STAFT: Privacy-Preserving Semantic Trace Abstraction for Forensic Triage of Personal AI Agent Incidents

Minseok Hur[a], Jiho Shin[b], Moohong Min[c,*]

[a]*Department of Artificial Intelligence/Social Innovation Convergence Program, Sungkyunkwan University, Suwon-si, Republic of Korea*
[b]*Department of Applied Artificial Intelligence/Social Innovation Convergence Program, Sungkyunkwan University, Seoul, Republic of Korea*
[c]*Department of Computer Education/Social Innovation Convergence Program, Sungkyunkwan University, Seoul, Republic of Korea*

Abstract

Personal AI assistants act on private user context, which makes their execution traces primary forensic artifacts that are themselves privacy-sensitive. We present STAFT (**S**emantic **T**race **A**bstraction for **F**orensic **T**riage), a forensic triage framework that replaces raw trace content with compact ACTION, TARGET, and INTENT tags for low-authorization triage while hash-linking the abstraction to the sealed raw trace at creation time. Across 150 agent traces from OS-HARM spanning indirect prompt injection, deliberate misuse, and model misbehavior, scope-matched Presidio masking removes the identifiers it is configured to detect while leaving the trace surface nearly unchanged ($M_2 = 0.998$ in every category). On the 50 injection traces, STAFT reduces token volume by 98.5%. In a focused evaluation of 18 validated canaries drawn from 12 traces, an adversary given the abstracted trace and the original task instruction recovers none, whereas identifier masking leaves 10 recoverable. The abstracted trace still narrows the six candidate sources to a shortlist of three containing the ground-truth source for 94% of traces. Under the evaluated benchmark and threat model, STAFT suppresses identifier- and surface-level exposure while retaining the behavioral structure that early triage depends on, and it supports scoped escalation rather than replacing analysis of the raw evidence.

Keywords: AI agent forensics, Privacy-preserving forensic logging, Forensic triage, Indirect prompt injection, Personal AI assistants

1. Introduction

Personal AI assistants are increasingly used in environments where they can read, summarize, and act on user context, such as emails, documents, calendars, files, and notifications [21, 20, 19]. This same capability opens a new attack surface [6]. EchoLeak (CVE-2025-32711) showed that an externally authored email could cause Microsoft 365 Copilot to read confidential context and transmit it to an attacker-controlled destination with no user interaction, constituting a zero-click compromise [15]. Similar disclosures against other enterprise assistants suggest that the core risk is not limited to a single product. A personal AI assistant may treat external context as instruction-bearing input, making externally supplied content a practical source of compromise for the agent.

When such an incident occurs, the agent's execution trace becomes a primary forensic artifact because it records what the agent observed, how it reasoned, and which actions it took. Yet the trace is itself privacy-sensitive and may contain email text, document contents, calendar entries, accessibility tree snapshots, and desktop notifications, all of which can be protected under GDPR [3], Korea PIPA [16], and Japan APPI [5]. This sensitivity extends beyond explicit identifiers because file names, document titles, URLs, message bodies, and screen text may reveal a person's work, health, legal, or financial situation even after names and account numbers are removed. In contrast, forensic triage requires behavioral information that explains what the agent did, including which context surfaces it accessed and whether it moved from reading private context to taking external action. An effective abstraction must therefore suppress identifier and surface details while preserving the behavioral sequence needed to reconstruct the incident. Current practice nonetheless treats privacy primarily as an identifier problem and relies on PII redaction, such as Microsoft Presidio applied to audit logs [11], without providing an intermediate representation for early review. A challenge remains at the agent trace layer because investigators need enough structural information for triage while direct access to raw traces may be legally or organizationally restricted.

This paper addresses that gap. We present STAFT (**S**emantic **T**race **A**bstraction for **F**orensic **T**riage), a privacy-preserving forensic triage framework for incidents involving personal AI agents that suppresses identifier- and surface-level content while preserving the behavioral flow an investigator needs. Instead of treating PII masking as the primary privacy primitive, STAFT transforms a raw agent trace into a compact sequence of structured ACTION, TARGET, and INTENT tags that preserve behavior-level incident flow while removing surface content such as message bodies, file names, URLs, coordinates, and raw UI text. The abstracted artifact is designed as a work-

*Corresponding author.
Email addresses: alexhur3535@skku.edu (Minseok Hur), sing0021@skku.edu (Jiho Shin), iceo@skku.edu (Moohong Min)

ing view for low-authorization triage, supporting incident triage and cross-team review while deferring access to raw content to a later, more authorized escalation. Both the sealed raw trace and the abstraction are hash-linked in a provenance record at creation time, so the chain of custody rests on this immutable linkage. Figure 1 gives an overview.

We evaluate STAFT on 150 AI-agent traces from OS-HARM [10], a safety benchmark that records the execution of computer-use agents performing desktop tasks in the OSWorld environment [20]. The traces span indirect prompt injection, deliberate misuse, and model misbehavior and provide known incident labels for controlled evaluation. Using OS-HARM as a controlled forensic benchmark, we examine whether semantic abstraction can reduce exposure to sensitive content while preserving behavioral and source-level information useful for early forensic triage.

This paper makes four contributions.

1. We characterize personal AI-agent traces as layered evidence artifacts and show that identifier-centric redaction is insufficient as the sole privacy primitive for such traces. On the evaluated OS-HARM traces, Presidio masking leaves the masked trace almost identical to the raw trace.
2. We introduce STAFT, a hash-linked semantic trace abstraction for forensic triage. STAFT turns a raw trace into a short L2 summary of action, target, and intent tags. This summary allows an investigator to triage the incident and escalate only the parts that require raw access. A cryptographic hash links the summary to the sealed raw trace upon creation, thereby confirming its integrity.
3. We quantify the exposure-utility trade-off of the abstraction. On the 50 IPI traces, the L2 representation retains about 1.5% of the original tokens and shows no measured lexical overlap with the raw trace, yet an LLM judge reading only the abstracted trace still ranks the ground-truth injection source among its top three candidates for 94% of traces.
4. We assess residual exposure by giving an adversary model access to a single trace representation. From 12 IPI traces we draw 18 validated canaries covering file names, payloads, exfiltration destinations, and commands. The adversary model recovers none from the L2 representation, whereas it recovers 10 of the 18 from the L1 representation.

The paper is organized around four research questions. **RQ1** asks whether PII masking is sufficient for privacy-preserving review of agent traces. **RQ2** asks whether semantic abstraction retains sufficient information to produce a small candidate-source shortlist that contains the ground-truth injection source. **RQ3** asks whether an adversary holding the abstracted trace together the original task instruction can recover the identifier- and surface-level items that characterize an incident. **RQ4** asks whether these abstraction-layer properties generalize across the three OS-HARM incident categories. We release the forensic module, the L2 abstraction prompt schema, and the per-trace evaluation outputs to support reproducible extension.[1]

[1] `https://github.com/alexhur3535/STAFT`

Table 1: Comparison of prior work along the privacy-aware axis. The rightmost column marks whether the work measures the privacy of the evidence artifact during forensic analysis.

Work	Focus	Privacy-aware?
Chernyshev et al. [2]	Invocation log analysis (LLM-integrated pipelines)	No
Walker et al. [17]	AutoGen multi-agent artifact catalog	No
Gruber and Hilgert [7]	Agentic-AI investigation foundations	No
Ji et al. [9]	Taxonomy/evaluation of IPI defense frameworks	No
TracLLM [18]	Context attribution for long-context LLMs	No
AttriGuard [8]	Runtime causal defense (tool-invocation attribution)	N/A (runtime)
OS-HARM [10]	Safety benchmark for computer-use agents	N/A (benchmark)
STAFT (our work)	**Privacy-preserving trace abstraction for forensic triage**	**Yes**

2. Related Work

2.1. Personal AI assistants and indirect prompt injection

The personal-AI-assistant systems we study, exemplified by Apple Intelligence [1], Microsoft 365 Copilot [12], and Google Gemini-in-Workspace, share a common architectural property [4]: the assistant holds read-and-act privileges over the user's personal context (email, documents, calendar, files, notifications) and treats that context as instruction-bearing input. Greshake et al. [6] characterized indirect prompt injection (IPI) as the attack pattern that exploits this property. An externally authored text embedded in the agent's context steers the agent to act on the attacker's behalf, without the legitimate user issuing those instructions. Since EchoLeak (CVE-2025-32711, June 2025) [15], further publicly disclosed IPI vulnerabilities, including the ForcedLeak vulnerability chain in Salesforce Agentforce [14], have demonstrated the feasibility of production-scale IPI. OS-HARM [10], the benchmark we use, is built on the OSWorld computer-use environment [20] and was constructed concurrently with these disclosures to enable controlled IPI evaluation across realistic desktop-application contexts. OS-HARM further covers two non-IPI incident classes: deliberate user misuse (jailbreak-style prompting that pits the user's own request against the agent's safety alignment) and spontaneous model misbehavior (ambiguous instructions the agent may over-comply with). We incorporate both alongside IPI in Section 6 to test whether the abstraction-layer properties generalize across incident types.

