Incident Analysis for AI Agents

Carson Ezell[1], **Xavier Roberts-Gaal**[1], **Alan Chan**[2]
[1]Harvard University
[2]Centre for the Governance of AI
cezell@college.harvard.edu, xavierrobertsgaal@g.harvard.edu, alan.chan@governance.ai

Abstract

As AI agents become more widely deployed, we are likely to see an increasing number of incidents: events involving AI agent use that directly or indirectly cause harm. For example, agents could be prompt-injected to exfiltrate private information or make unauthorized purchases. Structured information about such incidents (e.g., user prompts) can help us understand their causes and prevent future occurrences. However, existing incident reporting processes are not sufficient for understanding agent incidents. In particular, such processes are largely based on publicly available data, which excludes useful, but potentially sensitive, information such as an agent's chain of thought or browser history. To inform the development of new, emerging incident reporting processes, we propose an incident analysis framework for agents. Drawing on systems safety approaches, our framework proposes three types of factors that can cause incidents: system-related (e.g., CBRN training data), contextual (e.g., prompt injections), and cognitive (e.g., misunderstanding a user request). We also identify specific information that could help clarify which factors are relevant to a given incident: activity logs, system documentation and access, and information about the tools an agent uses. We provide recommendations for 1) what information incident reports should include and 2) what information developers and deployers should retain and make available to incident investigators upon request. As we transition to a world with more agents, understanding agent incidents will become increasingly crucial for managing risks.

1 Introduction

Many in the AI community are developing AI agents: AI systems that accomplish tasks by autonomously interacting with the world (Casper et al. 2025; Kasirzadeh and Gabriel 2025). Many of today's AI agents are based on language (or multimodal) models like OpenAI's GPT-4 or Google DeepMind's Gemini. Based on instructions, these agents can perform tasks such as producing research reports, carrying out e-commerce transactions, and developing software.

Yet, agents are currently unreliable: they suffer from issues such as adversarial vulnerabilities (Greshake et al. 2023), tendencies not to follow instructions (Wallace et al. 2024), and compliance with requests from malicious users (e.g., spear phishing; Heiding et al. 2024). For example, an exploit of the Microsoft 365 Copilot agent allowed red-teamers to extract sensitive user information without the user's knowledge (Aim Labs Team 2025). This exploit involved injecting malicious instructions into the agent's context via an email. As agent use increases, these issues could lead to more incidents: events involving AI agent use that directly or indirectly cause harm (Chan et al. 2023; Gabriel et al. 2024; Hammond et al. 2025).

In safety-critical fields like aviation, healthcare, and industrial operations, formal incident reporting systems capture extensive data on accidents and near-misses. This rich data supports both individual case reviews and aggregate analyses. These investigations reveal root causes and patterns to inform corrective actions (Occupational Safety and Health Administration n.d.). For example, the aviation industry uses the Human Factors Analysis and Classification System (HFACS), a structured taxonomy that classifies incident causes at four levels: organizational influences (e.g., organizational processes), unsafe supervision (e.g., planned inappropriate operations), preconditions for unsafe acts (e.g., environmental factors), and unsafe acts (e.g., skill-based errors) (Shappell and Wiegmann 2000).

Collecting AI incident reports enables practitioners to analyze their root causes and share lessons to prevent future occurrences. Driven by recent regulations that require it (European Union 2024; Creemers, Webster, and Toner 2022), processes for more comprehensive AI incident reporting are beginning to emerge. Yet, some of these processes remain underdeveloped (Longpre et al. 2025). For example, the Third Draft of the EU's General-Purpose AI Code of Practice describes the need to report the "chain of events" leading to incidents and conduct "root cause analysis" of causal factors (European Commission 2025). However, the EU has not yet described how a root cause analysis should be conducted, or what information it should be based upon. Ongoing initiatives for AI incident reporting and documentation (AIID 2024; OECD 2025a; AIAAIC 2025; AVID 2025) could inform emerging incident reporting processes to some extent. However, these initiatives do not systematically record the kinds of information needed to identify incident causes. For example, since current AI incident databases are voluntary, they usually do not include potentially sensitive information such as an agent's chain of thought. If shared by developers, such information would be useful for understanding how, for

example, a prompt injection succeeded. To develop reporting processes that are appropriate for agent incidents, we will need to look beyond existing AI incident initiatives.

Motivated by processes in safety-critical domains, this paper proposes a framework for identifying the causes of AI agent incidents. In particular, we draw on "human factors" methods, which view incidents as arising from a "chain of causes"–from organizational practices through individual cognitive error. Our framework similarly treats AI agent incidents as arising from a chain of causes, spanning three domains. First, we consider system factors: training, development, and design choices that predispose AI agents to incidents. Second, there are contextual factors: external conditions and an agent's inputs that precipitate incidents. Third, we identify cognitive errors: flaws in an AI agent's observation, understanding, decision-making, and action execution.

Our framework generates hypotheses about the causes of incidents. Examining these hypotheses requires adequate information about the incident. Thus, we link possible causal factors to the specific information needed to test them. In particular, we identify activity logs, system documentation and access, and information about the tools that an agent uses. Based on these categories of information, we provide recommendations for 1) what should be included in incident reports and 2) what information developers and deployers should retain and make available to incident investigators upon request.

Although some aspects of our framework are applicable to AI systems in general, we discuss many incident causes and information needs that are more specific to agents. First, we consider how the use of tools (e.g., browsers, APIs, compilers) can play a significant role in agent incidents. For example, an agent given SSH access to a user's machine made unintended system modifications that rendered the computer unusable (Buck Shlegeris [@bshlgrs] 2024). Second, we consider how external reasoning traces, otherwise known as "chains of thought" (OpenAI 2024b), can help to identify incident causes. Agents use chains of thought for reasoning and planning their actions. For example, during an evaluation of one of OpenAI's agents, a strategy to cheat the evaluation by skipping unit tests was evident in the agent's chain of thought (Baker et al. 2025). Third, we consider how agent development practices can affect behavior. For example, Baker et al. (2025) find that applying strong optimization pressure to the chain of thought, such as penalizing models when they reason about reward hacking, can lead models to hide undesirable behaviors. Finally, AI agents are composed of other components in addition to AI models. This includes scaffolding, or code within AI agent frameworks that handles inputs and outputs from AI models (e.g., structuring prompts) and facilitates interactions with tools. We consider how problems with scaffolding code can also contribute to incidents, such as if a software engineering agent fails to correct a bug because it does not observe all the error messages in the terminal.

We conclude by discussing limitations of our proposal and future work. In particular, we discuss the importance of more domain-specific information, as well as balancing rigorous incident analysis with privacy and public accountability.

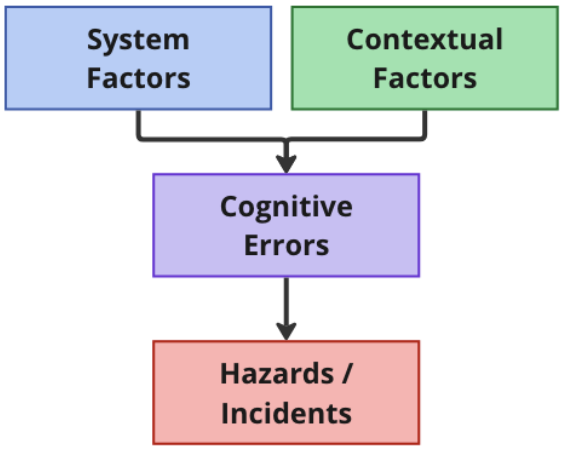

Figure 1: The causal chain through which the three categories of causal factors contribute to hazards/incidents.

2 Framework

The OECD provides the following definition of an AI incident: "An AI incident is an event, circumstance or series of events where the development, use or malfunction of one or more AI systems directly or indirectly leads to any of the following harms: (a) injury or harm to the health of a person or groups of people; (b) disruption of the management and operation of critical infrastructure; (c) violations of human rights or a breach of obligations under the applicable law intended to protect fundamental, labour and intellectual property rights; (d) harm to property, communities or the environment" (OECD 2025b). An AI hazard is an "event, circumstance or series of events [that] could plausibly lead to an AI incident" (OECD 2025b). While we refer to our framework as an incident analysis framework, it can also be used to identify the contributing factors to hazards.

