338

[35] Microsoft. 2019. *Presidio Demo - a Hugging Face Space by presidio.* https://huggingface.co/spaces/presidio/presidio_demo

[36] Microsoft. 2025. *AutoGen.* https://microsoft.github.io/autogen/stable/index.html

[37] Microsoft. 2025. Home - Microsoft Presidio. https://microsoft.github.io/presidio/

[38] Todd Mytkowicz, Madanlal Musuvathi, and Wolfram Schulte. 2014. Data-parallel finite-state machines. In *Architectural Support for Programming Languages and Operating Systems, ASPLOS 2014, Salt Lake City, UT, USA, March 1-5, 2014*, Rajeev Balasubramonian, Al Davis, and Sarita V. Adve (Eds.). ACM, 529–542. doi:10.1145/2541940.2541988

[39] OpenAI. 2025. *Data controls in the OpenAI platform.* https://platform.openai.com

[40] OpenAI. 2025. *Privacy policy.* https://openai.com/policies/row-privacy-policy/

[41] Rodrigo Pedro, Daniel Castro, Paulo Carreira, and Nuno Santos. 2023. From Prompt Injections to SQL Injection Attacks: How Protected is Your LLM-Integrated Web Application? *CoRR* abs/2308.01990 (2023). doi:10.48550/ARXIV.2308.01990 arXiv:2308.01990

[42] Embrace The Red. 2023. *ChatGPT Plugins: Data Exfiltration via Images & Cross Plugin Request Forgery · Embrace The Red.* https://embracethered.com/blog/posts/2023/chatgpt-webpilot-data-exfil-via-markdown-injection/

[43] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. 2024. Identifying the Risks of LM Agents with an LM-Emulated Sandbox. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.* OpenReview.net. https://openreview.net/forum?id=GEcwtMk1uA

[44] Ahmed Salem, Andrew Paverd, and Boris Köpf. 2023. Maatphor: Automated Variant Analysis for Prompt Injection Attacks. *CoRR* abs/2312.11513 (2023). doi:10.48550/ARXIV.2312.11513 arXiv:2312.11513

[45] Ryoma Sin'ya, Kiminori Matsuzaki, and Masataka Sassa. 2013. Simultaneous Finite Automata: An Efficient Data-Parallel Model for Regular Expression Matching.

In *42nd International Conference on Parallel Processing, ICPP 2013, Lyon, France, October 1-4, 2013.* IEEE Computer Society, 220–229. doi:10.1109/ICPP.2013.31

[46] TAUVOD. 2019. Presidio - Automated identification and anonymization of PII data at scale. https://www.youtube.com/watch?v=1pUEG0MZxvM

[47] AutoGPT Team. 2025. *Platform Privacy Policy.* https://agpt.co/legal/platform-privacy-policy

[48] David Rodríguez Torrado, Ian Yang, José M. del Álamo, and Norman Sadeh. 2024. Large language models: a new approach for privacy policy analysis at scale. *Computing* 106, 12 (2024), 3879–3903. doi:10.1007/S00607-024-01331-9

[49] Wolford and Ben. 2018. Writing a GDPR-compliant privacy notice (template included). https://gdpr.eu/privacy-notice/

[50] Fangzhou Wu, Shutong Wu, Yulong Cao, and Chaowei Xiao. 2024. WIPI: A New Web Threat for LLM-Driven Web Agents. *CoRR* abs/2402.16965 (2024). doi:10.48550/ARXIV.2402.16965 arXiv:2402.16965

[51] Qinge Xie, Karthik Ramakrishnan, and Frank Li. 2025. Evaluating privacy policies under modern privacy laws at scale: an LLM-based automated approach. In *Proceedings of the 34th USENIX Conference on Security Symposium* (Seattle, WA, USA) *(SEC '25).* USENIX Association, USA, Article 298, 20 pages.

[52] Mian Yang, Vijayalakshmi Atluri, Shamik Sural, and Ashish Kundu. 2025. Automated Privacy Policy Analysis Using Large Language Models. In *Data and Applications Security and Privacy XXXIX - 39th IFIP WG 11.3 Annual Conference on Data and Applications Security and Privacy, DBSec 2025, Gjøvik, Norway, June 23-24, 2025, Proceedings (Lecture Notes in Computer Science, Vol. 15722)*, Sokratis K. Katsikas and Basit Shafiq (Eds.). Springer, 23–43. doi:10.1007/978-3-031-96590-6_2

[53] Dongyu Yao, Jianshu Zhang, Ian G. Harris, and Marcel Carlsson. 2023. FuzzLLM: A Novel and Universal Fuzzing Framework for Proactively Discovering Jailbreak Vulnerabilities in Large Language Models. *CoRR* abs/2309.05274 (2023). doi:10.48550/ARXIV.2309.05274 arXiv:2309.05274

[54] Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. 2024. R-Judge: Benchmarking Safety Risk Awareness for LLM Agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 1467–1490. doi:10.18653/V1/2024.FINDINGS-EMNLP.79

[55] Xinyu Zhang, Huiyu Xu, Zhongjie Ba, Zhibo Wang, Yuan Hong, Jian Liu, Zhan Qin, and Kui Ren. 2024. PrivacyAsst: Safeguarding User Privacy in Tool-Using Large Language Model Agents. *IEEE Trans. Dependable Secur. Comput.* 21, 6 (2024), 5242–5258. doi:10.1109/TDSC.2024.3372777