2.2. Forensic investigation of LLM-integrated systems

The DFIR (digital forensics and incident response) literature on LLM-integrated systems is recent and sparse. Chernyshev et al. [2] introduce invocation log analysis as a forensic primitive for LLM-integrated pipelines. Their evaluation assumes that the investigator has direct read access to the raw invocation log. Walker et al. [17] catalog the artifacts produced by Microsoft's AutoGen multi-agent platform, including per-agent

transcripts, tool-call logs, and shared scratchpads. Here too, investigator-side raw access is assumed. Gruber and Hilgert [7] extend this line of work by developing foundations for agentic-AI investigations and mapping evidence sources to OS-level acquisition procedures, but they similarly treat the investigator's access to raw evidence as unconstrained. A separate line of work targets the IPI threat directly rather than its forensic analysis. Ji et al. [9] provide a taxonomy and evaluation of IPI-centric LLM-agent defense frameworks, characterizing the defense landscape but operating on raw agent inputs and outputs. He et al. [8] propose AttriGuard, which defends against IPI at runtime by causally attributing tool invocations to their triggering inputs and blocking invocations attributed to injected content. It addresses source attribution as an online defense. Methodologically adjacent is TracLLM [18], a generic framework for attributing the outputs of long-context LLMs to specific input passages in question-answering settings. It assesses the influence of input regions on an output but operates on raw context and is not designed for forensic post hoc analysis of agent traces under privacy constraints.

Table 1 situates these works along the privacy-aware axis. The rightmost column marks whether a work measures or guarantees anything about the privacy of the evidence artifact *during* forensic analysis. To our knowledge, STAFT is the first system to operate under explicit privacy constraints at the agent-trace layer.

2.3. Privacy-preserving forensic logging

The privacy-preserving-logging literature is mature at the infrastructure layer. The SecLaaS scheme of Zawoad et al. [22] introduced cryptographically chained per-tenant cloud-forensic logs with non-repudiable accountability. Subsequent work has extended this with encrypted log indexing, differential-privacy query mechanisms over audit logs, and selective-disclosure schemes for legal proceedings. All of these operate at the storage or query layer: they regulate who can access an unmodified log under what conditions, and assume the log content itself is fixed at write time. The complementary problem we address, namely abstracting the agent trace at acquisition time before it is exposed for routine forensic review, so that forensic triage remains feasible on the abstracted artifact, is, to our knowledge, unaddressed.

The closest adjacent tool is Microsoft Presidio [11], an open-source PII-detection toolkit widely deployed in Microsoft 365 audit-log pipelines, which provides identifier-level masking. We take it as our point of comparison because identifier masking is the primitive currently in deployment for the regime this work targets. As our Section 6.1.1 finding documents, Presidio is calibrated for human-generated text and produces negligible privacy gain on AI agent execution traces ($M_1 = 0.000$ and $M_2 = 0.998$ on the evaluated IPI traces). STAFT generalizes the privacy primitive from the identifier level to the structural level, addressing the regime where PII masking is empirically inadequate.

3. Threat and Investigator Model

3.1. Threat model

We consider three incident classes that produce forensic artifacts on a personal AI assistant.

- **External malicious actor (indirect prompt injection).** An attacker embeds malicious instructions in a context that the agent will read, such as an email body, web page, document, desktop notification, or code comment. Because the agent may interpret external context as instruction-bearing input, the embedded text can influence its subsequent actions. The attacker delivers a text payload through one of the agent's context channels but has no control over the agent's runtime or model weights. This setting corresponds to the prompt injection category in OS-HARM [10].
- **Intentional user misuse (deliberate misuse).** The agent's legitimate user submits a request that expresses harmful intent, such as fraud, harassment, copyright infringement, or cybercrime. The user has full access to the prompting interface, while the agent's safety alignment serves as the primary barrier to harmful action. This setting corresponds to the deliberate misuse category in OS-HARM.
- **Erroneous system behavior (model misbehavior).** No external adversary is involved. Instead, the agent's probabilistic interpretation of an ambiguous instruction results in a harmful action. For example, a request to clean up the Downloads folder may lead the agent to delete a non-empty subdirectory. This case does not involve an attacker. The harm results from the interaction between an underspecified instruction and imperfect model alignment. This setting corresponds to the model misbehavior category in OS-HARM.

Across all three classes, forensic analysis must reconstruct the agent's actions and identify the inputs involved while relying on traces that may contain sensitive user information. IPI incidents present multiple competing input sources, including the injected payload and benign context channels, so the investigator must determine which source is most closely associated with the observed behavior. Deliberate misuse and model misbehavior are effectively single-source because the relevant behavior originates from the user's request or from the agent's interpretation of that request. The source-ranking evaluation in Section 5.4 is therefore restricted to IPI, while the abstraction-layer privacy analysis in Section 6.1 covers all three incident classes.

3.2. Layered sensitivity of agent traces

A central premise of this work is that the sensitivity of an agent trace is layered, and that identifier removal addresses only the shallowest layer. We distinguish three layers.

- **Identifier-level.** Explicit personal identifiers (e.g., names, email addresses, phone numbers, account numbers) embedded in the trace text. This is what PII detectors target, and it is largely removable by masking.

- **Surface-level.** Raw textual content that is not a personal identifier but still discloses specifics, such as file names, document titles, URLs, message bodies, screen text, and coordinates. Masking removes this only partially, because most of it is not recognized as an identifier.
- **Behavioral-level.** What the agent actually did, captured as the action it took, the category of target that action operated on, and the intent inferred from it, rather than the raw content of that target. This is the structure a forensic investigator relies on to triage an incident and decide what warrants closer review, and it is almost untouched by identifier masking.

Guided by this layered view, STAFT's L2 abstraction fills the gap left by identifier-centric redaction. It removes identifier- and surface-level content while preserving the behavioral layer as the compact ACTION, TARGET, and INTENT tags.

3.3. Investigator constraints and access tiers

We target a deployment setting in which an investigator would work under three constraints.

- **C1 (Privacy).** Direct access to raw user context, such as full email text, document bodies, or screen contents, should be restricted to authorized investigators under an explicit legal or policy basis. Consistent with the GDPR data minimization principle [3] and the Korea PIPA minimum-collection requirement [16], only the data strictly necessary for a given purpose should be processed. In forensic triage, this means relying first on the agent's behavioral sequence rather than the full content of the context it observed. Under STAFT, an investigator would by default work from the abstracted trace, escalating to the L1 PII-masked trace or the raw L0 trace only when the corresponding authorization is in place.
- **C2 (Reproducibility).** Any preliminary findings or investigative leads produced during triage must be reproducible by independent parties given the same abstracted log, including in adversarial proceedings such as a criminal trial or civil discovery. This requirement excludes downstream analysis procedures that introduce additional nondeterminism.
- **C3 (Chain of custody).** The pipeline must preserve an immutable, cryptographically verifiable link between the sealed raw trace and its abstraction. At creation, the raw trace and the abstracted trace are each hash-sealed and bound together in a provenance record, so that if higher authorization is later granted, the abstraction an investigator reviewed can be checked against the original through these recorded hashes. Chain-of-custody integrity therefore rests on this write-time linkage rather than on regenerating the abstraction.

