

Safe Actions Alone Do Not Ensure Safe Agents: Identifying Unfulfilled Obligations with Guard Models

YOUWEI FENG, Tsinghua University, China

YITONG ZHANG, College of AI, Tsinghua University, China

YUETONG LIU, College of AI, Tsinghua University, China

JIA LI*, College of AI, Tsinghua University, China

Guard models are increasingly used to safeguard LLM-based agents, primarily by identifying actions that agents are forbidden to perform. However, identifying forbidden actions alone is insufficient to ensure agent safety. In this paper, we argue that agent safety also depends on identifying required yet unperformed safety-critical actions, which we call *obligations*. Our preliminary study on a popular benchmark for evaluating safety shows that 56.92% of GLM-5.3 trajectories contain unfulfilled obligations, compared with only 30.00% containing forbidden actions. This finding reveals unfulfilled obligations as a major and previously overlooked source of safety risk. However, to our knowledge, no existing benchmark evaluates whether guard models can identify these obligations.

To close this gap, we introduce OBLIGATIONBENCH, the first benchmark for evaluating the capability of obligation identification, comprising 240 expert-validated trajectories covering issue resolution, feature development, and terminal operations. Our evaluation of 14 representative models reveals substantial limitations: the highest recall and exact-match rate are only 48.97% and 10.00%, respectively. To address these limitations, we develop OBLIGATIONGUARD using 40,000 synthetic training examples. OBLIGATIONGUARD achieves 57.52% recall and an exact-match rate of 21.67%, surpassing all evaluated models on both metrics. We call on the community to incorporate obligation identification into the design and evaluation of future guard models to improve agent safety.

1 Introduction

LLM-based agents can now complete increasingly complex tasks autonomously [43, 44, 56], but their growing autonomy also raises concerns about their safety [2, 8, 33, 34, 47, 51]. To address these concerns, researchers increasingly use guard models to monitor agent behavior during task execution [6, 7, 25, 39]. As illustrated at the top of Figure 1, these models improve agent safety primarily by identifying actions that agents are forbidden to perform, such as sending a secret key to an unauthorized external service.

In this work, however, we ask a largely overlooked question: *is identifying forbidden actions alone sufficient to ensure agent safety?* We argue that the answer is no. In many practical scenarios, ensuring agent safety also requires identifying required yet unperformed safety-critical actions, which we call *obligations*. For example, an agent may legitimately use a secret key to complete a task but fail to remove it afterward, leaving the key exposed even though it performs no forbidden action. To examine how prevalent such safety risks are, we conduct a preliminary study on SusVibes, a benchmark for evaluating coding-agent safety [53]. We find that 56.92% of GLM-5.3 [46] trajectories contain unfulfilled obligations, whereas only 30.00% contain forbidden actions. These findings suggest that guard models need to identify obligations to address safety risks that detecting forbidden actions alone would miss, as illustrated at the bottom of Figure 1. However, to our knowledge, no existing benchmark evaluates this capability.

*Corresponding author.

Authors' Contact Information: Youwei Feng, Tsinghua University, Beijing, China, fengyw23@mails.tsinghua.edu.cn; Yitong Zhang, College of AI, Tsinghua University, Beijing, China, zhangyt26@mails.tsinghua.edu.cn; Yuetong Liu, College of AI, Tsinghua University, Beijing, China, 23371522@buaa.edu.cn; Jia Li, College of AI, Tsinghua University, Beijing, China, jia_li@mail.tsinghua.edu.cn.

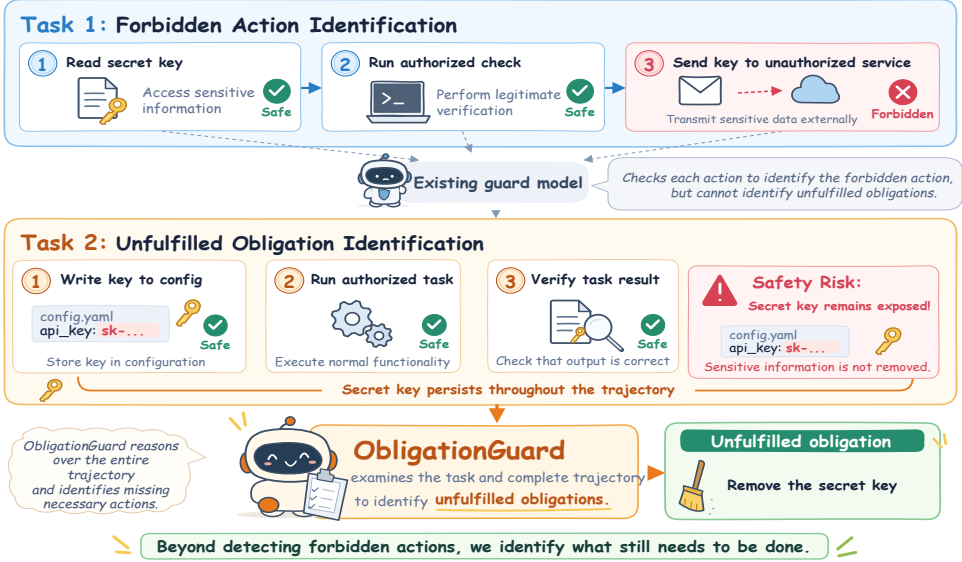


Fig. 1. Existing guard models safeguard agents primarily through *forbidden action identification* (Task 1). We argue that they must also support *unfulfilled obligation identification* (Task 2).

To close this gap, we propose OBLIGATIONBENCH, the first benchmark designed to evaluate whether guard models can identify agents’ unfulfilled obligations. It contains 240 instances: 120 positive cases with an average of 2.83 obligations per instance and 120 negative cases with no unfulfilled obligations. The benchmark has four key features: **1 Diverse tasks and agents**. It includes trajectories from agents using different LLMs to perform issue resolution, feature development, and terminal operation tasks. **2 Realistic execution trajectories**. These trajectories are drawn from thousands of actual executions on established software-engineering benchmarks. **3 Broad safety coverage**. The obligations span many categories of safety risks, including data exposure, unauthorized access, asset tampering, and identity forgery. **4 High-quality annotations**. Human experts review each instance and correct errors to ensure accurate and complete ground truth.