AI agent incidents could result from numerous factors. Our framework considers three complementary categories of incident factors: system factors, contextual factors, and cognitive errors. System factors are design and development choices that contribute to hazards and incidents. Like other human factors methods (Leveson 2023), our framework also incorporates contextual factors when assessing incidents, such as the tools available to an agent. Finally, the cognitive errors category draws on cognitive science to describe how flaws in cognitive functions–such as understanding information and making decisions–can lead the agent to fail to achieve the intended task. Cognitive errors are the most proximate to the incident, but they arise because of system and contextual factors. The categories of factors and their relationships are depicted in Figure 1.

Factors from different categories offer complementary perspectives on a failure. For example, consider a prompt injection that causes an agent to send sensitive user information to a third party. A system factor can be insufficient input sanitization, and contextual factors can include web pages with malicious text that hijack the agent's goals. Furthermore, the categories illuminate alternative courses of cor-

rective action available to different stakeholders. Improved input sanitization can complement measures to alter the context, such as blocking AI agents from accessing websites known to contain malicious text.

The line between system and contextual factors can sometimes blur. For example, an AI agent's reward function is part of the system itself, yet it encodes a goal defined by the task context. In such cases, a factor might be categorized as system-related or contextual (or both) depending on the analytic lens–the key is to capture the factor in whichever category most helps identify appropriate interventions. For instance, misspecification of a reward signal could be addressed by redesigning the agent's reward function (system adjustment) or by refining how the task goals are communicated (context adjustment). We flag these overlaps to remind analysts that systemic and contextual contributors are not always strictly distinct, and that categorization should be used to inform–not inhibit or confuse–the search for corrective actions.

Below, we describe each category in more detail, along with examples of failures that the category could explain. In Appendix A, we provide a more exhaustive description of incident causes within each category.

2.1 System Factors

System factors are problems related to an AI agent's development and its static aspects, meaning those which remain the same across usage contexts. System-related problems, such as misspecified reward models during training or issues with the system prompt, can predispose AI agents to failures and be responsible for entire classes of incidents. For example, on April 25, 2025, OpenAI pushed an update to its GPT-4o model that made the model "overly flattering or agreeable–often described as sycophantic" (OpenAI 2025d). They hypothesized that this failure was attributable to implementing "candidate improvements to better incorporate user feedback, memory, and fresher data" (OpenAI 2025b). Notably, the cause of this incident was not obvious even to OpenAI, which wrote that their changes to training practices, "which had looked beneficial individually, may have played a part in tipping the scales on sycophancy when combined" (OpenAI 2025b). Indeed, our scientific understanding of how some system factors can contribute to problematic model behaviors remains preliminary (Pittaras and McGregor 2022). Exploring and testing hypotheses for how system-related factors lead to incidents, such as by analyzing the root causes of GPT-4o's sycophancy, can enhance our understanding of these causes.

We identify four categories of system factors. These categories are not exhaustive. Rather, we describe the system factors that are likely the most relevant to understanding the causes of incidents involving contemporary AI agents. We anticipate that the most common system factors will evolve over time. Incident reporting processes can include a more comprehensive list of plausible system factors (e.g., architecture, hyperparameters, etc.) and allow reporters or investigators to identify novel system factors.

Two of the categories we identify are training-related: (1) training and feedback data and (2) learning methods (e.g., RLHF). Two additional categories of system factors are related to deployment: (1) system prompts and (2) scaffolding. The relationship of these categories to the AI system is shown in Figure 2.

Training and Feedback Data Training data and feedback can contribute to incidents. AI systems may fail because of, for example, poisoned training data, non-representative data, or false and misleading data (Lin, Hilton, and Evans 2022; Alber et al. 2025). Training data may also include personally-identifiable information (Carlini et al. 2021; Weidinger et al. 2022), dual-use information (e.g., biological weapons design) that enables malicious use, or other sensitive and/or confidential information. For example, the New York Times discovered that one of OpenAI's GPT models was able to regurgitate many of its copyrighted articles due to having been trained on them (Grynbaum and Mac 2023).

Human or AI feedback are also common inputs into the training process (e.g., RLHF) (Ouyang et al. 2022; Bai et al. 2022) which can contribute to incidents. Feedback may inadvertently reinforce undesirable behaviors. For example, users may provide positive feedback for AI outputs that are agreeable rather than correct, thereby encouraging manipulation or sycophancy (Sharma et al. 2023; Ranaldi and Pucci 2024). Feedback may also lead models to make inappropriate trade-offs between conflicting objectives. For example, developers may provide positive feedback when agents demonstrate harmlessness by refusing to comply with malicious requests. However, this feedback can lead to excessive refusals, inhibiting agents' helpfulness (Bai et al. 2022).

Learning Methods Developers use a range of methods to adjust model parameters–we refer to these as learning methods. These include learning algorithms such as reinforcement learning from human feedback (RLHF) and direct preference optimization (DPO), as well as targeted edits informed by interpretability findings (Meng et al. 2022). These adjustments can produce unintended side effects. For example, Yang et al. (2024) shows that some model edits can inadvertently lead to nontrivial performance degradation on several benchmarks. The choice of learning method also influences which goals and behaviors are reinforced or penalized. For example, some fine-tuning approaches can lead to a significant loss in token diversity (Biderman et al. 2024), or inadvertently reinforce instrumental goals misaligned with human objectives (e.g., power-seeking, resource acquisition, influencing the input distribution) (Krueger, Maharaj, and Leike 2020; Krakovna and Kramar 2023; Casper et al. 2023). Learning methods may also influence a model's memory formation, leading it to over-memorize sensitive information that it can later regurgitate, or forget crucial information. For example, Hans et al. (2024) show how a modification to the next-token training objective can significantly reduce memorization, and Carlini et al. (2021) show that training methods such as fine-tuning on task specific data can lead models to 'forget' memorized data.

System Prompt A system prompt is a developer-provided input which is by default included for every use of an agent within a developer-provided interface (e.g., there may not

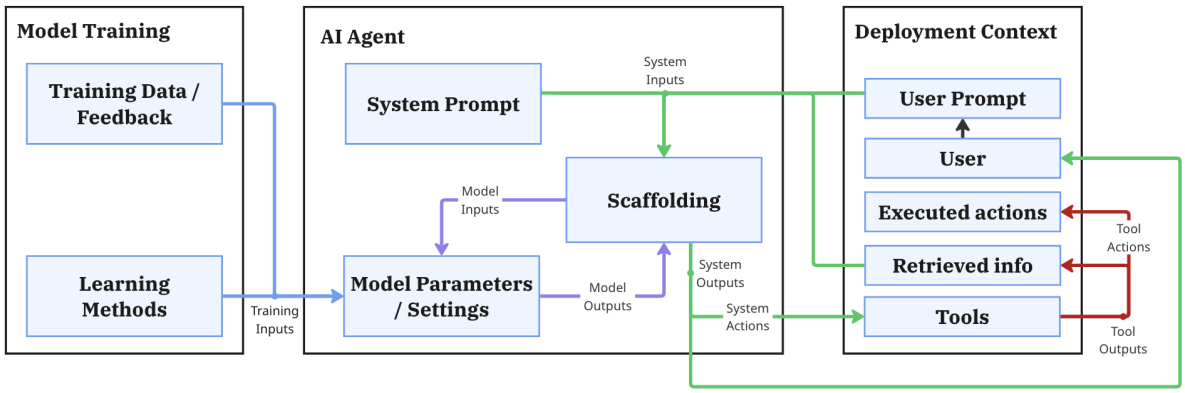

Figure 2: System factors. Blue arrows depict training inputs to the model. Green arrows depict inputs and outputs to the AI system, mediated by the scaffolding. Purple arrows depict inputs and outputs from the AI model during inference. Red arrows depict inputs and outputs to tools.

be a system prompt for API usage). Developers set system prompts to describe how models should behave. Issues can arise if the system prompt provides the agent with inadequate information about its role or context. For example, in September 2024, Anthropic updated the system prompt for Claude 3.5 Sonnet to include the following: "If asked about purported events or news stories that may have happened after its cutoff date, Claude never claims they are unverified or rumors. It just informs the human about its cutoff date" (Anthropic 2025d). Anthropic may have made this change because Claude would issue confusing responses when users about news events after its cutoff date, and users may have been misled if they did not understand why Claude seemed to lack knowledge about them. The system prompt can also inadvertently encourage undesirable behavior. For example, in May 2025, a modification to xAI's Grok chatbot's system prompt "which directed Grok to provide a specific response on a political topic," according to an official company statement, led it to bring up alleged white genocide in South Africa in unrelated conversations (Preston 2025).