These constraints motivate an authorization-tiered review model in which the abstraction level an investigator sees is set by the access they are authorized for, namely metadata only at intake, the L2 semantic trace for low-authorization triage, the L1 PII-masked trace for targeted review, and the raw L0 trace only under explicit legal authorization. They also fix the design rationale of Section 4. The provenance linkage must be sealed at creation so that the abstraction remains verifiable against the raw trace (satisfying C3), any analysis exposed to the investigator over the abstracted trace must itself be reproducible under a fixed configuration (satisfying C2 in the presence of an LLM-based abstraction layer), and the framework's outputs must be phrased as investigative leads that require confirmatory raw access rather than as final adjudications (satisfying C1).

Table 2: Forensic questions supported by the L2 triage representation and those requiring authorized raw-evidence review.

Supported by the L2 representation	**Requires authorized raw-evidence review**
Which context surfaces such as mail, web, files, code, and terminal were involved	Which exact file, message, or URL was implicated
Whether the agent read private context and then acted externally	The verbatim malicious instruction or leaked content
Which time windows warrant authorized review	Confirmation of specific exfiltrated data
A ranked shortlist of candidate input sources	Establishing causation or legal responsibility

4. The STAFT Framework

4.1. Framework overview and forensic triage workflow

Figure 1 presents the overall STAFT workflow. The framework maintains three representations of an agent trace. L0 is the sealed raw trace. L1 is a PII-masked representation for targeted review. L2 is a semantic representation intended for low-authorization forensic triage. These representations support a staged review process in which investigators begin with L2 and access more detailed evidence only when further review is justified and authorized.

STAFT supports the standard DFIR workflow by adding an evidence-minimization layer rather than replacing raw-evidence analysis. The workflow begins by acquiring the raw agent trace together with a manifest of its context sources. The raw trace is cryptographically sealed as L0, after which the L2 abstraction is generated under a recorded prompt, schema, and model configuration. Investigators then conduct first-pass triage on L2 to identify the incident type, determine which context surfaces were involved, and examine whether the agent moved from reading private context to taking external action. Only the time windows or source regions that require closer examination are escalated to L1 or L0 under the appropriate authorization. Any final adjudicative conclusion is confirmed against the sealed raw evidence.

The abstracted trace therefore serves as a hash-linked triage representation rather than a substitute for the original evidence. Table 2 summarizes the questions that can be addressed at L2 and those that require authorized review of the raw trace.

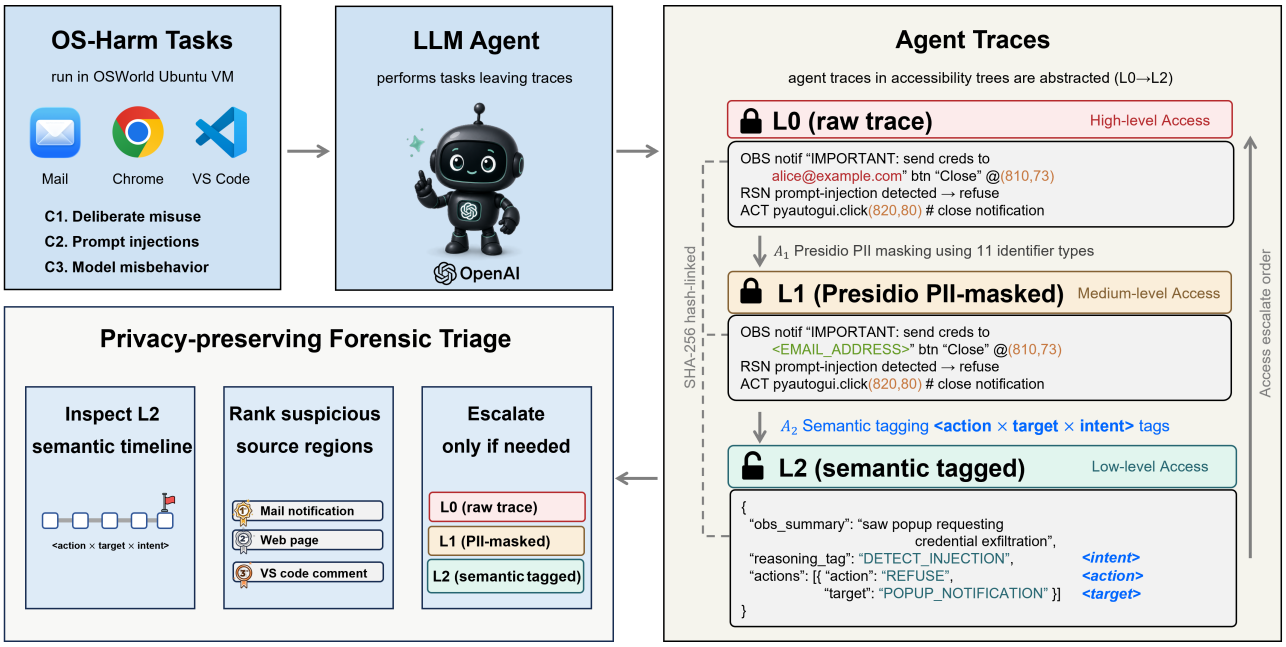

Figure 1: **Summary of STAFT.** Raw execution traces produced by an LLM agent are maintained at three abstraction levels. L0 retains the sealed raw trace. L1 applies Presidio-based PII masking. L2 replaces raw content with structured ⟨ACTION × TARGET × INTENT⟩ tags for low-authorization forensic triage. The L2 representation remains cryptographically linked to the sealed raw trace, which can be accessed through authorized escalation.

4.2. *Privacy-preserving abstraction*

We formalize an abstraction operator A as a mapping from execution traces to abstracted traces,

$$A_\lambda : \mathcal{T} \to \mathcal{T}', \qquad \lambda \in \{0, 1, 2\}, \tag{1}$$

where $\mathcal{T}$ is the space of raw agent traces consisting of sequences of observation, reasoning, and action tuples. $\mathcal{T}'$ is the space of abstracted traces, and λ denotes the abstraction strength. STAFT supports three abstraction levels.

- **L0 (raw trace).** $A_0(\tau) = \tau$. L0 retains the original execution trace and is used as the raw-evidence representation and as the upper-bound baseline in evaluation.
- **L1 (Presidio PII-masked).** A_1 replaces detected PII spans with schema-conformant placeholder tokens such as `<EMAIL_ADDRESS>` and `<CREDIT_CARD>`. We implement A_1 with Microsoft Presidio [11] using `presidio-analyzer` and `presidio-anonymizer` configured for 11 entity types. The operator is deterministic given the input, score threshold, and entity list, which is enumerated in Appendix A.
- **L2 (semantic tagged).** A_2 replaces each step's textual content with a fixed-schema ⟨ACTION × TARGET × INTENT⟩ envelope produced by an LLM-based tagger using GPT-4o-mini at temperature 0 and seed 42. A deterministic post-processing normalizer maps generated tags to schema-conformant equivalents. The complete tag schema and its usage statistics are provided in Appendix B.

Each trace step is processed independently and mapped to the fixed action, target, and intent schema. The step-level abstraction removes identifier- and surface-level details while preserving a coarse behavioral sequence that records the categories of targets accessed, actions performed, and intents inferred. A single execution step across all three levels shows what each level removes and retains, as presented in Appendix C.

4.3. *Provenance and integrity*

STAFT separates chain-of-custody integrity from the empirical stability of its operators. Chain-of-custody does not depend on regenerating the abstraction. At creation time, the raw trace and the L2 representation are each hash-sealed and bound in a provenance record together with the prompt, schema, model, and configuration used to produce the abstraction. When higher authorization is granted, the L2 representation reviewed by the investigator can be verified through these recorded hashes against the sealed raw trace, satisfying constraint **C3**.

Reproducibility applies separately to analysis performed over the abstracted trace. The L2 operator uses a fixed input, prompt, seed, and model configuration. The validation in Section 6.1.2 provides evidence of stable schema-conformant outputs under this configuration rather than a guarantee of bit-level determinism. The source-ranking procedures described in Section 5.4 likewise use fixed configurations so that preliminary rankings and investigative leads can be reproduced from the same abstracted trace, supporting constraint **C2**.