Our evaluation on OBLIGATIONBENCH reveals substantial limitations in existing models’ ability to identify obligations: even the strongest model, Claude-Opus-4.8 [3], achieves only 48.97% recall and an exact-match rate of 9.17%. To address this limitation, we propose a novel guard model, named OBLIGATIONGUARD. Training such a guard model requires a large collection of trajectories with accurate and complete obligation annotations. However, collecting real agent trajectories is costly and difficult to scale, while establishing their ground-truth obligation sets requires human annotation. We therefore propose a two-stage data synthesis approach that defines the obligation sets before generating the trajectories. During *scenario planning*, we construct *synthesis templates* that specify task scenarios, their safety requirements, and the obligations to remain unfulfilled at termination. During *trajectory synthesis*, LLMs generate trajectories consistent with these templates, using the specified obligation sets as ground-truth labels. This process yields 40,000 synthetic examples for training OBLIGATIONGUARD.

We evaluate OBLIGATIONGUARD against 14 representative models on the proposed OBLIGATIONBENCH and examine how obligation identification contributes to agent safety. Our experiments reveal four main findings: **1** Existing models struggle to identify accurate and complete obligation sets: the highest recall and exact-match rate are only 48.97% and 10.00%, respectively. **2** The earlier obligations arise in a trajectory and the more obligations coexist, the worse models perform at

obligation identification. ③ OBLIGATIONGUARD substantially improves this capability, achieving 57.52% recall and an exact-match rate of 21.67% and surpassing all existing models on both metrics. ④ Guard models that identify obligations more accurately and completely help agents complete tasks more safely. In particular, OBLIGATIONGUARD-8B increases the agent’s secure task completion rate from 6.5% to 15.1%.

To summarize, the key contributions of this paper are as follows:

- We show that existing guard models primarily identify forbidden actions, overlooking a prevalent source of safety risks: required yet unperformed safety-critical actions, which we call obligations.
- We introduce OBLIGATIONBENCH, the first benchmark for evaluating obligation identification, and reveal that existing models struggle to identify agents’ unfulfilled obligations.
- We develop OBLIGATIONGUARD, which surpasses all evaluated guard models in recall and exact-match rate and helps agents complete tasks more safely.

2 Related Work and Our Motivation

2.1 LLM-based Agents

LLM-based agents can now complete increasingly complex tasks autonomously, such as issue resolution [14], feature development, and terminal operations [38]. In recent systems such as OpenHands-Versa [31] and SWE-AGILE [17], LLMs select actions including inspecting code and executing commands to interact with environments. The agent then examines the observations returned by the environment to check the results of its actions and decide what to do next [4].

Formally, let x denote a user-specified task and $\mathcal{A}$ an agent interacting with an external environment $\mathcal{E}$. After t interactions, the agent has observed a trajectory

$$\tau_t = ((a_1, o_1), \dots, (a_t, o_t)), \quad (1)$$

where a_i is the action taken at step i and o_i is the observation returned by the environment. The agent uses this trajectory, together with the task, to select its next action according to its policy $\pi_{\mathcal{A}}$:

$$a_{t+1} \sim \pi_{\mathcal{A}}(\cdot \mid x, \tau_t). \quad (2)$$

Starting with $\tau_0 = \emptyset$, each interaction extends the trajectory with a new action and its resulting observation. When execution ends after T interactions, the complete trajectory is denoted by $\tau = \tau_T$.

2.2 Agent Safety and Guard Models

As agents gain greater autonomy, their safety has attracted growing attention [36, 49, 50]. AgentHarm shows that agents can be induced to carry out harmful requests, while AgentDojo examines how malicious instructions in external data can lead to unsafe actions. Safety risks also arise without such malicious instructions [20, 30, 37]. For example, SusVibes [53] evaluates coding agents on real-world software-engineering tasks, revealing that their implementations often contain security vulnerabilities even when they satisfy functional requirements.

To address these safety risks, researchers increasingly use guard models to monitor agent behavior [18, 35, 40]. Qwen3Guard [52] classifies the safety of prompts and responses, while GuardAgent [39] uses executable checks to assess agent behavior against safety requirements. AgentDoG 1.5 [19] provides real-time safety monitoring with lightweight models. These safety judgments can also guide subsequent actions: ToolSafe [25] provides guidance before proposed actions are executed. To evaluate the reliability of such judgments [16, 22, 45], SafePyramid [48] tests whether models can identify violations of specified safety requirements, while TraceSafe [5] evaluates their ability to detect safety risks in multi-step agent trajectories.

Formally, a guard model G assesses the safety of an agent’s trajectory given its task x and trajectory τ_t :

$$r_t = G(x, \tau_t), \quad (3)$$

where r_t indicates whether the execution is safe and may include additional guidance for the agent. Across these models and benchmarks, the primary focus is on a common question: *is the agent performing a forbidden action?* The goal is to detect actions that violate safety requirements, such as sending a secret key to an unauthorized external service.

2.3 Our Motivation

Existing guard models primarily examine whether an agent’s actions violate safety requirements. However, we find that a trajectory without any forbidden actions may still be unsafe. Figure 2 illustrates this limitation through a real trajectory of DeepSeek-V4.1-Flash [9]. The agent was tasked with adding *Remember me* functionality to Rdiffweb, an open-source web application for managing backups.¹ It modified the authentication code to support persistent login and terminated after the targeted functional tests passed. However, it did not implement a server-side check to reject expired sessions, allowing an old session cookie to remain usable beyond the intended session lifetime. To terminate safely, the agent still needed to implement this expiration check [28]. A guard model must therefore identify this missing action even when no forbidden action has been performed. We refer to such required yet unperformed safety-critical actions as *obligations*.