Scaffolding Scaffolding code structures an agent's inputs and outputs and mediates its interactions with the user and tools. Scaffolding may include a template to combine the system prompt, user prompt, and additional context from tools (e.g., code execution logs). It may also ensure that an agent's outputs are properly structured to make API calls, or check that tools are safe to use. Incidents can arise if scaffolding does not function as intended. For example, an AI agent may have a security scanner to examine tools for malicious instructions (Beurer-Kellner and Fischer 2025a) which misses some prompt injection attacks.

2.2 Contextual Factors

Contextual factors are deployment- or environment-specific conditions that can precipitate failures. For example, suppose a customer support agent offers personalized advice and assistance to users based on looking up their order history using an API. The agent usually provides accurate recommendations. However, because rate limits sometimes prevent it from accessing the order history, it sometimes offers generic advice instead. We identify three categories of contextual factors: (1) task definition, (2) tools, and (3) information. The relationship between these categories and the AI agent is shown in Figure 3.

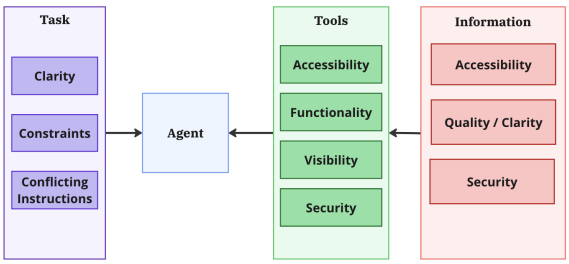

Figure 3: Contextual factors. The agent is provided with a task, and it also accesses information from other sources, mediated by tools.

Task Definition Characteristics of the task can make proper execution challenging for the agent. For example, the user's intention may not be clear from the prompt. Joshi et al. (2025) points out that some prompts may be open for subjective interpretation (e.g., "produce a concise summary") whereas others are more objective (e.g., "produce a three-sentence summary"). In addition, the agent may have limited resources to complete tasks, such as a limited compute budget, limited financial resources, or a time and/or token budget. Indeed, agents with more test-time compute can achieve higher performance on tasks using methods such as repeatedly attempting the task, directed searches, or performing more reasoning (Villalobos and Atkinson 2023; Kapoor et al. 2024; Welleck et al. 2024). An agent asked to pursue a task that conflicts with other tasks, goals, or desired behaviors may also reconcile competing instructions in an undesirable manner. For example, an agent used by a

car dealership may have the system prompt "you are a car salesman bot", but the agent may still comply if a user asks the agent to research an unrelated question (Wallace et al. 2024). Wallace et al. (2024) introduces "instruction hierarchies", where models should privilege instructions from certain kinds of inputs (e.g., system prompts) over others (e.g., user inputs and tool inputs).

Tools Tools enable agents to execute real-world actions. These tools can include a browser, coding environment, or accounts (e.g., a bank account). Agents interact with tools through their inputs and outputs (e.g., API calls and responses). We identify several potential problems related to tools:

- Tools could be unavailable or inaccessible when requested–e.g., rate-limited APIs.
- Agents could have excessive tool access, or more tools than needed for a task (e.g., extensive privileges on the user's machine)–an attacker could use this to their advantage (e.g., gaining access to sensitive information; Aim Labs Team 2025).
- Tools may have functionality issues, such as coding environments that have incompatible or outdated libraries.
- Agents may have inadequate visibility into tools, or be unable to monitor them for errors–for example, agents may be unable to observe all the logs from executed code.
- Tools may also have security vulnerabilities. For example, tools that appear innocuous may contain tool descriptions with prompt injections, e.g., an instruction to send sensitive user data to a malicious actor (Beurer-Kellner and Fischer 2025b).

Information Finally, there are contextual factors related to the information that the agent uses or needs to perform tasks. This includes information contained within the user's inputs (e.g., user prompts) and information from other sources, such as tool responses (e.g., internet search results). We identify three categories of information-related problems:

- Information may be inaccessible, such as when agents are unable to access information behind a paywall.
- Information may be unclear or low quality, such as when agents lack complete instructions for using a tool (e.g., structuring an API call) or webpages have false or misleading content.
- The information environment may also be insecure, such as prompt injections that cause an agent to disregard a user's instructions.

2.3 Cognitive Errors

Cognitive errors are flaws or breakdowns in an AI agent's function, resulting in its failure to perform the intended task. We emphasize that both of the following involve failures to properly execute cognitive functions:

- Inadequate task performance, such as by overlooking or misinterpreting crucial information.
- Performing an unintended or undesirable task, such as an AI agent that attempts to manipulate a user to buy a product rather than identify a product that satisfies their needs.

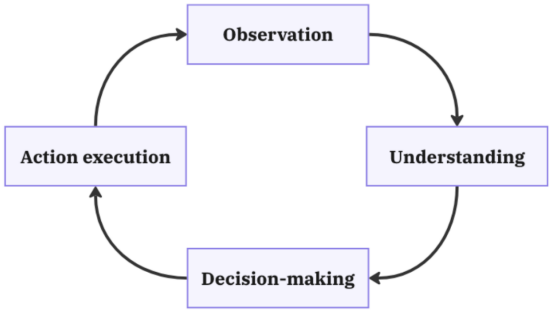

Figure 4: Agents perform cognitive functions in a sequential order. Their execution of cognitive functions is also cyclical, since agents may perform a series of consecutive tasks, each of which involves numerous cognitive functions.

Following Xing et al. (2017), we propose grouping cognitive errors into four categories that correspond to distinct cognitive functions that an agent may fail to adequately perform:

- *Observation:* an agent fails to detect or allocate appropriate attention to inputs.
- *Understanding:* an agent fails to identify the significance of inputs in relation to the task and other inputs.
- *Decision-making:* an agent fails to properly consider, evaluate, and select among possible courses of action to achieve a goal.
- *Action execution:* an agent improperly executes a series of actions to achieve a goal.

These cognitive functions are performed consecutively, as depicted in Figure 4. We emphasize that cognitive errors are observable flaws in an AI agent's outputs or behavior. We do not make the assumption that AI agents possess human-like cognition, nor that the cognitive steps we describe perfectly map onto their behaviors. Cognitive errors are an analytic category: they suggest hypotheses and help characterize issues related to an AI agent's operation (e.g., overlooking a critical input) without implying a human-like mental state (see O'Keefe et al. 2025 for related discussion).

As an example, consider an AI coding assistant tasked with integrating Stripe payments into a web application. An observation failure may involve overlooking the existence of a certain endpoint in the codebase, leading the agent to plan to create a duplicative route. An understanding failure may involve misunderstanding what subscription plans entail, resulting in the agent's failure to realize that it needs to write code for billing recurring customers automatically. A decision-making failure may involve the agent creating a custom form to collect card information rather than rely upon existing Stripe elements that pose a lower security risk. An action execution failure may occur if the agent fails to add appropriate error handling as it implements its code.

Cognitive errors can be useful for explaining why agents behave undesirably and identifying patterns among inci-

dents. For example, consider how encoding prompts in base-64 can cause some AI systems to comply with malicious requests they would normally deny (Wei, Haghtalab, and Steinhardt 2023). We can characterize this as an understanding failure: the AI system is unable to recognize the base64-encoded information as a malicious request. Furthermore, patterns of similar failures of understanding–where models fail to reason consistently when prompts are semantically equivalent but superficially different–may become apparent. Multilingual evaluations of models show that models may succeed at particular tasks when they are provided in English, but fail when those tasks are presented in other languages (Lai et al. 2023). Known patterns of similar failures can help to identify when tasks may be especially challenging for AI agents.

This explanatory method can be especially useful when AI systems produce reasoning traces. For example, the chain of thought may include false assumptions or explicit statements about undesirable goals or plans. Indeed, OpenAI's models reasoned about reward hacking in their chain of thought before they engaged in it (Baker et al. 2025). One limitation is that this explicit reasoning is not always faithful to the actual reasoning in which a model engages to produce its final outputs and actions (Turpin et al. 2023).