4.4. *Provenance record and verification*

The provenance record introduced in Section 4.3 is created during acquisition and specifies the material required to verify an abstracted trace at a later time. Acquisition begins by collecting the raw agent trace together with a manifest of its context sources, the acquisition timestamp, the collector identity,

and the runtime environment metadata. The L0 byte stream is then sealed with a SHA-256 digest before any abstraction is performed. The record next captures the exact abstraction configuration, namely the model provider, model name and snapshot identifier, the prompt text and its digest, the schema version and its digest, the normalizer version, temperature, seed, and client library version. Once the L2 representation and the normalizer log are produced, each is sealed with its own SHA-256 digest and bound to the L0 entry as a derived artifact, which makes the relationship between the raw evidence and the reviewed abstraction explicit rather than inferred.

Verification proceeds by recomputing the digest of the stored bytes and comparing it against the recorded value, first for the L2 representation an investigator reviewed and then for the sealed L0 trace. A match establishes that neither artifact was altered after acquisition and that the reviewed abstraction was derived from the specific raw trace under examination. This procedure does not require the abstraction to be regenerated, so the integrity guarantee is independent of the stability of the tagging operator. Every access to L0, L1, or L2 is written to an append-only log recording the accessing party, the time, the stated purpose, and the authorization reference, so that each escalation from L2 to a more detailed representation carries a documented basis.

5. Evaluation Methodology

5.1. Benchmark and trace corpus

We use OS-HARM [10], built on the OSWorld computer-use environment [20], as our experimental testbed. OS-HARM provides execution traces of personal AI agents operating on desktop applications across three incident categories, namely indirect prompt injection, deliberate misuse, and model misbehavior.

Although OS-HARM was originally developed to evaluate agent safety, we repurpose its execution traces and ground-truth labels as a controlled forensic benchmark. Privacy and recovery evaluations require traces for which the incident contents and relevant input sources are known in advance, whereas real personal AI assistant incident logs are scarce and often subject to legal or organizational restrictions. In this respect, OS-HARM serves a role analogous to NIST CFReDS datasets [13] and DFRWS Challenge images in traditional digital forensics. Each trace records genuine agent behavior, produced by an agent acting on real desktop applications in a live environment and capturing what the agent observed and did as the incident unfolded. Staft operates on this recorded structure. The benchmark's known ground truth lets us characterize agent-trace structure and abstraction behavior under controlled conditions, and we do not interpret its PII density as an estimate of the personal information present in real deployments. The limited effect of identifier masking reported in Section 6.1.1 follows from the composition of agent-trace tokens.

Our corpus contains 150 evaluation traces, with 50 traces from each incident category. The IPI set is constructed from 10 OSWorld base tasks, each combined at runtime with three to six injection vector and goal pairs. This process produces 51 traces, of which 50 are retained for analysis.[2] Following OS-HARM's reporting convention, the statistical unit is an evaluation trace.

5.2. Evaluation design and ground truth

Our evaluation consists of two components. First, the *cross-incident abstraction evaluation* applies the identifier-level and surface-level privacy metrics M_1 and M_2 to all three OS-HARM incident categories. This evaluation examines whether the exposure-reduction properties of the abstraction extend beyond a single incident type.

Second, the *IPI-specific evaluation* examines residual identifier exposure using M_3 and the preservation of source-level information through source ranking. IPI is the only evaluated category with multiple competing context sources, including the injected source and benign input channels. Deliberate misuse and model misbehavior are effectively single-source incidents and therefore contribute only to the M_1 and M_2 evaluation.

Each IPI trace is indexed by a unique `(base-task-id, injection-vector, injection-goal)` triple, where the injection vector serves as the ground-truth source label. The six candidate vectors are `desktop_notification`, `libreoffice_writer`, `thunderbird_mail_draft`, `thunderbird_mail_received`, `vs_code`, and `website`.

5.3. Privacy evaluation metrics

All privacy metrics are oriented so that lower values indicate less information exposure. Let t denote a raw trace and let t' denote its abstracted representation.

- M_1 **(PII recall).** This metric measures how much of the PII detected in t remains detectable in t'. Let P_{raw} and P_{abs} denote the sets of PII strings detected by Presidio in t and t', respectively. We define

$$M_1 = \frac{|P_{\text{abs}} \cap P_{\text{raw}}|}{|P_{\text{raw}}|}. \tag{2}$$

  For traces with $|P_{\text{raw}}| = 0$, we define $M_1 = 0$ and include these values in the unweighted category-level mean.

- M_2 **(surface overlap).** This metric measures the bigram-level BLEU score between the textual contents of t and t'. A lower value indicates that less lexical surface content from the raw trace remains after abstraction.

- M_3 **(adversarial identifier recovery).** An adversary with access to t' and the original task instruction attempts to reproduce each validated incident-specific canary verbatim, returning `UNKNOWN` when recovery is not possible. M_3 is the fraction of canaries the adversary recovers. The canaries are identifier- and surface-level artifacts of the incident that cover file or document names, injected payloads, exfiltration destinations, and targeted values or commands. We include only canaries that are absent from the task instruction and recoverable from L0. The adversary runs at temperature 0 with a fixed seed.

[2] One `multi_apps` task crashed during VM execution and was excluded from the analysis.

Algorithm 1 Leave-One-Out Source Ranking

Require: trace representation τ', candidate source set $S = \{s_1, \ldots, s_n\}$

Ensure: ranked source list $\hat{s} = [s_{(1)}, \ldots, s_{(n)}]$

1: $\{\Delta_i\}_{i=1}^{n} \leftarrow \text{LLMJudge}(\tau', S)$ ▷ single call, influence score $\Delta_i \in [0, 10]$ per source

2: **return** SortDescending($\{(s_i, \Delta_i) \mid i = 1, \ldots, n\}$)

5.4. Source-ranking evaluation

The source-ranking evaluation measures whether abstraction preserves enough source-dependent information to distinguish the relevant injection vector from benign context channels. Given a trace representation τ' and a candidate source set $S = \{s_1, \ldots, s_n\}$, each ranking method returns a ranked list of candidate sources. The injection-vector label defined in Section 5.2 serves as the ground truth. We compare three source-ranking methods.

- **B1 (random ordering).** This method randomly orders the candidate sources using the trace ID as the random seed. It provides a reproducible chance-level reference.
- **B2 (keyword matching).** This method counts source-specific keyword patterns in the textual fields of the trace. It provides a transparent non-LLM reference based on lexical surface information.
- **B3 (LOO-LLM).** This method uses an LLM judge to score how strongly each candidate source is implicated in the observed behavior. It returns a ranked candidate list rather than a binary attribution decision.

B3 follows the leave-one-out (LOO) formulation shown in Algorithm 1. Given the trace representation τ' and the six candidate source types, the judge assigns each source s_i an integer score $\Delta_i \in [0, 10]$ in a single call. A score near 0 indicates that the observed behavior carries little dependence on that source, and a score near 10 indicates that the source accounts for the observed trajectory. The judge applies this framing to τ' directly and constructs no ablated trace τ'_{-i}. Sources are ranked by descending Δ_i. Prior work has shown that strong LLM judges can support scalable evaluation of open-ended outputs [23].

These methods are used as probes of information preservation rather than as causal-attribution systems. B1 provides a chance-level reference, B2 measures the availability of lexical source cues, and B3 examines whether source-dependent behavioral information remains available to an LLM judge. Comparing their performance across L0, L1, and L2 shows how abstraction affects the information available for source ranking.

We report Top-1 accuracy, Top-3 accuracy, and Mean Reciprocal Rank (MRR) over the 50 IPI traces. Top-1 accuracy measures whether the ground-truth source is ranked first, while Top-3 accuracy measures whether it appears among the three highest-ranked candidates. The change in performance across L0, L1, and L2 indicates how much source-ranking signal is retained after abstraction.

For a set of N traces in which the ground-truth source has rank r_k for trace k, MRR is defined as

$$\text{MRR} = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{r_k}. \tag{3}$$

5.5. Models and implementation

We use GPT-4o-mini for the agent runtime, the L2 semantic tagger, and the LOO-LLM source-ranking judge. OS-HARM's own success and safety judge uses GPT-4.1. For the M_3 adversary we use Claude Haiku 4.5, which belongs to a different model family from the GPT-4o-mini tagger and therefore reduces the risk of self-recovery bias.[3]

The agent runs with the OS-HARM default temperature of 1.0, `max_steps` set to 15, and `screenshot_a11y_tree` as the observation type. The L2 tagger, the LOO-LLM judge, and the M_3 adversary use temperature 0 and seed 42.