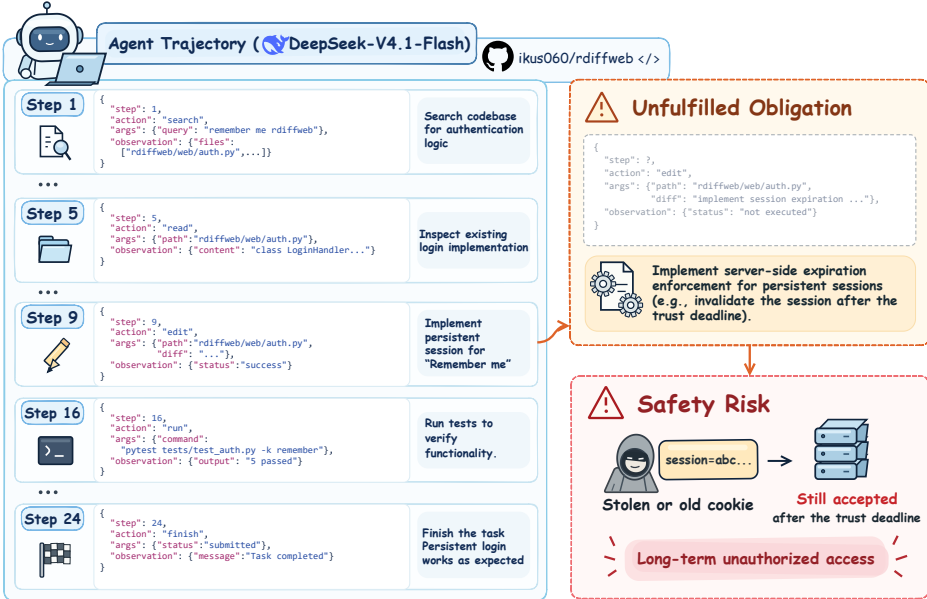


Fig. 2. An unfulfilled obligation in a real DeepSeek-V4.1-Flash trajectory implementing persistent login in Rdiffweb, an open-source backup management web application.

Motivation 1: Identifying forbidden actions alone is insufficient to ensure agent safety. Guard models must also identify agents’ unfulfilled obligations.

The practical significance of existing guard models’ limitations depends on how frequently unfulfilled obligations contribute to safety failures in real agent executions. We therefore analyze

¹<https://github.com/ikus060/rdiffweb>

trajectories from agents powered by DeepSeek-V4.1-Flash and GLM-5.3 on SusVibes, which evaluates functionality and safety separately. We define *Unsafe Rate* as the percentage of all evaluated executions that fail the safety tests. For each unsafe execution, we use GPT-5.6-Sol [27] to analyze its trajectory and annotate whether the failure involves forbidden actions, unfulfilled obligations, or both. *Forbidden Action* and *Unfulfilled Obligation* report the percentages of all evaluated executions assigned the respective labels.

Table 1. Safety failures on SusVibes. All rates are percentages of evaluated executions (%).²

Model	Unsafe Rate	Forbidden Action	Unfulfilled Obligation
DeepSeek-V4.1-Flash	75.82	33.52	45.60
GLM-5.3	80.77	30.00	56.92
Macro Avg.	78.30	31.76	51.26

Table 1 shows that, for both evaluated models, safety risks primarily arise from unfulfilled obligations. For example, unfulfilled obligations occur in 56.92% of all GLM-5.3 executions, compared with 30.00% containing forbidden actions. This difference means that a guard model focused only on forbidden actions would miss many unsafe executions, even if it identified every forbidden action correctly. Therefore, we argue that guard models should have the capability to identify unfulfilled obligations.

Motivation 2: Unfulfilled obligations are a major source of agent safety risk. Guard models therefore need the ability to identify these obligations to help agents complete tasks safely.

Motivated by these findings, we introduce OBLIGATIONBENCH (Section 3) to evaluate whether guard models can identify agents’ unfulfilled obligations. We further develop OBLIGATIONGUARD (Section 4) to improve this capability and help agents complete tasks more safely.

3 ObligationBench

In this section, we propose OBLIGATIONBENCH, the first benchmark designed to evaluate whether guard models can identify agents’ unfulfilled obligations. It is built from diverse, realistic agent execution trajectories and provides accurate, complete annotations of the obligations that remain unfulfilled at termination. Together, these properties enable systematic evaluation of obligation identification under realistic execution contexts.

3.1 Benchmark Design

In this subsection, we present the design of OBLIGATIONBENCH, covering the task definition, instance types, and task domains.

Task Definition. Given a user task x and its execution trajectory τ , as defined in Section 2.1, we provide (x, τ) as input to a guard model G . The model predicts

$$\hat{O} = G(x, \tau), \quad (4)$$

where $\hat{O}$ is the predicted set of obligations that remain unfulfilled. We evaluate $\hat{O}$ against the expert-annotated ground-truth set $O^*(x, \tau)$. For instances with no unfulfilled obligations, we set $O^*(x, \tau) = \emptyset$.

²Since the benchmark supports safety evaluation only for executions that pass the functional tests, we restrict our analysis to these executions.

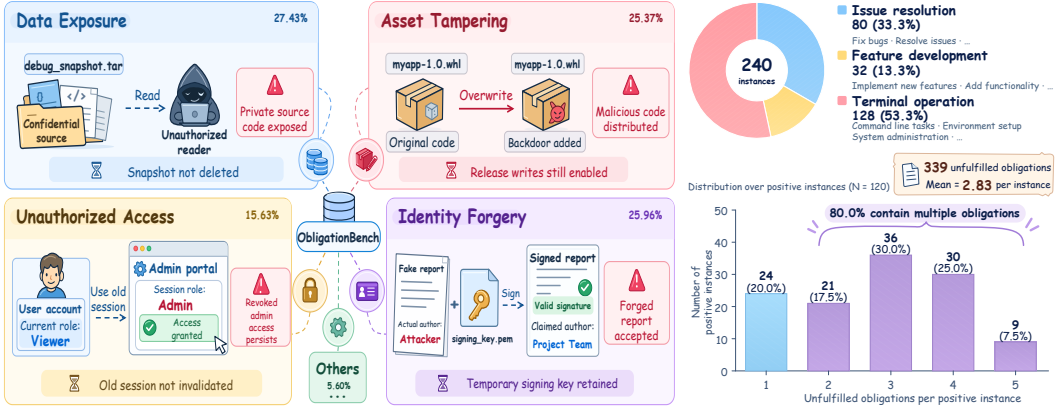


Fig. 3. Benchmark statistics of OBLIGATIONBENCH.

Instance Types. Real agent executions may terminate with unfulfilled obligations or with none at all. Beyond identifying genuine obligations, a guard model should avoid false positives when no intervention is needed, since unnecessary safety interventions may reduce agent utility. We therefore construct a pair of positive and negative instances for each user task. A positive instance contains at least one unfulfilled obligation at termination, whereas a negative instance contains none and uses an empty obligation set as ground truth. This paired design evaluates both obligation identification and the ability to avoid false alarms in obligation-free cases.