[56] Sebastian Zimmeck and Steven M. Bellovin. 2014. Privee: An Architecture for Automatically Analyzing Web Privacy Policies. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, Kevin Fu and Jaeyeon Jung (Eds.). USENIX Association, 1–16. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/zimmeck

[57] Sebastian Zimmeck, Peter Story, Daniel Smullen, Abhilasha Ravichander, Ziqi Wang, Joel R. Reidenberg, N. Cameron Russell, and Norman M. Sadeh. 2019. MAPS: Scaling Privacy Compliance Analysis to a Million Apps. *Proc. Priv. Enhancing Technol.* 2019, 3 (2019), 66–86. doi:10.2478/POPETS-2019-0037

A Proofs

A.1 Proof of Theorem 1

Proof. Given M independent LLMs, each with an individual accuracy of α (i.e. the probability of correctly extracting a policy element), if there are m LLMs that agree on a specific extraction result, the probability that the result is correct can be computed using Bayes' theorem.

Specifically, let T denote the event that the extraction result is correct, and A_m denote the event that m out of M LLMs agree on the result. We want to compute $\Pr[T|A_m]$, the probability that the extraction is correct given that m LLMs agree on it. By Bayes' theorem, we have:

$$\Pr[T|A_m] = \frac{\Pr[A_m|T]\Pr[T]}{\Pr[A_m]}$$

where:

$$\Pr[A_m|T] = \binom{M}{m}\alpha^m(1-\alpha)^{M-m}, \text{ and } \Pr[T] = \frac{1}{2}.$$

Here, we assume a uniform prior probability for T and $\neg T$, i.e. $\Pr[T] = \Pr[\neg T] = 1/2$. Meanwhile, the total probability of A_m comes from two mutually exclusive cases: when the extraction is correct (T) and when it is incorrect ($\neg T$):

$$\Pr[A_m] = \Pr[A_m|T]\Pr[T] + \Pr[A_m|\neg T]\Pr[\neg T],$$

where:

$$\Pr[A_m|T] = \binom{M}{m}\alpha^m(1-\alpha)^{M-m}.$$

Substituting these expressions into Bayes' theorem, we get:

$$\Pr[T|A_m] = \frac{\binom{M}{m}\alpha^m(1-\alpha)^{M-m}\cdot\frac{1}{2}}{\binom{M}{m}\alpha^m(1-\alpha)^{M-m}\cdot\frac{1}{2} + \binom{M}{m}(1-\alpha)^m\alpha^{M-m}\cdot\frac{1}{2}}.$$

Simplifying this expression, we obtain:

$$\begin{aligned}\Pr[T|A_m] &= \frac{\alpha^m(1-\alpha)^{M-m}}{\alpha^m(1-\alpha)^{M-m} + (1-\alpha)^m\alpha^{M-m}} \\ &= \frac{1}{1+\left(\frac{1-\alpha}{\alpha}\right)^{2m-M}},\end{aligned}$$

which is the same as the expression in Theorem 1. □

A.2 Proof of Theorem 3

Proof. Recall that we assume the privacy policy model P has the forms specified in Theorem 2, i.e.

- Each data type $d^{\text{col}} \in D^{\text{col}}$ is constrained by allowance conditions for collection, processing purposes, and disclosure.
- The collection conditions $C^{\text{col}} = \{\text{direct}, \text{indirect}\}$.
- The processing conditions $C^{\text{pur}} = \{\text{relevant}, \text{irrelevant}\}$.
- The disclosure conditions C^{dis} are the names of third parties.
- The retention conditions C^{ret} are time durations.

We prove that the automata defined in Section 3.3.2 are sound and complete with respect to P.

Soundness: If the automaton $A_{d^{\text{col}}}$ accepts a sequence of runtime annotations, then that sequence satisfies the policy P for data type d^{col}. There are two cases depending on whether disclosure constraints for d^{col} appear in P.

(i) If disclosure constraints exist, $A_{d^{\text{col}}}$ reaches the accepting state dis only after the disclosure conditions are satisfied; prior to that it enforces the collection, purpose, and retention checks. Thus, acceptance implies that all collection, purpose, disclosure, and retention requirements of P are met, or that the data is not collected in the current task but remains within an allowed retention period.

(ii) If no disclosure constraints exist, $A_{d^{\text{col}}}$ reaches the accepting state pur only after the collection, purpose, and retention checks pass, or when the data is not collected in the current task but is within the retention period. Hence, acceptance again implies compliance with P.

Completeness: Conversely, if a sequence of runtime annotations satisfies all conditions of P for d^{col}, then $A_{d^{\text{col}}}$ will accept it. Concretely, when d^{col} is collected during the agent's task and complies with the collection, purpose, disclosure (if applicable), and retention constraints of P, the automaton's transitions are designed to reach the corresponding accepting state (dis when disclosure constraints exist, otherwise pur). Similarly, if the data is not collected but falls within an allowed retention period, the automaton accepts. Therefore, every sequence that satisfies P is accepted by $A_{d^{\text{col}}}$, proving completeness. □