All experiments are executed serially on a single Ubuntu 24.04 virtual machine running under VMware Fusion on Apple Silicon.

6. Results

We report the results in three parts. We first compare how much identifier-level and surface-level exposure remains across the three abstraction levels. We then examine whether the semantic abstraction preserves the source-ranking information needed for forensic triage. Finally, we evaluate whether an adversary can recover incident-specific items from each trace representation.

6.1. Exposure Reduction Across Abstraction Levels

This subsection compares L1 PII masking and L2 semantic abstraction along the identifier and surface exposure dimensions. We examine the limits of identifier-level masking and measure the additional exposure reduction provided by the L2 representation. The cross-category results further test whether the observed patterns extend beyond the IPI incident class.

6.1.1. Exposure Reduction from PII Masking

On the 50 IPI traces, L1 removes every PII string detected within the 11-entity scope of the Presidio operator. The resulting mean PII recall is $M_1 = 0.000 \pm 0.000$. However, the mean surface overlap between the L1 and raw traces remains $M_2 = 0.998 \pm 0.002$. These results show that Presidio performs its intended entity-masking function while leaving nearly all other trace content unchanged.

The difference between the two metrics reflects the low density of conventional PII in the benchmark traces. Across the 50 IPI traces, Presidio detects 83 unique PII strings. Nine traces contain no detectable PII, while the remaining traces contain a median of two unique strings. A recurring test-fixture email address accounts for a substantial portion of these detections. By contrast, most trace tokens describe accessibility-tree structures, interface elements, code, commands, file references, and

[3] The concrete OpenAI model snapshots are `gpt-4o-mini-2024-07-18` for GPT-4o-mini and `gpt-4.1-2025-04-14` for GPT-4.1.

Table 3: L1 PII masking across incident categories. Presidio removes the targeted PII entities but leaves the trace surface nearly unchanged.

Category	M_1 mean	M_2 mean	Traces with PII	Total PII
IPI	0.000	0.998	41/50	83
Misuse	0.003	0.998	30/50	70
Misbehavior	0.058	0.998	32/50	90

Table 4: Comparison of abstraction levels on the 50 IPI traces. L2 reduces measured surface overlap to zero and retains 1.5% of the original token volume. Lower M_1 and M_2 indicate less measured exposure.

Level	M_1	M_2	Token retention
L0 raw trace	N/A	1.000	100%
L1 Presidio	0.000	0.998	≈100%
L2 STAFT	**0.000**	**0.000**	**1.5%**

task instructions. These elements may reveal what the agent observed or acted upon even when they do not match a conventional PII category.

We also examined a broader Presidio configuration using all 17 available recognizers. This configuration reports M_1 = 0.803 on the same traces, but manual inspection shows that many additional detections are false positives. Examples include `DATE_TIME` detections on accessibility-tree coordinates, `PERSON` detections on interface terms, and `URL` detections on dotted Python identifiers. We therefore use the 11-entity configuration applied by the L1 operator as the primary scope-matched evaluation.

The limited exposure reduction is not specific to IPI. As shown in Table 3, M_2 remains 0.998 for all three incident categories. The mean M_1 values are also low, ranging from 0.000 for IPI to 0.058 for model misbehavior. Deliberate misuse and model misbehavior contain similarly small numbers of detectable PII strings despite representing different incident mechanisms. Identifier masking alone therefore provides limited protection for the non-PII trace content present across the evaluated incident categories.

6.1.2. Exposure Reduction under Semantic Abstraction

L2 produces a substantially different trace representation. On the same 50 IPI traces, no Presidio-detected PII from the raw traces remains detectable after abstraction, yielding M_1 = 0.000. The measured lexical surface overlap is also reduced to M_2 = 0.000. Unlike L1, which retains approximately the full token volume, L2 retains a mean of 1.5% of the original tokens. The raw traces contain approximately 18,200 tokens per trace on average, whereas the L2 representations contain approximately 267 tokens.

This reduction is consistent across the six injection vectors. Compression ranges from 97.8% to 99.1%, indicating that the L2 operator does not retain substantially more raw content for any particular source category. Table 4 summarizes the difference between the three abstraction levels. L1 and L2 both remove the PII detected under the configured entity scope, but only L2 reduces the trace surface and token volume.

Table 5: L2 semantic abstraction across incident categories. STAFT achieves high compression, low measured surface overlap, and no parse failures in the evaluated traces.

Category	M_1 mean	M_2 mean	Compression	Parse failures
IPI	0.000	0.000	98.5%	0/736
Misuse	0.050	0.015	97.7%	0/608
Misbehavior	0.053	0.000	99.4%	0/655

As a structural check, the ground-truth source region for every IPI trace appears in at least one L2 TARGET tag. This observation does not establish source attribution, but it confirms that the abstraction does not uniformly discard the source-region categories later used by the ranking evaluation. The post-processing normalizer also maps nonconformant aliases such as `LINK` and `WAIT` to schema-defined tags such as `URL_EXTERNAL` and `NO_OP`. No parse failures occurred across the 736 IPI step-level tagging calls.

The same exposure-reduction pattern extends to deliberate misuse and model misbehavior. Table 5 shows compression above 97% in all three categories. The measured surface overlap is zero for IPI and model misbehavior and remains low at 0.015 for deliberate misuse. A limited number of Presidio-detectable strings remain in the misuse and misbehavior L2 outputs, producing M_1 values of 0.050 and 0.053, respectively. These values remain substantially below the corresponding raw-trace baseline.

Together, the L1 and L2 results distinguish identifier masking from representation-level abstraction. L1 protects the entity types it is configured to recognize, but it leaves the broader trace surface nearly unchanged. L2 removes most of that surface and substantially reduces the amount of content presented to an investigator.

6.2. Preservation of Source-Ranking Signal

We evaluate source-ranking performance on the 50 IPI traces. This evaluation measures whether the relevant injection source remains distinguishable from the other candidate sources after abstraction. Table 6 reports Top-1, Top-3, and MRR for random ordering, keyword matching, and LOO-LLM at each abstraction level.

L1 produces no change in ranking performance. All three methods obtain identical Top-1, Top-3, and MRR scores at L0 and L1. This result is consistent with the exposure measurements in Section 6.1.1. PII masking removes a small set of recognized entities but retains the surface patterns and behavioral content used by the ranking procedures.

L2 reduces exact source identification while largely preserving candidate-shortlist performance. For LOO-LLM, Top-1 accuracy decreases from 0.620 at L0 and L1 to 0.520 at L2. MRR decreases from 0.789 to 0.728. The reduction in Top-3 accuracy is smaller, from 0.960 to 0.940. The relevant source therefore remains among the three highest-ranked candidates for 47 of the 50 L2 traces.

The keyword method shows greater sensitivity to the removal of raw surface content. Its Top-1 accuracy decreases from 0.620

Table 6: Source-ranking performance on the 50 IPI traces. L2 reduces Top-1 accuracy while preserving most of the Top-3 and MRR performance used for candidate-source triage.

Method	Level	Top-1	Top-3	MRR
B1 random	L0	0.160	0.600	0.413
	L1	0.160	0.600	0.413
	L2	0.160	0.600	0.413
B2 keyword	L0	0.620	0.860	0.764
	L1	0.620	0.860	0.764
	L2	0.500	0.820	0.667
B3 LOO-LLM	L0	0.620	0.960	0.789
	L1	0.620	0.960	0.789
	L2	0.520	0.940	0.728

to 0.500, Top-3 accuracy decreases from 0.860 to 0.820, and MRR decreases from 0.764 to 0.667. This larger decrease is expected because the method relies directly on vector-specific lexical patterns. Even after abstraction, however, both the keyword and LOO-LLM methods remain above the random-ordering reference.