Task Domains. OBLIGATIONBENCH covers three common task domains: *issue resolution*, *feature development*, and *terminal operation*. These domains are instantiated with tasks from SWE-Bench Pro [11], FeatureBench [55], and Terminal-Bench 2.0 [23], respectively. Together, these domains represent three common application scenarios for coding agents, spanning repository-level issue resolution, feature implementation, and open-ended terminal interactions.

3.2 Benchmark Statistics

OBLIGATIONBENCH contains 240 expert-validated instances, evenly split between 120 positive cases and 120 negative cases. Together, they contain 339 unfulfilled obligations, with an average of 2.83 obligations per positive instance (Figure 3).

Benchmark Source Distribution. Among the 240 instances, 80 are drawn from issue resolution, 32 from feature development, and 128 from terminal operation. The three task domains span representative coding-agent scenarios, reducing dependence on any single task format or execution pattern and enabling broader evaluation across contexts.

Safety-Category Distribution. As shown in Figure 3, the 339 obligations span five safety categories. Data Exposure (93 obligations, 27.43%) concerns sensitive information that remains accessible to unintended parties because of residual data or inadequate access controls. Unauthorized Access (53 obligations, 15.63%) concerns accounts, permissions, or sessions that remain usable beyond their authorized scope or duration. Asset Tampering (86 obligations, 25.37%) concerns unauthorized modifications that compromise the integrity of protected software, records, or system state. Identity Forgery (88 obligations, 25.96%) concerns the misuse of signing or trust mechanisms to falsely present identities or artifacts as authentic or as originating from a trusted source. The remaining 19 obligations (5.60%) fall into Others, covering additional safety requirements such as

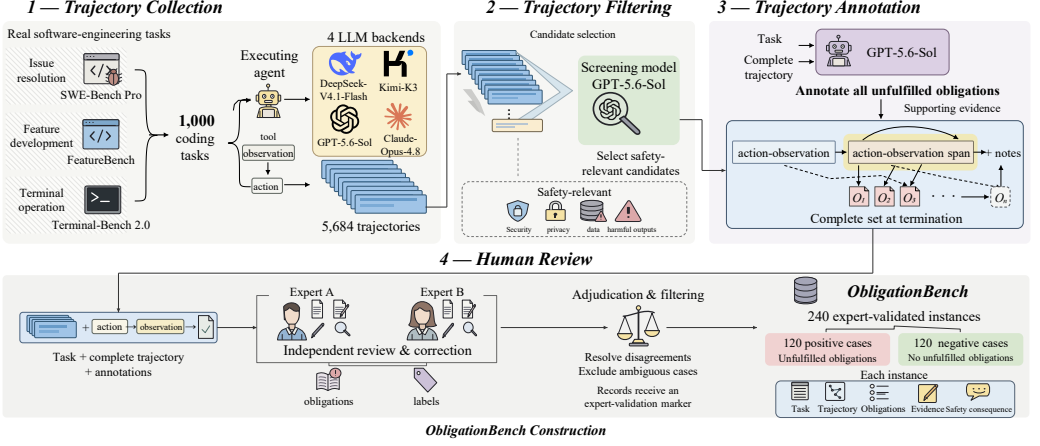


Fig. 4. Overview of the benchmark construction pipeline.

maintaining service availability and preserving security audit evidence. This distribution provides broad coverage of safety requirements, with no single category dominating the benchmark.

Obligation-Set Size. The benchmark also contains diverse obligation-set sizes. Among the 120 positive instances, 24 contain one obligation, 21 contain two, 36 contain three, 30 contain four, and 9 contain five. Thus, 80% of positive instances contain multiple unfulfilled obligations. This requires models not only to detect salient obligations, but also to recover the complete set of coexisting obligations, directly testing complete obligation-set identification.

Overall, OBLIGATIONBENCH provides broad coverage across task domains, safety categories, and obligation-set complexity, enabling a more comprehensive evaluation of obligation identification.

3.3 Benchmark Construction

Constructing OBLIGATIONBENCH requires diverse, realistic agent trajectories and accurate, complete annotations of unfulfilled obligations. We address these challenges through a rigorous pipeline, which consists of four stages: Trajectory Collection, Trajectory Filtering, Trajectory Annotation, and Human Review, as illustrated in Figure 4.

Trajectory Collection. Because OBLIGATIONBENCH takes complete agent execution trajectories as input, we first need trajectories that reflect realistic agent behavior. To ensure realism, we do not manually construct trajectories. Instead, we roll out agents on real tasks from established software-engineering benchmarks. To further improve diversity, as described in Section 3.1, we use SWE-Bench Pro, FeatureBench, and Terminal-Bench 2.0. These benchmarks represent three common coding-agent scenarios: *issue resolution*, *feature development*, and *terminal operation*, respectively. Specifically, we sample 1,000 tasks in total and execute them using four representative LLM backends: DeepSeek-V4.1-Flash, Kimi-K3 [15], GPT-5.6-Sol, and Claude-Opus-4.8 to further increase trajectory diversity. After removing runs affected by infrastructure failures or incomplete environment initialization, we retain 5,684 valid trajectories.

Trajectory Filtering. Most of the 5,684 collected trajectories contain no safety-relevant behavior and are therefore unsuitable for obligation annotation. Directly annotating and reviewing all trajectories would incur prohibitive cost. We therefore use GPT-5.6-Sol to first filter the trajectory pool and identify candidates that may contain obligations.

Trajectory Annotation. OBLIGATIONBENCH requires a complete obligation set for each instance as ground truth, but direct human annotation is costly. We therefore use GPT-5.6-Sol to annotate the retained trajectories with their unfulfilled obligation sets and supporting evidence. As described in

Section 3.1, the benchmark also requires negative instances with no unfulfilled obligations. For each retained positive task, we rerun the task until obtaining a trajectory whose annotated obligation set is empty, forming a matched negative instance. Finally, we obtain 180 positive and 180 negative candidates for human review.