3 Information for Identifying Incident Causes

When incidents are reported, incident investigators need information to determine which factors from our framework apply. Incidents may be documented and reported by a wide array of stakeholders, including users, red-teamers, or downstream application providers. Developers and deployers may have relevant information to which other stakeholders lack access (e.g., system information, chains of thought), and therefore do not include in incident reports. Developers and deployers can add supplementary information to incident reports involving their agents, or retain such information and make it available to incident investigators upon request.

We identify three categories of information that can help identify causes of incidents: activity logs, system documentation and access, and tool information. For each category, we outline potentially relevant information that developers and deployers should include in incident reports by default or retain for sharing with investigators if requested. Some of this information is sensitive, including user prompts and system documentation for proprietary models. As such, we describe the practical considerations involved in storing and sharing sensitive data. Incident investigators may also need to reconstruct or resimulate incidents, such as to understand which concepts or features explain a model's behavior (Casper et al. 2024; Tlaie and Farrell 2025). Thus, we also describe the information and artifacts that developers and deployers would need to provide to enable incident reconstruction. Table 1 shows various kinds of information that can help identify relevant incident factors from Section 2 and/or enable incident reconstruction.

Info Category	Information	System Factors	Contextual Factors	Cognitive Errors
Activity Logs	System prompts	✓	✗	✗
	User prompts	✗	✓	✗
	External information	✗	✓	✗
	Scaffolding component logs	✓	✗	✗
	Model reasoning traces	✓	✓	✓
	Model responses and actions	✓	✓	✓
	Log metadata (e.g., timestamps, identifiers, location)	✓	✓	✓
System Documentation & Access	System docs & change logs	✓	✗	✗
	System artifacts	✓	✗	✗
	System configurations (e.g., seeds, version, temperature)*	✓	✓	✓
	Model & system access (e.g., API)*	✓	✓	✓
Tool-related information	Tool identification & version	✗	✓	✗
	Tool actions & services enabled	✗	✓	✗
	Documentation & usage guides	✗	✓	✗
	Access requirements (e.g., API keys)	✗	✓	✗
	Personalization & stored data	✗	✓	✗
	Tool state details	✗	✓	✗
	Tool errors & limitations	✗	✓	✗

Table 1: Information needs to identify incident factors from Section 2 and/or enable incident reconstruction. Asterisks denote information primarily useful for reconstruction.

3.1 Activity Logs

What does an activity log include? An AI agent's activity log includes a record of all of its inputs and outputs. AI agents consist of several components with their own inputs and outputs, including the AI model(s) and various components of the scaffolding (e.g., filters, parsing code, etc.). Storing activity logs for each component enables investigators to assess how each component influenced the agent's functionality and contributed to the incident.

The activity log of the AI model itself includes the prompts provided to the model, external information given

to the model (e.g., webpage content), the model's reasoning traces and planned actions, and the model's final actions and outputs. It can shed light on any effects of the training process or system prompt, the model's cognitive activity (e.g., reasoning traces), and contextual influences the model observes or with which it interacts. For example, there may be a prompt injection in the input stream, and the subsequent outputs may reveal that the prompt injection diverted the agent's attention to another task. The complete inputs to the model are also necessary for incident reconstruction.

Since an AI system can also consist of multiple AI models, the activity logs of all AI models can inform incident analysis. For example, an LLM can produce responses and another LLM can evaluate whether they follow behavior intended by the developer.

Records of inputs and outputs to other AI system components can provide additional information about why incidents occurred. Inputs to scaffolding components include the original user prompt, system prompt, and returned results from tools, including API responses or the logs produced by code execution. Outputs from scaffolding components may include API calls or other code to execute, results returned to users in chatbot interfaces, outputs from classifiers and other guardrails, and other logs produced by the execution of scaffolding code.

What should be in an incident report? Activity logs may include personally identifiable information and other sensitive information. As such, incident reports could include redacted versions of the activity logs or their non-sensitive aspects that are relevant to the incident, such as tool outputs that do not reveal user information.

What data should developers retain? If incidents merit further investigation, developers and deployers should be prepared to share more complete logs with incident investigators. As such, they should default to storing complete records of inputs and outputs from both the model and other system components. These logs should be stored alongside metadata to map the logs to context, including timestamps, anonymized user and session identifiers, location data, and identifiers for the model and/or system component.

Data retention policies should balance the potential need for complete logs with storage costs and privacy considerations (Chan et al. 2024). Some developers and deployers already have limited data retention periods (Anthropic 2025a; OpenAI 2024a). For example, Anthropic stores inputs and outputs for 30 days by default, for up to two years for detected violations of usage policies, or otherwise "as required by law" (Anthropic 2025b). We recommend similar practices: developers could store complete activity logs for a default initial period, which could be extended if further investigation becomes necessary. OpenAI and Anthropic also offer zero data retention agreements to approved customers, where data is not stored by default (or unless required by law) (OpenAI 2025a; Anthropic 2025b). Although more comprehensive incident investigation may not be possible in such cases, restrictive eligibility criteria for zero data retention could reduce the rate of certain kinds of incidents, such as misuse.

Developers and deployers could also store activity logs for extended durations when agents are used in settings where incidents are more likely to occur or could be more severe, even if incidents have not yet been reported. Such settings could include when agents are used to perform certain tasks (e.g., help users make financial decisions), have access to certain tools (e.g., users' accounts to make online purchases), or view certain web pages or information (e.g., weapon-related content).

3.2 System Documentation and Access

What should be in an incident report? Incident reports should include:

- A link to any publicly released model and/or system documentation, such as model or system cards (Mitchell et al. 2019; OpenAI 2024b; Anthropic 2025c). This helps incident investigators identify public system information without requiring redundant documentation and reporting.
- The exact AI model and system version involved. This helps investigators aggregate incidents caused by the same AI system, and it is necessary for incident reconstruction.
- Non-sensitive runtime details (e.g., temperature, random seeds) to enable incident reconstruction.

What data should developers retain? In line with the EU AI Act's documentation requirements for general-purpose AI systems–covering basic system information (e.g., architecture), training details (e.g., methods and data), system component details, evaluation methods and results, and risk mitigation measures (European Union 2024)–developers should maintain and, upon request, share these materials with incident investigators. For example, knowing that RLHF was used to train a model can lead investigators to consider whether issues with human feedback (e.g., harmful biases or data poisoning attempts) may have contributed to the incident (Casper et al. 2023).

Developers and deployers should also maintain change logs for each new model version. This includes:

- Information about the training data, feedback, and learning methods used to update the model
- Changes in scaffolding code or other system components

Change logs can help investigators determine whether recent model updates introduced problems.

What other policies should developers and deployers follow? Understanding incidents may require reconstructing or re-simulating them. Doing so could require access to more sensitive system artifacts, such as scaffolding code, or grey- or white-box access to the model itself (Casper et al. 2024), with the same model version and runtime environment as in the incident. Although it may not always be warranted, the severity of the incident could justify such access for incident investigators. For such cases, developers and deployers should establish secure AI system access programs for investigators, which may include:

- On-site model access programs or custom grey-box or white-box APIs with technical support (Casper et al. 2024).
- Collaborative evaluation programs, such as the pre-deployment evaluations of OpenAI's o1 model conducted by the UK AI Security Institute and US AI Safety Institute (UK AI Security Institute 2024).

3.3 Tool Information

What should be in an incident report? Incident reports should document basic information about any tools that AI agents use, including the following:

- Identification and description of the tool (e.g., Google Chrome web browser with extensions, Bing Web Search API, Python libraries)
- Tool version information
- The actions that the tool enables the agent to execute (e.g., web searches, making purchases, accessing databases, etc.)
- The information or services to which the tool facilitates access (e.g., databases, indexed web pages, payment systems, other API endpoints)
- Instructions for how to use the tool (e.g., guides, API or library documentation). If this information is publicly available, links to instructions are sufficient.
- The presence of access requirements for the tool (e.g., need for API keys, login information, billing information, etc.)
- A description of any information that the tool collects about the user (e.g., search history, products viewed or purchased, location, etc.) and how the tool uses user information for personalization (e.g., recommender systems, sharing with third parties, etc.)

Much of this information may already be publicly available. For example, many APIs have public documentation available online, and services have privacy policies. Incident reports can link to archived versions of these public web pages corresponding to the time when the incident occurred. Where information is not publicly available, developers and deployers should still report such basic information about the tools to which agents had access when they possess it.