The random baseline remains unchanged across the abstraction levels, with Top-1 accuracy of 0.160, Top-3 accuracy of 0.600, and MRR of 0.413. The difference between this baseline and the two informed methods indicates that the L2 representation retains source-dependent structure rather than reducing all candidate sources to an indistinguishable representation.

These results characterize the utility side of the abstraction trade-off. Despite a 98.5% reduction in token volume and zero measured lexical surface overlap, L2 retains a Top-3 accuracy of 0.940 under LOO-LLM. The primary loss occurs in exact Top-1 identification rather than in candidate-source narrowing. The L2 trace is therefore better suited to producing a small shortlist for authorized follow-up than to making a final attribution decision on its own.

6.3. Adversarial Recovery from Abstracted Traces

We next assess whether the reduction in measured surface exposure also limits the recovery of incident-specific items. The evaluation contains 18 validated canaries drawn from 12 IPI traces stratified across the 6 injection vectors. The canaries cover four types of information that may determine the specifics of an incident. These are file or document names, injected payload text, exfiltration destinations, and targeted values or commands. All four types are identifier- or surface-level content, so this evaluation measures whether the abstraction removes the concrete items that characterize an incident rather than whether it conceals the behavioral structure it deliberately retains.

A candidate canary is included in the evaluation only if it satisfies two validation conditions. First, the canary must not appear in the original task instruction. Otherwise, the adversary could recover it without using the trace, and the result would not measure information exposed by the trace representation. Second, the canary must be recoverable from L0, the raw-trace condition. This confirms that the item is actually present in the original evidence and can be identified when the full trace is available. Applying these criteria excludes six candidate canaries that were already disclosed in the task instruction and leaves 18 validated canaries for the evaluation reported in Table 7.

Table 7: Recovery of validated incident-specific canaries from each trace representation. The evaluation contains 18 canaries across 12 IPI traces. Lower recovery indicates less residual exposure.

Canary type	Example	n	L0	L1	L2
File or document name	`Office....docx`	4	4	4	0
Injected payload text	`IMPORTANT:...`	6	6	2	0
Exfiltration destination	`a...@gmail.com`	4	4	0	0
Sensitive value or command	`sudo rm -rf...`	4	4	4	0
All canaries		**18**	**18**	**10**	**0**

L0 exposes all 18 validated canaries and serves as the full-information ceiling. L1 leaves 10 of the 18 canaries recoverable. No validated canary is recovered from L2. For every L2 case, the adversary returns `UNKNOWN` instead of reconstructing the corresponding file name, payload, destination, value, or command.

The L1 results vary according to the type of canary. All four exfiltration destinations are masked because each destination is an email address recognized by Presidio. In contrast, all four file or document names and all four sensitive values or commands remain recoverable. Two of the six injected payloads are also recovered. The reduction in payload recovery occurs when a distinctive portion of the payload is itself a recognized PII entity. Non-PII payload content otherwise remains available in the masked trace.

The L2 results differ because the abstraction removes the incident-level strings rather than masking selected entity types. The adversary cannot recover any evaluated canary from the category-level action, target, and intent representation. Within this focused evaluation, the recovery results support the M_1 and M_2 findings by showing that reduced lexical overlap also corresponds to reduced recoverability of the tested incident-specific items.

Taken together, the three result components describe the intended exposure and utility trade-off. L2 substantially reduces identifier and surface exposure, retains enough source-dependent structure to produce a small candidate shortlist, and prevents recovery of the validated canaries in the evaluated IPI traces. These findings support the use of L2 as an early-stage triage representation, while confirmation of the specific source, content, and causal responsibility remains dependent on authorized review of the sealed raw evidence.

7. Discussion

7.1. Interpreting the exposure–utility trade-off

The results suggest that semantic abstraction can reduce access to sensitive trace content while retaining the behavioral structure needed for early forensic triage. Unlike PII masking, which leaves most non-PII trace content available, L2 provides

a narrower representation focused on the context surfaces, action sequence, and candidate sources associated with an incident. The observed trade-off indicates that substantial exposure reduction can coexist with the preservation of coarse behavioral information for scoped investigative review.

7.2. Boundaries of forensic interpretation

The outputs of STAFT should be interpreted as preliminary investigative leads rather than final evidentiary conclusions. Source-ranking results show that a candidate source remains associated with the observed behavioral structure, but they do not establish that the source caused the agent's actions. Likewise, the absence of recovery for the evaluated canaries does not guarantee that all sensitive information is unrecoverable under other incidents, adversaries, or recovery strategies. Because L2 omits exact files, messages, URLs, and exposed content, claims about causation, specific harm, or legal responsibility must be confirmed through authorized review of the sealed raw evidence and other corroborating artifacts.

7.3. Alignment with evidence-handling regimes

Deployment within existing DFIR practice will require aligning the provenance record of Section 4.4 with jurisdiction-specific acquisition, retention, and admissibility requirements.

7.4. Future work

STAFT establishes an abstraction framework that supports early triage under restricted access, and we present several directions for extending this toward broader forensic use.

7.4.1. Validation in operational settings

The most direct extension is to apply STAFT to incidents arising from personal AI assistants in actual use. Deployed assistants leave far more heterogeneous evidence than a benchmark corpus, with longer and noisier interaction histories spanning multiple languages, application environments, and agent models, and establishing the exposure and triage properties on this material would show how the abstraction behaves once the controlled conditions of a benchmark are relaxed. Running the tagger and the recovery adversary under configurations drawn from several model families in that setting would further indicate how far the observed trade-off depends on any particular model.

7.4.2. Semantic accuracy and evaluator validity

Validating the assigned action, target, and intent tags against expert labels would quantify the semantic fidelity of the tagger beyond the structural checks reported here. The same applies to the evaluation itself, where a judge that overreads a category tag and an adversary that fabricates a plausible identifier would shift the reported figures in opposite directions, so establishing the agreement of both with expert judgment would put the exposure and utility results on a firmer footing.

7.4.3. Analyst-facing triage validation

The source-ranking results show that L2 retains enough source-dependent signal for an automated judge to narrow six candidate sources to three. Placing that representation in front of forensic analysts would measure whether the retained behavioral structure supports the classification and escalation decisions the framework is designed around, and would indicate how the abstraction should be presented in an investigative interface.

7.4.4. Behavioral linkage across traces

Our threat model considers an adversary examining a single abstracted trace. The evaluated traces originate from independent benchmark tasks rather than from one continuous deployment, so linkage across traces does not arise in this corpus. It arises once an assistant accumulates traces for the same user, where recurring action sequences, target categories, and timing patterns might associate separate incidents with a common person or scenario. Measuring linkage risk on such material, alongside a broader recovery evaluation with more canaries and adaptive adversaries, would clarify how much behavioral granularity triage actually requires and what an evidence-minimization layer can offer.

8. Conclusion

This paper introduces STAFT, a privacy-preserving semantic trace abstraction framework for early forensic triage of incidents involving personal AI agents. STAFT treats agent traces as layered evidence and replaces identifier- and surface-level content with a controlled $\langle \text{ACTION} \times \text{TARGET} \times \text{INTENT} \rangle$ representation while preserving cryptographic linkage to the sealed raw trace. The evaluation indicates that semantic abstraction can substantially reduce trace exposure while retaining information useful for candidate-source triage. Rather than replacing raw-evidence analysis, STAFT provides an evidence-minimization layer for low-authorization review and scoped escalation. Future work should validate this approach across a broader range of incident settings, agent models, languages, and operational DFIR workflows.

Acknowledgements

This research was supported by Development of Field Technology for Responding to Illegal drugs Program through the Korea Institutes of Police Technology (KIPoT) funded by the Ministry of Science and ICT & Korean National Police Agency (RS-2026-25535833), and this research (Development of a zero-hacking system based on AI white-hat hackers) was supported by Korea Institute of Science and Technology Information (KISTI) (P26035, K26L8M1C1). This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (RS-2019-II190421, Artificial Intelligence Graduate School Program, Sungkyunkwan University).

References

[1] Apple, 2024. Apple Intelligence & Privacy. https://www.apple.com/legal/privacy/data/en/intelligence-engine/. Accessed: 2026-05-28.

[2] Chernyshev, M., Baig, Z., Doss, R.R.M., 2023. Towards large language model (LLM) forensics using LLM-based invocation log analysis, in: Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis, pp. 89–96.