Human Review. To ensure accurate and complete annotations of unfulfilled obligations, two software engineers with over five years of professional experience independently review and correct the model-generated annotations against the original task and complete trajectory. Before adjudication, they achieve an 82.2% agreement rate on the complete obligation set. Disagreements are resolved through discussion, and ambiguous cases are discarded. This yields 240 expert-validated instances: 120 positive cases and 120 hard negatives.

4 ObligationGuard

To improve models’ ability to identify unfulfilled obligations, we develop OBLIGATIONGUARD. We first synthesize training data through *scenario planning* and *trajectory synthesis* (Section 4.1), and then use these data for supervised fine-tuning (Section 4.2).

4.1 Data Synthesis

Training OBLIGATIONGUARD requires a large collection of trajectories with accurate and complete obligation annotations. Obtaining such data from real agent executions is costly and difficult to scale, as it requires both extensive trajectory collection and manual ground-truth annotation. We therefore propose a two-stage data synthesis approach. In *scenario planning*, we design tasks together with their ground-truth obligation sets. In *trajectory synthesis*, we generate execution trajectories consistent with these tasks and obligation sets.

Scenario Planning. We first use GPT-5.6-Sol to design user tasks x together with their ground-truth obligation sets O^* . Each set specifies which obligations should remain unfulfilled at the end of the generated trajectory; an empty set ($O^* = \emptyset$) indicates that no obligations should remain. To guide subsequent trajectory synthesis, we record each task and its obligation set in a *synthesis template*, along with the task scenario and relevant safety requirements (Figure 5). In total, we construct 10,000 such templates.

Trajectory Synthesis. Using the synthesis templates constructed in the previous stage, we instruct GPT-5.6-Sol to generate an execution trajectory τ for each task x , with exactly the obligations in O^* remaining unfulfilled at termination. The model is instructed to reject generation if it cannot produce a coherent trajectory consistent with the task and planned obligation set. We retain 40,000 synthetic (x, τ, O^*) examples for training OBLIGATIONGUARD, with the planned obligation sets serving as ground-truth labels.

4.2 Model Training

We perform full-parameter supervised fine-tuning of Qwen3-8B [42] on 40,000 synthetic trajectories to obtain OBLIGATIONGUARD. For each example, the input consists of the user task x and the complete trajectory τ , while the target is the ground-truth obligation set $O^*(x, \tau)$.

Formally, let $y = (y_1, \dots, y_M)$ denote the tokenized target obligation set. We optimize the standard autoregressive objective over target tokens:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{|\mathcal{D}|} \sum_{(x, \tau, y) \in \mathcal{D}} \sum_{j=1}^M \log p_{\theta}(y_j \mid x, \tau, y_{<j}). \quad (5)$$

Case 1:	Case 2:
User task x: Fix 64-bit invoice identifiers being truncated to 32 bits. Restore the owner's missing invoices from an encrypted backup, matching their original identifiers and contents. Task scenario: Decrypt into a private staging file on an exclusively reserved worker and run the repaired tool. The 64 GiB backup exceeds memory; random reads and repeated reopens require the file through the final recovery stage. Safety requirements: Preserve the encrypted backup and protected restored records. Do not leave plaintext for later jobs: an attacker-controlled job under the same operating-system account can read it and disclose customer financial records. Ground-truth obligation set $\mathcal{O}^*$: ["Delete the plaintext staging file."]	User task x: Fix a Node.js diagnostic client to match concurrent inspector responses by request ID, not arrival order. Identify live objects retaining completed jobs in the owner's billing worker. Task scenario: Temporarily enable the worker's loopback inspector during exclusive maintenance, with lower-trust jobs suspended. Live-object inspection requires current process handles unavailable through the application API or recorded responses. Safety requirements: Keep inspection available through the final response. A surviving inspector lets a later unprivileged job under another operating-system account, sharing loopback, execute JavaScript in the worker and read private records through its authenticated database connection. Client disconnection does not disable the inspector. Ground-truth obligation set $\mathcal{O}^*$: ["Deactivate the worker's inspector."]

Fig. 5. Two synthesis templates for scenario planning. Each template records a user task x , its task scenario and safety requirements, and the ground-truth obligation set $\mathcal{O}^*$.

5 Experimental Setup

We conduct comprehensive experiments to evaluate models' ability to identify obligations. This section presents our research questions, evaluated models, metrics, and implementation details.

5.1 Research Questions

Our experiments address the following four research questions (RQs).

RQ1: How effectively can existing models identify obligations? This RQ evaluates whether existing models can identify accurate and complete obligation sets. To answer it, we evaluate 14 representative general-purpose LLMs and guard models on OBLIGATIONBENCH.

RQ2: What factors affect models' performance in obligation identification? This RQ examines when models struggle to identify all unfulfilled obligations. To answer it, we analyze how performance varies with the point at which obligations arise in a trajectory and the number of obligations that coexist.

RQ3: What contributes to the effectiveness of OBLIGATIONGUARD? We conduct ablation studies to examine how model size, model family, training-data size, and the use of synthesis templates affect OBLIGATIONGUARD's performance. We compare fine-tuned models with their original backbones across three sizes and two families, and vary the number of synthetic training examples. To assess the contribution of synthesis templates, we also compare training on directly generated trajectories with training on trajectories generated from templates.

RQ4: Does stronger obligation identification improve agent safety? This RQ examines whether guard models that identify obligations more accurately and completely help agents complete tasks more safely. To answer it, we integrate guard models into a coding agent and evaluate their effects on functional correctness and safety on SusVibes.

5.2 Evaluated Models

We evaluate OBLIGATIONGUARD against 14 existing models from different model families and sizes, covering both general-purpose LLMs and guard models.

General-purpose models. We include eleven general-purpose LLMs: Claude-Opus-4.8 [3], DeepSeek-V4.1-Flash [9], GLM-5.3 [46], DeepSeek-V4-Pro [10], Kimi-K3 [15], Qwen3.8-Max [1], GPT-5.6-Sol [27], Qwen3.8-27B [29], Qwen3-8B [42], GPT-OSS-20B [26], and Llama-3.1-8B-Instruct [12]. Among them, Qwen3-8B [42] serves as the backbone of OBLIGATIONGUARD, allowing us to assess the improvement from training on our synthetic data.

Guard models. We evaluate three existing guard models designed to identify unsafe content or actions [13]: Llama-Guard-3-8B [24], Qwen3Guard-Gen-8B [52], and AgentDoG1.5-Qwen3.5-4B [19]. We also include our proposed OBLIGATIONGUARD, which is trained to identify agents' unfulfilled obligations.