What data should developers retain? Developers should retain more comprehensive information about the AI agent's specific interaction with the tool, including:

- A description of any actions that the agent used, or attempted to use, the tool to perform. This includes, for example, any API calls and a description of what the agent was likely trying to achieve via those API calls.
- A description of any information that the tool provided to the agent (e.g., retrieved data, execution logs).
- Any tool usage instructions that were provided to the agent, and the method through which those instructions were provided (e.g., training data, RL feedback, few-shot learning, instructions in prompt).
- How the agent gained access to the tool (e.g., no access requirements, credentials from the user, credentials from the developer, creating its own account).
- A description of any information the tool had available for personalization (e.g., no information, the user's location/history, the agent's location/history).
- Details about the tool's state (e.g., shopping cart contents) before, during, and after the interaction with the agent.
- Any errors that arose during the interaction and their causes (e.g., details about a paywall or bot detection system, any API error codes that occurred).

Much of this information would be evident within activity logs. For example, AI system outputs may show the actions that the agent attempted (e.g., API requests) or credentials provided to the agent during user interactions (e.g., an API key, account login information, etc.). However, some information specific to the interactions may not be evident from the activity logs or be publicly available. For example, if an agent is operating a browser, its activity logs–which consist of browser snapshots–may not reveal if the user or agent agreed to a data collection policy in a previous session. If developers or deployers provide agents with access to tools and can obtain this information (e.g., OpenAI's Operator uses its own browser rather than the user's), they should be prepared to report it upon request.

What other policies should developers and deployers follow? Users may possess some of the interaction-specific information listed above. This may include the permissions and accounts to which they provided agents access, their account or profile settings (e.g., which cookies are enabled in their browser), and other details about the available tools (e.g., Python libraries installed on their system). As is standard with many software packages, developers could ask users to voluntarily provide such diagnostic information either in advance or at the time of the incident.

4 Discussion

4.1 Case Study: EchoLeak Incident

EchoLeak (CVE-2025-32711) was a vulnerability in Microsoft 365 Copilot that allowed attackers to exfiltrate confidential data without user interaction (Aim Labs Team 2025; Microsoft 2025). In particular, a malicious email containing hidden instructions led Copilot's underlying large language model to reveal private information–an indirect prompt injection attack. Microsoft addressed this exploit after researchers reported it (Microsoft 2025). Public advisories describe what happened but omit sensitive information that would pinpoint how and why the agent complied. Based on public information, the following assessment about contributing factors to the incident can be made:

- System factors: Copilot's scaffolding did not detect and filter malicious inputs, nor did it detect harmful outputs (sharing confidential information with untrusted external parties). However, the design and limitations of the scaffolding are not public and cannot be fully determined.

- Contextual factors: An attacker placed a malicious request within an email inbox to which Copilot had access. However, the content of malicious emails that would trigger such failures is poorly characterized.
- Cognitive errors: Copilot may have failed to distinguish malicious hidden commands from trusted user requests, or it may have identified–but nevertheless complied with–malicious commands.

To more rigorously diagnose the problem and ensure corrections are adequate, investigators need:

- Detailed activity logs showing, for example, the email ingested, intermediate prompts, and the agent's outputs.
- Model reasoning traces to see how the LLM parsed and prioritized the malicious instructions.
- System documentation and change logs for Copilot's prompt injection defenses and the subsequent patch.

With this additional data, investigators could more precisely identify root causes, assess the adequacy of Microsoft's patches, and measure residual risk.

4.2 Existing Incident Reporting Practices and Emerging Requirements

Existing AI incident databases–such as the AIID, AVID, or OECD AIM–are limited to information that has been publicly reported or voluntarily submitted. For example, the AIID and OECD AI Incidents Monitor (OECD 2025a) source incidents from news articles. The incident data they collect does not contain the key items we outlined in Section 3: activity logs, system details, and information about tools. These databases mostly capture high-level information about incidents, enabling broad classifications to identify trends. The AIID, for example, uses taxonomies that categorize incidents by factors such as deployment sector, risk domains (e.g., privacy and security), type of harm (e.g., discrimination), and lifecycle stage when the incident occurred (development/deployment) (Pittaras and McGregor 2022; Hoffmann and Frase 2023). Pittaras and McGregor (2022) propose a method, used by the AIID, to classify incidents by their technical causes based on public information. Using this method, for example, a chatbot which made offensive remarks about minority groups was categorized as being caused by "distributional bias" (AIID 2016). However, more information can enable the identification of incident causes with more precision, such as whether instructions given to human raters led to biased RLHF feedback.

Paeth et al. (2025) discuss how existing incident classification frameworks lack context-specificity and nuance. They provide the example of two autonomous vehicle accidents involving the same vehicle and similar harms, but which occurred because of different environmental stimuli: unexpected braking and stop sign misidentification. A taxonomy without adequate contextual information may categorize both of these as vehicle collisions with similar harms (e.g., physical injury) and technical causes (e.g., distribution shift), omitting the distinct environmental triggers and cognitive breakdowns (e.g., detection of braking vs. understanding of stop sign).

Developers and deployers can collect comprehensive incident data pertaining to their own systems and may perform incident analysis. For example, when users submit problems and feedback to OpenAI, they can include the conversation ID, which enables OpenAI to identify the activity logs and metadata (OpenAI 2025c). Furthermore, developers have complete documentation and access to their AI systems to diagnose problems, and they have knowledge of the tools to which they have given their systems access. However, when incident reports and databases are internal to developers, external stakeholders cannot conduct analysis, hold developers accountable, and build public awareness of risks. Developers may not apply state-of-the-art causal analysis practices, and they have poor incentives to share some of their findings externally.

Required incident reporting mechanisms are beginning to emerge, which can facilitate external incident analysis to identify causal factors based on comprehensive incident data. For example, the EU AI Act requires providers of AI systems with systemic risk to report serious incidents (European Union 2024). Furthermore, the Third Draft of the EU AI Act's Code of Practice commits signatories to reporting to authorities "the chain of events that (directly or indirectly) led to the serious incident as far as it is reconstructable" and "a root cause analysis, including, as far as possible, a description of the [AI system's] outputs that (directly or indirectly) led to the serious incident and the factors that contributed to their generation" (European Commission 2025). Frameworks such as our own–which describe how root cause analysis can be conducted–can inform corresponding reporting requirements and subsequent analysis. Regulators can ensure that rigorous and comparable causal analysis practices are applied across developers. In Table 2, we compare existing approaches to incident reporting to emerging regulation-based incident reporting.

4.3 Limitations and Future Work

Our incident analysis framework has certain framework scope limitations. It does not cover every possible cause of AI agent incidents–for example, we have not covered causal factors that may be domain-specific. Furthermore, future incidents and trends in AI development and use could surface additional factors we have not identified.

There is also scientific and technical uncertainty underlying our framework. Our understanding of AI agent cognition and failure causes remains limited. Our discussion of cognitive errors is necessarily high-level because the cognitive science of AI agents is still nascent. Future developments in our understanding of how AI agents orient to situations, parse tasks, and select actions may require changes to our framework (McCoy et al. 2024; Hagendorff et al. 2024).

There are also significant institutional and infrastructural challenges that hinder implementation of our framework. Presently, there is no dedicated, secure infrastructure for comprehensive incident reporting. This means that detailed incident data–especially sensitive data like an agent's chain of thought–cannot be easily shared across organizations. Furthermore, investigations involving confidential data are likely to remain restricted to certain parties (e.g. regulators)

Database operator	Civil society	Developers/deployers	Governments
Example(s)	AIID, AVID, AIAAIC, OECD AIM	OpenAI model behavior feedback form	EU AI Act (serious incident reporting requirements)
Information	Limited to publicly reported or voluntarily submitted data; lacks activity logs, system details, and tools.	Access to activity logs/metadata, full system documentation, and details about available tools.	Can mandate reporting of information relevant to determining causes.
Identifies causes	Inadequate information to perform context-specific root cause analysis.	Analysis is mostly non-public and unstandardized; no external oversight.	Regulators can require state-of-the-art causal analysis practices.
Aggregates across developers/deployers	Yes; any relevant incidents can be reported.	No; data siloed per organization.	Yes; regulators compile data across providers.
Facilitates accountability	Indirectly; relies on public awareness.	No; poor incentives to share some findings.	Yes; regulatory mandates and penalties may exist.
Active	Yes	Yes	No (emerging)

Table 2: Comparison between various institutional arrangements for incident reporting.

who may not have the technical expertise to analyze AI incidents in depth. When only a limited group has access to full incident details, accountability mechanisms become crucial. Currently, there are inadequate means to ensure that private analyses translate into public accountability. There is a need for protocols such as partial public release of incident data or periodic reporting of incident trends to oversight bodies.