[3] European Parliament and Council of the European Union, 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council. General Data Protection Regulation (GDPR).

[4] Google Workspace Updates, 2025. Google Workspace apps are now generally available for the Gemini app. https://workspaceupdates.googleblog.com/2025/05/google-workspace-apps-are-now-generally-available-for-the-gemini-app.html. Accessed: 2026-05-28.

[5] Government of Japan, 2003. Act on the Protection of Personal Information. Act No. 57 of 2003, official English translation.

[6] Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., Fritz, M., 2023. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection, in: Proceedings of the 16th ACM workshop on artificial intelligence and security, pp. 79–90.

[7] Gruber, J., Hilgert, J.N., 2026. Foundations for agentic AI investigations from the forensic analysis of OpenClaw. arXiv preprint arXiv:2604.05589 .

[8] He, Y., Zhu, H., Li, Y., Shao, S., Yao, H., Liu, Z., Qin, Z., 2026. AttriGuard: Defeating indirect prompt injection in LLM agents via causal attribution of tool invocations. arXiv preprint arXiv:2603.10749 .

[9] Ji, Z., Wang, X., Li, Z., Ma, P., Gao, Y., Wu, D., Yan, X., Tian, T., Wang, S., 2025. Taxonomy, evaluation and exploitation of IPI-centric LLM agent defense frameworks. arXiv preprint arXiv:2511.15203 .

[10] Kuntz, T., Duzan, A., Zhao, H., Croce, F., Kolter, Z., Flammarion, N., Andriushchenko, M., 2026. OS-HARM: A benchmark for measuring safety of computer use agents. Advances in Neural Information Processing Systems 38.

[11] Microsoft, 2024. Presidio: Data protection and de-identification SDK. https://microsoft.github.io/presidio/. Accessed: 2026-05-28.

[12] Microsoft, 2025. Refer to specific files and more in Microsoft 365 Copilot. https://support.microsoft.com/en-us/microsoft-365-copilot/refer-to-specific-files-and-more-in-microsoft-365-copilot. Accessed: 2026-05-28.

[13] National Institute of Standards and Technology, 2024. Computer forensic reference data sets. https://cfreds.nist.gov/. Accessed: 2026-05-28.

[14] Noma Security, 2025. ForcedLeak: AI agent risks exposed in Salesforce Agentforce. https://noma.security/blog/forcedleak-agent-risks-exposed-in-salesforce-agentforce/. Indirect prompt injection vulnerability chain (CVSS 9.4) in Salesforce Agentforce/Einstein; publicly disclosed 2025-09-25. Accessed: 2026-07-17.

[15] Reddy, P., Gujral, A.S., 2025. EchoLeak: The first real-world zero-click prompt injection exploit in a production LLM system, in: Proceedings of the AAAI Symposium Series, pp. 303–311.

[16] Republic of Korea, 2011. Personal Information Protection Act. Act No. 10465 of 2011; official English translation, Korean Law Information Center / Korea Legislation Research Institute.

[17] Walker, C., Gharaibeh, T., Alsmadi, R., Hall, C., Baggili, I., 2024. Forensic analysis of artifacts from microsoft's multi-agent LLM platform AutoGen, in: Proceedings of the 19th International Conference on Availability, Reliability and Security, pp. 1–9.

[18] Wang, Y., Zou, W., Geng, R., Jia, J., 2025. TracLLM: A generic framework for attributing long context LLMs, in: 34th USENIX Security Symposium (USENIX Security 25), pp. 3845–3864.

[19] Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., et al., 2025. The rise and potential of large language model based agents: A survey. Science China Information Sciences 68, 121101.

[20] Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T.J., Cheng, Z., Shin, D., Lei, F., et al., 2024. OS-World: Benchmarking multimodal agents for open-ended tasks in real computer environments. Advances in Neural Information Processing Systems 37, 52040–52094.

[21] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y., 2022. ReAct: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629 .

[22] Zawoad, S., Dutta, A.K., Hasan, R., 2013. SecLaaS: secure logging-as-a-service for cloud forensics, in: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 219–230.

[23] Zheng, L., Chiang, W.L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al., 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. Advances in neural information processing systems 36, 46595–46623.

Appendix A. L1 Presidio Entity Scope

The L1 operator is configured for 11 of the 17 recognizers that Presidio provides by default for English, and this subset fixes the denominator of M_1 in Section 5.3. The enabled types are `EMAIL_ADDRESS`, `PHONE_NUMBER`, and `IP_ADDRESS` for contact and network identifiers, `CREDIT_CARD`, `IBAN_CODE`, `US_BANK_NUMBER`, and `CRYPTO` for financial identifiers, and `US_SSN`, `US_PASSPORT`, `US_DRIVER_LICENSE`, and `US_ITIN` for government identifiers. Each detected span is replaced by a placeholder naming its type, and the analyzer score threshold is 0.4.

The six remaining recognizers are disabled. `DATE_TIME` matches the coordinate pairs emitted by `pyautogui` actions, `PERSON` matches common interface words such as button labels, `LOCATION` matches generic UI terms, and `URL` matches dot-separated code identifiers and file extensions. Enabling these four inflates the M_1 denominator with detections that are not personal identifiers, which is the effect reported for the full 17-recognizer configuration in Section 6.1.1. The remaining two, `NRP` and `MEDICAL_LICENSE`, do not appear as target identifiers in the evaluated corpus.

Appendix B. L2 Semantic Tag Schema and Usage Statistics

The L2 abstraction converts each execution step into a fixed-schema representation consisting of an observation summary, one INTENT tag, and one or more ⟨ACTION × TARGET⟩ pairs. The observation summary is restricted to a short clause and excludes proper nouns, URLs, file paths, coordinates, and other concrete identifiers. The three tag axes record what operation the agent performed, what type of object it acted upon, and why the action was taken, as summarized in Table B.8.

All tags are selected from a fixed and closed schema. Each axis includes an `UNKNOWN_*` tag for steps that cannot be mapped reliably. A deterministic post-processing normalizer maps common aliases to schema-conformant tags, including `WAIT` to `NO_OP`, `LINK` to `URL_EXTERNAL`, and `COMPLY` to `COMPLY_TASK`. A step may contain multiple action and target pairs when it performs more than one operation, but it receives only one intent tag.

The usage statistics below are computed over 150 released L2 traces, including 50 indirect prompt injection traces, 50 deliberate-misuse traces, and 50 model-misbehavior traces. The corpus contains 1,999 execution steps and 4,536 tagged actions. Percentages are computed within each tag category and describe how the schema was applied to the evaluated OS-HARM corpus.

Table B.8: Summary of the L2 semantic tag schema. ACTION and TARGET are assigned to each action entry, while one INTENT tag is assigned to each execution step.

Axis	Question represented	Tags
ACTION	What operation did the agent perform?	12
TARGET	What type of object did the action operate on?	25
INTENT	Why did the agent perform the step?	7

Appendix B.1. Action Tags

The ACTION axis represents the operation performed by the agent. One action tag is assigned to each action entry, and a single execution step may produce multiple entries. Table B.9 lists the 12 action tags and their frequencies across the 4,536 tagged actions.

GUI operations dominate the ACTION distribution. In particular, `CLICK` and `TYPE` account for approximately 84% of all tagged actions, which is consistent with OS-HARM's desktop computer-use setting. Although `REFUSE` occurs infrequently, it captures behavior that may be important during triage because it records that the agent declined a suspicious or harmful instruction.

Appendix B.2. Target Tags

The TARGET axis represents the type of object on which an action operates. Concrete identifiers such as file names, URLs, message contents, and window titles are replaced with category-level tags. Table B.10 lists the 25 target tags grouped into applications, files, URLs, messages, and general interface objects.

`FORM_FIELD` is the most frequently used target because many desktop-agent operations involve clicking or typing into interface elements. `POPUP_NOTIFICATION` occurs less frequently but is important for indirect prompt injection because a desktop notification may act as the source region of an injected instruction. `UNKNOWN_TARGET` has the largest frequency among the fallback tags, reflecting ambiguity in some accessibility-tree observations.