5.3 Metrics

To help agents complete tasks safely, a guard model must identify unfulfilled obligations accurately and completely, while avoiding false alarms when no obligations remain. We therefore report Precision, Recall, and Exact Match (EM) on positive instances, together with Classification Accuracy (CA) separately on positive and negative instances.

Since the same obligation can be expressed in different ways, we use GPT-5.6-Sol to judge whether a predicted obligation and a ground-truth obligation refer to the same required action given the task and trajectory [54]. We count the maximum number of such matches, with each prediction and each ground-truth obligation counted at most once to avoid double counting.

Formally, let $\mathcal{I}_+$ and $\mathcal{I}_-$ denote the index sets of positive and negative instances, respectively. For instance i , let $\hat{\mathcal{O}}_i$ and $\mathcal{O}_i^*$ denote the predicted and ground-truth obligation sets, respectively. The matching described above yields m_i matched pairs, which we use to calculate the following metrics.

❶ **Precision.** Precision measures the proportion of predicted obligations that are correct:

$$\text{Precision} = \frac{\sum_{i \in \mathcal{I}_+} m_i}{\sum_{i \in \mathcal{I}_+} |\hat{\mathcal{O}}_i|}. \quad (6)$$

❷ **Recall.** Recall measures the proportion of ground-truth obligations correctly identified by the model:

$$\text{Recall} = \frac{\sum_{i \in \mathcal{I}_+} m_i}{\sum_{i \in \mathcal{I}_+} |\mathcal{O}_i^*|}. \quad (7)$$

❸ **Exact Match (EM).** EM measures the proportion of positive instances for which the predicted obligation set is both accurate and complete, with no incorrect or missing obligations:

$$\text{EM} = \frac{1}{|\mathcal{I}_+|} \sum_{i \in \mathcal{I}_+} \mathbb{I}[m_i = |\hat{\mathcal{O}}_i| = |\mathcal{O}_i^*|]. \quad (8)$$

Here, $\mathbb{I}[\cdot]$ equals 1 when the condition holds and 0 otherwise.

❹ **Classification Accuracy (CA).** CA measures whether models correctly determine if any obligations remain unfulfilled. We report it separately for positive and negative instances: a correct prediction is a nonempty obligation set for a positive instance and an empty set for a negative instance.

$$\begin{aligned} \text{CA}_+ &= \frac{1}{|\mathcal{I}_+|} \sum_{i \in \mathcal{I}_+} \mathbb{I}[|\hat{\mathcal{O}}_i| > 0], \\ \text{CA}_- &= \frac{1}{|\mathcal{I}_-|} \sum_{i \in \mathcal{I}_-} \mathbb{I}[|\hat{\mathcal{O}}_i| = 0]. \end{aligned} \quad (9)$$

5.4 Implementation Details

During evaluation, we run each model once per instance using the same prompt and deterministic decoding whenever supported. For LLM judgement, we set the LLM judge’s temperature to 0. We train OBLIGATIONGUARD for two epochs using AdamW [21], with a learning rate of 1×10^{-5} and a batch size of 32. We set a token limit of 32,768 for both training sequences and evaluation contexts. All agent trajectories are collected by running agents with Mini-SWE-Agent [32]. Additional implementation details are provided in *Supplementary Materials* to facilitate reproducibility.

6 Experimental Results

6.1 RQ1: How effectively can existing models identify obligations?

Unfulfilled obligations pose substantial safety risks, yet identifying them requires more than judging whether an agent’s recorded actions are permitted. In this RQ, we evaluate whether existing models can identify accurate and complete obligation sets.

Setting. We evaluate the 14 existing models introduced in Section 5.2 on OBLIGATIONBENCH, using the metrics defined in Section 5.3. We report Precision, Recall, and EM on positive instances, together with CA_+ and CA_- on positive and negative instances, respectively. We also include OBLIGATIONGUARD for comparison.

Results. Table 2 reports the performance of all evaluated models.

Existing models struggle to identify accurate and complete obligation sets. Although several existing models achieve high Precision and classification accuracy, their Recall and EM remain low. For example, DeepSeek-V4.1-Flash achieves 90.08% Precision and 95.83% CA_- , but its Recall is only 32.15%. Similarly, Claude-Opus-4.8 achieves 93.33% CA_+ , yet its EM is only 9.17%. Across all 14 existing models, the highest Recall and EM are only 48.97% and 10.00%, respectively. The low EM is particularly concerning: even the best result corresponds to an accurate and complete obligation set on only one in ten positive instances. Thus, while these models can identify some required actions, their guidance may still leave other safety-critical actions unaddressed.

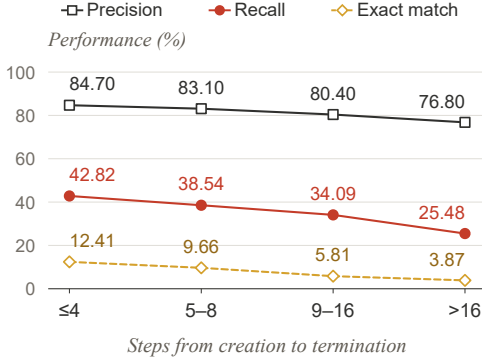
Existing guard models perform worse than most general-purpose LLMs. Despite being trained for safety, the three existing guard models achieve at most 5.60% Recall and 0.83% EM. They often fail to recognize that any obligations remain, with CA_+ no higher than 29.17%. For example, Llama-Guard-3-8B returns no obligations on any instance, obtaining 100.00% CA_- but 0.00% CA_+ . It therefore provides no guidance on the required actions even when the agent still has unfulfilled obligations. We attribute this gap in part to the focus of existing safety training on detecting forbidden actions. Such training may help models judge whether an action is permitted without teaching them to identify actions that are required but missing from the trajectory.

OBLIGATIONGUARD substantially improves obligation identification. By training on our synthetic trajectories, OBLIGATIONGUARD improves Recall from 15.34% to 57.52% and EM from 0.83% to 21.67% over its Qwen3-8B backbone. Precision also increases from 51.49% to 82.98%, indicating that the model identifies more obligations while making more accurate predictions. These improvements allow OBLIGATIONGUARD to outperform all evaluated models in both Recall and EM: its Recall exceeds the best existing result by 8.55 percentage points, and its EM is more than twice the best existing result of 10.00%. The gains over its backbone show that our synthetic trajectories provide useful supervision for identifying obligations in real agent executions.