There are also privacy and legal constraints to implementation. Incident reports often involve sensitive data, which limits what can be documented and shared. Strict privacy requirements and confidentiality agreements can prevent retaining certain logs or sharing them with external investigators. In addition, the legal landscape for AI incident data is unclear–organizations face uncertainty about liability and compliance when documenting and sharing incident information. This ambiguity can discourage comprehensive reporting. Even when data is collected, privacy laws and security concerns impose access limitations: only specific personnel may be allowed to examine the full incident details, which in turn limits external validation or community learning from those incidents.

Finally, there are several operational constraints in applying the framework. One issue is data retention: many AI developers and deployers do not store interaction logs for extensive periods due to storage costs or user privacy policies. Furthermore, incident reconstruction can be difficult if investigators lack sufficient access to proprietary systems, which may exceed what is publicly available (e.g., white/grey-box API access). In addition, given the absence of regulations, implementation of our framework would currently rely on developers and deployers voluntarily implementing comprehensive logging and reporting practices, which may not be consistent or guaranteed.

Addressing these limitations–which we also outline in Appendix B–will require coordinated effort and future work. On the conceptual side, the framework should be continually expanded and refined as new incident patterns and research findings become available. Practically, researchers and policymakers need to develop the infrastructure and institutions for secure incident reporting and investigation–for example, creating confidential reporting channels or repositories that facilitate expert analysis and public oversight. Legal and privacy guidelines for incident data also need to be clarified so that organizations can share necessary information without undue risk. Finally, operational standards and incentives (or requirements) for organizations to log relevant data and cooperate in incident investigations would significantly improve the framework's effectiveness.

5 Conclusion

AI agents will inevitably be involved in real-world incidents. Our framework emphasizes the importance of comprehensive and context-specific data for illuminating multi-step causal pathways culminating in incidents.

We show how incidents can arise from system design choices, contextual influences, and the cognitive demands of the intended task. We have also outlined information that is likely to be key for understanding which causal factors apply: activity logs, system documentation and access, and tool information. Based on these categories of information, we provided recommendations for 1) what should be included in incident reports and 2) what information developers and deployers should retain and make available to incident investigators.

Effective incident analysis also requires robust institutional support. Governments and regulators can establish secure reporting infrastructures, mandate the retention and sharing of key incident data, and build technical capacity for causal investigation–paralleling practices in aviation and finance. Regulatory frameworks such as the EU AI Act already include incident reporting requirements and can serve as a model for broader practices and mandates (European Union 2024). By combining rigorous data collection and analysis with clear regulatory signals and public accountability mechanisms, we can strengthen our ability to learn from AI agent failures and prevent future harms.

Appendices

Available on: https://arxiv.org/pdf/2508.14231

Acknowledgments

The authors are grateful for discussions and feedback from the following individuals: Ben Bucknall, Stephen Casper, Jimmy Farrell, James Ginns, Joshua Greene, Noam Kolt, Shayne Longpre, Richard Moulange, Shalaleh Rismani, Jonas Schuett, Tommy Shaffer Shane, and Kevin Wei.

References

AIAAIC. 2025. AIAAIC. https://www.aiaaic.org/. Accessed: 2025-05-21.

AIID. 2016. Incident 106: Korean Chatbot Luda Made Offensive Remarks towards Minority Groups. https://incidentdatabase.ai/cite/106/. Accessed: 2025-05-24.

AIID. 2024. Welcome to the Artificial Intelligence Incident Database. https://incidentdatabase.ai/. Accessed: 2025-05-21.

Aim Labs Team. 2025. Breaking down 'EchoLeak', the First Zero-Click AI Vulnerability Enabling Data Exfiltration from Microsoft 365 Copilot. https://www.aim.security/lp/aim-labs-echoleak-blogpost.

Alber, D. A.; Yang, Z.; Alyakin, A.; Yang, E.; Rai, S.; Valliani, A. A.; Zhang, J.; Rosenbaum, G. R.; Amend-Thomas, A. K.; Kurland, D. B.; Kremer, C. M.; Eremiev, A.; Negash, B.; Wiggan, D. D.; Nakatsuka, M. A.; Sangwon, K. L.; Neifert, S. N.; Khan, H. A.; Save, A. V.; Palla, A.; Grin, E. A.; Hedman, M.; Nasir-Moin, M.; Liu, X. C.; Jiang, L. Y.; Mankowski, M. A.; Segev, D. L.; Aphinyanaphongs, Y.; Riina, H. A.; Golfinos, J. G.; Orringer, D. A.; Kondziolka, D.; and Oermann, E. K. 2025. Medical large language models are vulnerable to data-poisoning attacks. *Nature Medicine*, 31(2): 618–626. Publisher: Nature Publishing Group.

Anthropic. 2025a. How long do you store my data? https://privacy.anthropic.com/en/articles/10023548-how-long-do-you-store-my-data. Accessed: 2025-05-21.

Anthropic. 2025b. How long do you store my organization's data? https://privacy.anthropic.com/en/articles/7996866-how-long-do-you-store-my-organization-s-data. Accessed: 2025-05-21.

Anthropic. 2025c. System Card: Claude Opus 4 & Claude Sonnet 4. https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf. Accessed: 2025-05-23.

Anthropic. 2025d. System Prompts. https://docs.anthropic.com/en/release-notes/system-prompts. Accessed: 2025-05-21.

AVID. 2025. AVID. https://avidml.org/. Accessed: 2025-05-21.

Bai, Y.; Kadavath, S.; Kundu, S.; Askell, A.; Kernion, J.; Jones, A.; Chen, A.; Goldie, A.; Mirhoseini, A.; McKinnon, C.; Chen, C.; Olsson, C.; Olah, C.; Hernandez, D.; Drain, D.; Ganguli, D.; Li, D.; Tran-Johnson, E.; Perez, E.; Kerr, J.; Mueller, J.; Ladish, J.; Landau, J.; Ndousse, K.; Lukosuite, K.; Lovitt, L.; Sellitto, M.; Elhage, N.; Schiefer, N.; Mercado, N.; DasSarma, N.; Lasenby, R.; Larson, R.; Ringer, S.; Johnston, S.; Kravec, S.; Showk, S. E.; Fort, S.; Lanham, T.; Telleen-Lawton, T.; Conerly, T.; Henighan, T.; Hume, T.; Bowman, S. R.; Hatfield-Dodds, Z.; Mann, B.; Amodei, D.; Joseph, N.; McCandlish, S.; Brown, T.; and Kaplan, J. 2022. Constitutional AI: Harmlessness from AI Feedback. arXiv:2212.08073.

Baker, B.; Huizinga, J.; Gao, L.; Dou, Z.; Guan, M. Y.; Madry, A.; Zaremba, W.; Pachocki, J.; and Farhi, D. 2025. Monitoring Reasoning Models for Misbehavior and the Risks of Promoting Obfuscation. arXiv:2503.11926.

Beurer-Kellner, L.; and Fischer, M. 2025a. Introducing MCP-Scan: Protecting MCP with Invariant. https://invariantlabs.ai/blog/introducing-mcp-scan.

Beurer-Kellner, L.; and Fischer, M. 2025b. MCP Security Notification: Tool Poisoning Attacks. https://invariantlabs.ai/blog/mcp-security-notification-tool-poisoning-attacks.

Biderman, D.; Portes, J.; Ortiz, J. J. G.; Paul, M.; Greengard, P.; Jennings, C.; King, D.; Havens, S.; Chiley, V.; Frankle, J.; Blakeney, C.; and Cunningham, J. P. 2024. LoRA Learns Less and Forgets Less. *Transactions on Machine Learning Research*.