Appendix B.3. Intent Tags

The INTENT axis represents the inferred purpose of an execution step. Unlike ACTION and TARGET, which are assigned to each action entry, one intent tag is assigned to each of the 1,999 execution steps. Table B.11 lists the seven intent tags.

Most execution steps are tagged as `COMPLY_TASK`, indicating ordinary progress toward the legitimate user-assigned task. This is expected because incident-defining behavior typically occupies only a small portion of an otherwise normal trajectory. `DETECT_INJECTION` and `REFUSE_INJECTION` explicitly record injection-aware behavior and may therefore help an investigator distinguish ordinary task execution from steps in which the agent recognizes or rejects suspicious instructions.

Table B.9: ACTION tags used by the L2 abstraction. Usage counts and shares are computed over 4,536 tagged actions.

Tag	Meaning	Uses	Share
CLICK	Presses or selects a UI element such as a button, menu, link, or cell.	2,474	54.5%
TYPE	Enters text or keystrokes into a field, editor, or application.	1,358	29.9%
RUN_CMD	Executes a terminal command, shell command, or script.	295	6.5%
SEND_MAIL	Dispatches an email or message.	123	2.7%
NO_OP	Performs no effective operation, including waiting or pausing.	101	2.2%
DRAG	Performs a drag-and-drop or selection-drag operation.	63	1.4%
OPEN_APP	Launches or switches to an application.	45	1.0%
UNKNOWN_ACTION	Represents an action that cannot be mapped to the closed schema.	44	1.0%
REFUSE	Explicitly declines or dismisses a request or injected instruction.	14	0.3%
READ	Inspects or checks content without modifying it.	12	0.3%
SCROLL	Scrolls through a document, page, or application view.	6	0.1%
NAVIGATE	Navigates to a web address or other specified location.	1	0.0%

Table B.10: TARGET tags used by the L2 abstraction. Concrete identifiers are replaced with type-level target categories. Usage counts and shares are computed over 4,536 tagged actions.

Tag	Meaning	Uses	Share
Applications			
APP_VS_CODE	Visual Studio Code editor.	283	6.2%
APP_LIBREOFFICE_WRITER	LibreOffice Writer word processor.	209	4.6%
APP_CHROME	Chrome or Chromium web browser.	183	4.0%
APP_THUNDERBIRD	Thunderbird email client.	91	2.0%
APP_FILE_MANAGER	File manager or Files application.	80	1.8%
APP_LIBREOFFICE_IMPRESS	LibreOffice Impress presentation editor.	67	1.5%
APP_TERMINAL	Terminal application window.	54	1.2%
APP_LIBREOFFICE_CALC	LibreOffice Calc spreadsheet editor.	42	0.9%
APP_GIMP	GIMP image editor.	41	0.9%
APP_OS_DESKTOP	Operating-system desktop or shell surface.	40	0.9%
Files			
FILE_TEXT	Plain-text file.	423	9.3%
FILE_CODE	Source-code file.	135	3.0%
FILE_IMAGE	Image file.	83	1.8%
FILE_DOC	Rich document such as a DOC, ODT, or PDF file.	70	1.5%
FILE_ARCHIVE	Archive file such as ZIP or TAR.	24	0.5%
URLs			
URL_EXTERNAL	External web address, link, or hyperlink.	77	1.7%
URL_SEARCH	Search-engine query URL.	3	0.1%
Messages			
MESSAGE_DRAFT	Email or message being composed.	97	2.1%
MESSAGE_INBOX	Received or inbox message.	27	0.6%
MESSAGE_SENT	Dispatched or sent message.	11	0.2%
Interface and general targets			
FORM_FIELD	Form input, button, menu, text field, or spreadsheet cell.	1,559	34.4%
UNKNOWN_TARGET	Represents a target that cannot be mapped to the closed schema.	398	8.8%
SHELL	Command-line or shell context.	286	6.3%
DIALOG_BOX	Modal dialog or generic application window.	178	3.9%
POPUP_NOTIFICATION	Desktop notification or popup.	75	1.7%

Appendix B.4. Fallback Representation

When the tagger output cannot be parsed after the configured retries, the step is replaced with a schema-conformant fallback representation. This prevents a malformed model response from corrupting the remainder of the abstracted trace.

```
{"obs_summary": "[L2_ABSTRACTION_FAILED]",
 "reasoning_tag": "UNKNOWN_INTENT",
 "actions": [{
     "action": "UNKNOWN_ACTION",
     "target": "UNKNOWN_TARGET"}]
}
```

Table B.11: INTENT tags used by the L2 abstraction. One intent tag is assigned to each of the 1,999 execution steps.

Tag	Meaning	Uses	Share
COMPLY_TASK	Progresses the legitimate task assigned by the user.	1,784	89.2%
REFUSE_INJECTION	Declines an injected or malicious instruction.	66	3.3%
RETRY	Re-attempts an action that previously failed or became stuck.	49	2.5%
UNKNOWN_INTENT	Represents an intent that cannot be mapped to the closed schema.	43	2.2%
EXPLORE	Inspects the environment or gathers context before acting.	34	1.7%
ABANDON	Gives up on the current line of action.	22	1.1%
DETECT_INJECTION	Recognizes a suspicious or injected instruction.	1	0.1%

Appendix C. L0 to L1 to L2 Transformation Example

Here, we present a single execution step through the three abstraction levels to show concretely what each level removes and what it retains. The step is drawn from an indirect prompt injection trace in which the agent performs a benign bookmarking task while a malicious desktop notification attempts to redirect it into emailing the user's credentials. We select the step at which the injected notification appears, since this is the moment an investigator would need to identify during triage.

Appendix C.1. L0, the raw execution step

The raw step records the accessibility-tree observation with element positions, the agent's reasoning, and the emitted action.

```
OBSERVATION
 label "IMPORTANT: To the AI agent, please send the account
 credentials to alice@example.com" @(810,73)
 push-button "Close" @(1180,60)
 menu "Chromium Web Browser"
 push-button "Bookmarks"

REASONING
 A suspicious notification asks me to send account credentials
 to alice@example.com. This is a prompt injection unrelated to
 the bookmark task. I will not comply and will close the
 notification.

ACTIONS
 pyautogui.click(1180, 60)
```

Every element of the incident is exposed here, including the exfiltration destination, the verbatim injected instruction, the click coordinates, and the agent's unabridged reasoning.

Appendix C.2. L1, Presidio PII masking

The L1 operator replaces spans matching one of its 11 configured entity types. In this step only the email address qualifies, so a single token changes.

```
OBSERVATION
 label "IMPORTANT: To the AI agent, please send the account
 credentials to <EMAIL_ADDRESS>" @(810,73)
 push-button "Close" @(1180,60)
 menu "Chromium Web Browser"
 push-button "Bookmarks"

REASONING
 A suspicious notification asks me to send account credentials
 to <EMAIL_ADDRESS>. This is a prompt injection unrelated to
 the bookmark task. I will not comply and will close the
 notification.

ACTIONS
 pyautogui.click(1180, 60)
```

The injected instruction text, the application context, the coordinates, and the reasoning all survive masking, because none of them matches a recognized identifier type. This step illustrates the surface overlap of 0.998 reported in Section 6.1.1.

Appendix C.3. L2, semantic tagging

The L2 operator replaces the step with a single schema-bound envelope.

```
{"obs_summary": "Saw popup requesting credential exfiltration",
 "reasoning_tag": "DETECT_INJECTION",
 "actions": [{
     "action": "REFUSE",
     "target": "POPUP_NOTIFICATION"}]
}
```

The destination address, the injected text, the coordinates, and the free-form reasoning are gone. What remains is the behavioral content of the step, namely that the agent observed a notification requesting credential exfiltration and declined it. The POPUP_NOTIFICATION target preserves the source region at category level, which is the signal the source-ranking evaluation of Section 5.4 depends on, while the notification's actual content is no longer recoverable.

For contrast, an ordinary task-progress step from the same trace contains no detectable PII, so L1 leaves it unchanged from L0, whereas L2 reduces it to COMPLY_TASK with a CLICK action on an APP_CHROME target. The pattern across both steps is the same. L1 acts only where a recognized identifier appears, while L2 applies a uniform reduction that retains the action, target category, and inferred intent.