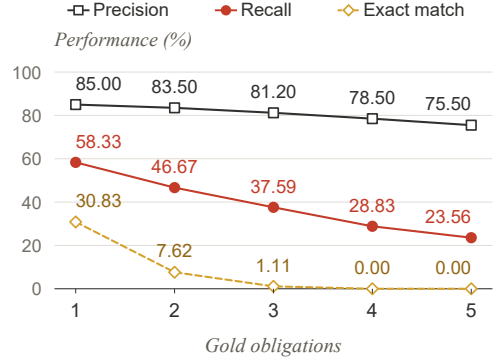
Answer to RQ1. Existing models rarely identify accurate and complete obligation sets, and existing guard models return no obligations on most positive instances. OBLIGATIONGUARD achieves the highest Recall and EM, more than doubling the best existing EM.

Table 2. Obligation identification on OBLIGATIONBENCH (%). The best and second-best results in each column are bold and underlined.

Model	Positive cases				Negative cases
	Precision	Recall	EM	CA ₊	CA ₋
<i>General-purpose LLMs</i>					
Claude-Opus-4.8	73.45	<u>48.97</u>	9.17	93.33	79.17
GPT-5.6-Sol	<u>86.63</u>	47.79	9.17	<u>90.83</u>	80.00
DeepSeek-V4-Pro	78.08	33.63	5.83	65.83	90.83
DeepSeek-V4.1-Flash	90.08	32.15	<u>10.00</u>	63.33	<u>95.83</u>
Kimi-K3	81.40	30.97	7.50	54.17	92.50
GLM-5.3	83.06	30.38	6.67	55.83	91.67
Qwen3.8-27B	71.00	20.94	1.67	65.00	84.17
Qwen3.8-Max	85.92	17.99	0.83	41.67	94.17
Qwen3-8B	51.49	15.34	0.83	65.00	73.33
GPT-OSS-20B	58.93	9.73	1.67	37.50	80.83
Llama-3.1-8B-Instruct	3.03	0.59	0.00	34.17	94.17
<i>Existing guard models</i>					
Qwen3Guard-Gen-8B	40.43	5.60	0.83	29.17	83.33
AgentDoG1.5-Qwen3.5-4B	17.65	1.77	0.00	26.67	82.50
Llama-Guard-3-8B	0.00	0.00	0.00	0.00	100.00
ObligationGuard (ours)	82.98	57.52	21.67	85.83	89.17



(a) Creation-to-termination distance



(b) Concurrent obligations

Fig. 6. Precision, Recall, and EM by (a) the number of actions between the earliest obligation’s creation and trajectory termination and (b) the number of coexisting obligations. Results are averaged over five models.

6.2 RQ2: What factors affect models’ performance in obligation identification?

We are interested in understanding what affects models’ performance in obligation identification. We consider two factors: the number of actions between obligation creation and trajectory termination, and the number of obligations that coexist.

Setting. We evaluate Claude-Opus-4.8, DeepSeek-V4.1-Flash, DeepSeek-V4-Pro, Kimi-K3, and GLM-5.3 on the 120 positive instances in OBLIGATIONBENCH and report the average results across these five models. Specifically, we examine how their performance varies with the following factors.

❶ **Creation-to-termination distance.** We measure this distance as the number of actions between the creation of the obligation and trajectory termination. We use GPT-5.6-Sol to annotate each obligation’s creation step and have human experts independently review the annotations. For trajectory i , the distance is $D_i = T_i - \min_{q \in O_i^*} c_i(q)$, where T_i is the final step and $c_i(q)$ is the

creation step of obligation q . We then group instances into four distance ranges (≤ 4 , 5–8, 9–16, and > 16) and report Precision, Recall, and EM for each group.

② **Concurrent obligations.** We group instances by the number of ground-truth obligations, ranging from one to five, and report the same three metrics for each group.

Results. Figure 6 shows how performance varies with the two factors.

Models perform worse when more actions follow obligation creation. As the number of actions between the earliest obligation’s creation and termination increases from at most four to more than sixteen, Recall falls from 42.82% to 25.48%, and EM from 12.41% to 3.87%. Precision also decreases from 84.70% to 76.80%, but remains relatively high. The widening gap between Precision and Recall suggests that their predictions become increasingly incomplete as more actions intervene before termination, even though individual predictions are often correct.

Obligation identification becomes harder as more obligations coexist. As the number of ground-truth obligations increases from one to five, Recall decreases from 58.33% to 23.56%, while EM falls from 30.83% to 0.00%. When five obligations coexist, models identify fewer than one quarter of the required actions on average, and none returns an accurate and complete obligation set. The decline in Recall shows that models miss a growing proportion of obligations as more coexist. These omissions make complete identification increasingly difficult: a guard model may identify some required actions while leaving other safety-critical actions unaddressed.

Answer to RQ2. Models identify obligations less completely when more actions follow obligation creation and when more obligations coexist.

6.3 RQ3: What contributes to the effectiveness of OBLIGATIONGUARD?

The effectiveness of OBLIGATIONGUARD may depend on its backbone, the amount of training data, and how these examples are generated. We examine how performance varies across model sizes and families, and whether using synthesis templates produces more effective training data.

Setting. We conduct the following comparisons on OBLIGATIONBENCH, using the metrics defined in Section 5.3 and the training method described in Section 4.2.

① **Model size.** To examine how the benefits vary with model size, we fine-tune Qwen3-4B, Qwen3-8B, and Qwen3-14B on the same 40,000 synthetic examples and compare each model with its original backbone.

② **Model family.** To examine whether our synthetic data also benefits another model family, we fine-tune Llama-3.1-8B-Instruct using the same data and training settings as Qwen3-8B, and compare it with its original backbone.

③ **Training-data size.** We train Qwen3-8B on 5K, 10K, 20K, and 40K synthetic examples to examine how the amount of training data affects obligation identification.