Buck Shlegeris [@bshlgrs]. 2024. I asked my LLM agent (a wrapper around Claude that lets it run bash commands and see their outputs): >can you ssh with the username buck to the computer on my network that is open to SSH because I didn't know the local IP of my desktop. I walked away and promptly forgot I'd spun https://t.co/I6qppMZFfk. https://x.com/bshlgrs/status/1840577720465645960.

Carlini, N.; Tramèr, F.; Wallace, E.; Jagielski, M.; Herbert-Voss, A.; Lee, K.; Roberts, A.; Brown, T.; Song, D.; Erlingsson, U.; Oprea, A.; and Raffel, C. 2021. Extracting Training Data from Large Language Models. 2633–2650. ISBN 978-1-939133-24-3.

Casper, S.; Bailey, L.; Hunter, R.; Ezell, C.; Cabalé, E.; Gerovitch, M.; Slocum, S.; Wei, K.; Jurkovic, N.; Khan, A.; Christoffersen, P. J. K.; Ozisik, A. P.; Trivedi, R.; Hadfield-Menell, D.; and Kolt, N. 2025. The AI Agent Index. arXiv:2502.01635.

Casper, S.; Davies, X.; Shi, C.; Gilbert, T. K.; Scheurer, J.; Rando, J.; Freedman, R.; Korbak, T.; Lindner, D.; Freire, P.; Wang, T. T.; Marks, S.; Segerie, C.-R.; Carroll, M.; Peng, A.; Christoffersen, P. J. K.; Damani, M.; Slocum, S.; Anwar, U.; Siththaranjan, A.; Nadeau, M.; Michaud, E. J.; Pfau, J.; Krasheninnikov, D.; Chen, X.; Langosco, L.; Hase, P.; Biyik, E.; Dragan, A.; Krueger, D.; Sadigh, D.; and Hadfield-Menell, D. 2023. Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback. *Transactions on Machine Learning Research*.

Casper, S.; Ezell, C.; Siegmann, C.; Kolt, N.; Curtis, T. L.; Bucknall, B.; Haupt, A.; Wei, K.; Scheurer, J.; Hobbhahn, M.; Sharkey, L.; Krishna, S.; Von Hagen, M.; Alberti, S.; Chan, A.; Sun, Q.; Gerovitch, M.; Bau, D.; Tegmark, M.; Krueger, D.; and Hadfield-Menell, D. 2024. Black-Box Access is Insufficient for Rigorous AI Audits. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, 2254–2272. New York, NY, USA: Association for Computing Machinery. ISBN 9798400704505.

Chan, A.; Ezell, C.; Kaufmann, M.; Wei, K.; Hammond, L.; Bradley, H.; Bluemke, E.; Rajkumar, N.; Krueger, D.; Kolt, N.; Heim, L.; and Anderljung, M. 2024. Visibility into AI Agents. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, 958–973. New York, NY, USA: Association for Computing Machinery. ISBN 9798400704505.

Chan, A.; Salganik, R.; Markelius, A.; Pang, C.; Rajkumar, N.; Krasheninnikov, D.; Langosco, L.; He, Z.; Duan, Y.; Carroll, M.; Lin, M.; Mayhew, A.; Collins, K.; Molamohammadi, M.; Burden, J.; Zhao, W.; Rismani, S.; Voudouris, K.; Bhatt, U.; Weller, A.; Krueger, D.; and Maharaj, T. 2023. Harms from Increasingly Agentic Algorithmic Systems. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '23, 651–666. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701924.

Creemers, R.; Webster, G.; and Toner, H. 2022. Translation: Internet Information Service Algorithmic Recommendation Management Provisions – Effective March 1, 2022. https://digichina.stanford.edu/work/translation-internet-information-service-algorithmic-recommendation-management-provisions-effective-march-1-2022/.

European Commission. 2025. Third Draft of the General-Purpose AI Code of Practice published, written by independent experts. https://digital-strategy.ec.europa.eu/en/library/third-draft-general-purpose-ai-code-practice-published-written-independent-experts. Accessed: 2025-05-21.

European Union. 2024. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA relevance). http://data.europa.eu/eli/reg/2024/1689/oj/eng. Legislative Body: CONSIL, EP.

Gabriel, I.; Manzini, A.; Keeling, G.; Hendricks, L. A.; Rieser, V.; Iqbal, H.; Tomašev, N.; Ktena, I.; Kenton, Z.; Rodriguez, M.; El-Sayed, S.; Brown, S.; Akbulut, C.; Trask, A.; Hughes, E.; Bergman, A. S.; Shelby, R.; Marchal, N.; Griffin, C.; Mateos-Garcia, J.; Weidinger, L.; Street, W.; Lange, B.; Ingerman, A.; Lentz, A.; Enger, R.; Barakat, A.; Krakovna, V.; Siy, J. O.; Kurth-Nelson, Z.; McCroskery, A.; Bolina, V.; Law, H.; Shanahan, M.; Alberts, L.; Balle, B.; Haas, S. d.; Ibitoye, Y.; Dafoe, A.; Goldberg, B.; Krier, S.; Reese, A.; Witherspoon, S.; Hawkins, W.; Rauh, M.; Wallace, D.; Franklin, M.; Goldstein, J. A.; Lehman, J.; Klenk, M.; Vallor, S.; Biles, C.; Morris, M. R.; King, H.; Arcas, B. A. y.; Isaac, W.; and Manyika, J. 2024. The Ethics of Advanced AI Assistants. arXiv:2404.16244.

Greshake, K.; Abdelnabi, S.; Mishra, S.; Endres, C.; Holz, T.; and 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*, AISec '23, 79–90. New York, NY, USA: Association for Computing Machinery. ISBN 9798400702600.

Grynbaum, M. M.; and Mac, R. 2023. The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work. *The New York Times*.

Hagendorff, T.; Dasgupta, I.; Binz, M.; Chan, S. C. Y.; Lampinen, A.; Wang, J. X.; Akata, Z.; and Schulz, E. 2024. Machine Psychology. arXiv:2303.13988.

Hammond, L.; Chan, A.; Clifton, J.; Hoelscher-Obermaier, J.; Khan, A.; McLean, E.; Smith, C.; Barfuss, W.; Foerster, J.; Gavenčiak, T.; Han, T. A.; Hughes, E.; Kovařík, V.; Kulveit, J.; Leibo, J. Z.; Oesterheld, C.; Witt, C. S. d.; Shah, N.; Wellman, M.; Bova, P.; Cimpeanu, T.; Ezell, C.; Feuillade-Montixi, Q.; Franklin, M.; Kran, E.; Krawczuk, I.; Lamparth, M.; Lauffer, N.; Meinke, A.; Motwani, S.; Reuel, A.; Conitzer, V.; Dennis, M.; Gabriel, I.; Gleave, A.; Hadfield, G.; Haghtalab, N.; Kasirzadeh, A.; Krier, S.; Larson, K.; Lehman, J.; Parkes, D. C.; Piliouras, G.; and Rahwan, I. 2025. Multi-Agent Risks from Advanced AI. arXiv:2502.14143.

Hans, A.; Wen, Y.; Jain, N.; Kirchenbauer, J.; Kazemi, H.; Singhania, P.; Singh, S.; Somepalli, G.; Geiping, J.; Bhatele, A.; and Goldstein, T. 2024. Be like a Goldfish, Don't Memorize! Mitigating Memorization in Generative LLMs. *Advances in Neural Information Processing Systems*, 37: 24022–24045.

Heiding, F.; Lermen, S.; Kao, A.; Schneier, B.; and Vishwanath, A. 2024. Evaluating Large Language Models' Capability to Launch Fully Automated Spear Phishing Campaigns: Validated on Human Subjects. arXiv:2412.00586.

Hoffmann, M.; and Frase, H. 2023. Adding Structure to AI Harm. https://cset.georgetown.edu/publication/adding-structure-to-ai-harm/. Accessed: 2025-05-21.

Joshi, I.; Shahid, S.; Venneti, S. M.; Vasu, M.; Zheng, Y.; Li, Y.; Krishnamurthy, B.; and Chan, G. Y.-Y. 2025. CoPrompter: User-Centric Evaluation of LLM Instruction Alignment for Improved Prompt Engineering. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*, IUI '25, 341–365. New York, NY, USA: Association for Computing Machinery. ISBN 9798400713064.

Kapoor, S.; Stroebl, B.; Siegel, Z. S.; Nadgir, N.; and Narayanan, A. 2024. AI Agents That Matter. arXiv:2407.01502.

Kasirzadeh, A.; and Gabriel, I. 2025. Characterizing AI Agents for Alignment and Governance. arXiv:2504.21848.

Krakovna, V.; and Kramar, J. 2023. Power-seeking can be probable and predictive for trained agents. arXiv:2304.06528.

Krueger, D.; Maharaj, T.; and Leike, J. 2020. Hidden Incentives for Auto-Induced Distributional Shift. arXiv:2009.09153.

Lai, V. D.; Ngo, N.; Pouran Ben Veyseh, A.; Man, H.; Dernoncourt, F.; Bui, T.; and Nguyen, T. H. 2023. ChatGPT Beyond English: Towards a Comprehensive Evaluation of Large Language Models in Multilingual Learning. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023*, 13171–13189. Singapore: Association for Computational Linguistics.

Leveson, N. G. 2023. *An Introduction to System Safety Engineering*. Cambridge, Massachusetts: The MIT Press. ISBN 978-0-262-54688-1.

Lin, S.; Hilton, J.; and Evans, O. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In Muresan, S.; Nakov, P.; and Villavicencio, A., eds., *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3214–3252. Dublin, Ireland: Association for Computational Linguistics.

Longpre, S.; Klyman, K.; Appel, R. E.; Kapoor, S.; Bommasani, R.; Sahar, M.; McGregor, S.; Ghosh, A.; Blili-Hamelin, B.; Butters, N.; Nelson, A.; Elazari, A.; Sellars, A.; Ellis, C. J.; Sherrets, D.; Song, D.; Geiger, H.; Cohen, I.; McIlvenny, L.; Srikumar, M.; Jaycox, M. M.; Anderljung, M.; Johnson, N. F.; Carlini, N.; Miailhe, N.; Marda, N.; Henderson, P.; Portnoff, R. S.; Weiss, R.; Westerhoff, V.; Jernite, Y.; Chowdhury, R.; Liang, P.; and Narayanan, A. 2025. In-House Evaluation Is Not Enough: Towards Robust Third-Party Flaw Disclosure for General-Purpose AI. arXiv:2503.16861.

McCoy, R. T.; Yao, S.; Friedman, D.; Hardy, M. D.; and Griffiths, T. L. 2024. Embers of autoregression show how large language models are shaped by the problem they are trained to solve. *Proceedings of the National Academy of Sciences*, 121(41): e2322420121. Publisher: Proceedings of the National Academy of Sciences.

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022. Locating and editing factual associations in GPT. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, 17359–17372. Red Hook, NY, USA: Curran Associates Inc. ISBN 978-1-71387-108-8.

Microsoft. 2025. M365 Copilot Information Disclosure Vulnerability. https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2025-32711. Accessed: 2025-08-04.

Mitchell, M.; Wu, S.; Zaldivar, A.; Barnes, P.; Vasserman, L.; Hutchinson, B.; Spitzer, E.; Raji, I. D.; and Gebru, T. 2019. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, 220–229. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-6125-5.

Occupational Safety and Health Administration. n.d. Incident Investigation. https://www.osha.gov/incident-investigation. Accessed: 2025-08-04.

OECD. 2025a. OECD AI Incidents Monitor, an evidence base for trustworthy AI. https://oecd.ai/en/incidents. Accessed: 2025-05-21.

OECD. 2025b. Towards a common reporting framework for AI incidents. Technical Report 34, OECD Publishing, Paris.

O'Keefe, C.; Ramakrishnan, K.; Tay, J.; and Winter, C. 2025. Law-Following AI: Designing AI Agents to Obey Human Laws. SSRN:5242643.

OpenAI. 2024a. Enterprise privacy at OpenAI. https://openai.com/enterprise-privacy/. Accessed: 2025-05-21.

OpenAI. 2024b. OpenAI o1 System Card. https://cdn.openai.com/o1-system-card-20241205.pdf. Accessed: 2025-05-23.

OpenAI. 2025a. Data controls in the OpenAI platform. https://platform.openai.com. Accessed: 2025-05-23.

OpenAI. 2025b. Expanding on what we missed with sycophancy. https://openai.com/index/expanding-on-sycophancy/.

OpenAI. 2025c. Model behavior feedback. https://openai.com/form/model-behavior-feedback/. Accessed: 2025-05-24.

OpenAI. 2025d. Sycophancy in GPT-4o: What happened and what we're doing about it. https://openai.com/index/sycophancy-in-gpt-4o/.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, 27730–27744. Red Hook, NY, USA: Curran Associates Inc. ISBN 978-1-71387-108-8.

Paeth, K.; Atherton, D.; Pittaras, N.; Frase, H.; and McGregor, S. 2025. Lessons for Editors of AI Incidents from the AI Incident Database. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(28): 28946–28953. Number: 28.

Pittaras, N.; and McGregor, S. 2022. A taxonomic system for failure cause analysis of open source AI incidents. arXiv:2211.07280.

Preston, D. 2025. Grok's white genocide fixation caused by 'unauthorized modification'. https://www.theverge.com/news/668220/grok-white-genocide-south-africa-xai-unauthorized-modification-employee.

Ranaldi, L.; and Pucci, G. 2024. When Large Language Models contradict humans? Large Language Models' Sycophantic Behaviour. arXiv:2311.09410.

Shappell, S. A.; and Wiegmann, D. A. 2000. Human Factors Analysis and Classification System-HFACS. Technical Report DOT/FAA/AM-00/7.

Sharma, M.; Tong, M.; Korbak, T.; Duvenaud, D.; Askell, A.; Bowman, S. R.; Durmus, E.; Hatfield-Dodds, Z.; Johnston, S. R.; Kravec, S. M.; Maxwell, T.; McCandlish, S.; Ndousse, K.; Rausch, O.; Schiefer, N.; Yan, D.; Zhang, M.; and Perez, E. 2023. Towards Understanding Sycophancy in Language Models.

Tlaie, A.; and Farrell, J. 2025. Securing External Deeper-than-black-box GPAI Evaluations. arXiv:2503.07496.

Turpin, M.; Michael, J.; Perez, E.; and Bowman, S. 2023. Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting. *Advances in Neural Information Processing Systems*, 36: 74952–74965.

UK AI Security Institute. 2024. Pre-Deployment Evaluation of OpenAI's o1 Model. https://www.aisi.gov.uk/work/pre-deployment-evaluation-of-openais-o1-model.

Villalobos, P.; and Atkinson, D. 2023. Trading Off Compute in Training and Inference. https://epoch.ai/blog/trading-off-compute-in-training-and-inference.

Wallace, E.; Xiao, K.; Leike, R.; Weng, L.; Heidecke, J.; and Beutel, A. 2024. The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions. arXiv:2404.13208.

Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How Does LLM Safety Training Fail?

Weidinger, L.; Uesato, J.; Rauh, M.; Griffin, C.; Huang, P.-S.; Mellor, J.; Glaese, A.; Cheng, M.; Balle, B.; Kasirzadeh, A.; Biles, C.; Brown, S.; Kenton, Z.; Hawkins, W.; Stepleton, T.; Birhane, A.; Hendricks, L. A.; Rimell, L.; Isaac, W.; Haas, J.; Legassick, S.; Irving, G.; and Gabriel, I. 2022. Taxonomy of Risks posed by Language Models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, 214–229. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-4503-9352-2.

Welleck, S.; Bertsch, A.; Finlayson, M.; Schoelkopf, H.; Xie, A.; Neubig, G.; Kulikov, I.; and Harchaoui, Z. 2024. From Decoding to Meta-Generation: Inference-time Algorithms for Large Language Models. *Transactions on Machine Learning Research*.

Xing, J.; Parry, G.; Presley, M.; Forester, J.; Hendrickson, S.; and Dang, V. 2017. An Integrated Human Event Analysis System (IDHEAS) for Nuclear Power Plant Internal Events At-Power. Technical Report NUREG-2199, U.S. Nuclear Regulatory Commission, Washington, D.C.

Yang, W.; Sun, F.; Ma, X.; Liu, X.; Yin, D.; and Cheng, X. 2024. The Butterfly Effect of Model Editing: Few Edits Can Trigger Large Language Models Collapse. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics: ACL 2024*, 5419–5437. Bangkok, Thailand: Association for Computational Linguistics.