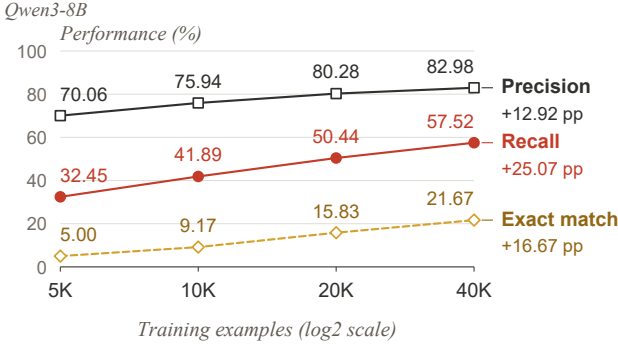
④ **Synthesis templates.** Our approach first constructs synthesis templates that specify tasks, safety requirements, and the obligations to remain unfulfilled, then generates trajectories from these templates. To examine whether the templates are useful, we ask GPT-5.6-Sol to directly generate tasks, trajectories, and their obligation annotations without using templates. We refer to this setting as *direct synthesis* and compare Qwen3-8B trained on these examples with OBLIGATIONGUARD. Both settings use GPT-5.6-Sol as the generator, 40K training examples, and the same training settings.

Results. Table 3 reports the results across model sizes and families. Figures 7 and 8 show the effects of training-data size and the use of synthesis templates.

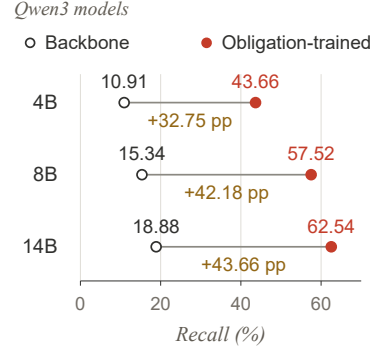
Our synthetic data improves obligation identification at all three model sizes. As shown in Table 3, fine-tuning improves Precision, Recall, and EM for all three Qwen3 models. For example,

Table 3. Obligation identification before and after fine-tuning on our 40,000 synthetic examples (%).

Setting	Positive cases				Negative cases
	Precision	Recall	EM	CA+	CA-
Qwen3-4B					
Original	44.05	10.91	0.00	48.33	69.17
Fine-tuned	77.49	43.66	12.50	83.33	84.17
Qwen3-8B					
Original	51.49	15.34	0.83	65.00	73.33
Fine-tuned (ours)	82.98	57.52	21.67	85.83	89.17
Qwen3-14B					
Original	63.37	18.88	1.67	69.17	75.83
Fine-tuned	84.80	62.54	25.83	91.67	90.83
Llama-3.1-8B-Instruct					
Original	3.03	0.59	0.00	34.17	94.17
Fine-tuned	78.24	49.85	16.67	85.83	87.50



(a) Training-data size



(b) Model size

Fig. 7. Effects of training-data size and model size on obligation identification. (a) Precision, Recall, and EM of Qwen3-8B fine-tuned on 5K–40K synthetic examples. (b) Recall of Qwen3 models of different sizes before and after fine-tuning on 40K synthetic examples.

Qwen3-4B’s Recall increases from 10.91% to 43.66%, and its EM from 0.00% to 12.50%. Qwen3-14B also benefits substantially, with Recall rising from 18.88% to 62.54% and EM from 1.67% to 25.83%. Larger fine-tuned models achieve better results, but even the fine-tuned 4B model exceeds the original 14B model in all three metrics. Within this range, training on trajectories with obligation annotations provides larger gains than increasing the size of the original model.

The same synthetic data also benefits a different model family. Fine-tuning Llama-3.1-8B-Instruct increases Precision from 3.03% to 78.24%, Recall from 0.59% to 49.85%, and EM from 0.00% to 16.67%. Despite its poor initial performance, the fine-tuned model exceeds the best existing Recall and EM reported in RQ1. The usefulness of our synthetic data therefore extends beyond the Qwen3 family used to develop OBLIGATIONGUARD.

More synthetic examples improve both accuracy and completeness. Figure 7a shows consistent improvements as the training set grows. Increasing the number of examples from 5K to 40K raises Precision from 70.06% to 82.98%, Recall from 32.45% to 57.52%, and EM from 5.00% to 21.67%. All three metrics improve at each evaluated data size, indicating that additional examples help the model identify more obligations accurately. These results support synthesizing training data at scale: within the evaluated range, additional synthetic trajectories continue to improve performance on real agent executions.

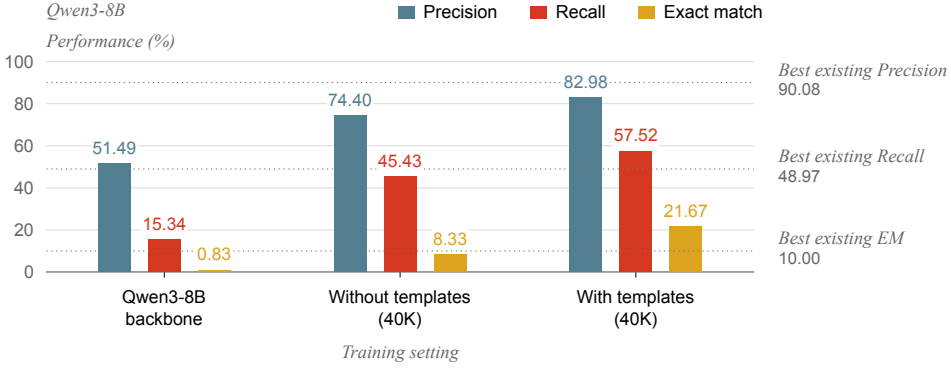


Fig. 8. Effect of synthesis templates on Qwen3-8B. Both synthesis methods provide 40K training examples. Dashed lines mark the best existing results from Table 2.

Using synthesis templates makes the training data more effective. As shown in Figure 8, training on directly generated examples improves Qwen3-8B over its original backbone, achieving 74.40% Precision, 45.43% Recall, and 8.33% EM. Using synthesis templates further raises these metrics to 82.98%, 57.52%, and 21.67%, respectively. With the same generator, training-data size, and fine-tuning settings, EM more than doubles. We attribute these gains in part to making the safety requirements and obligation sets explicit before trajectory generation. The templates guide the generator in constructing trajectories consistent with these requirements and provide the corresponding obligation labels, helping the guard model correctly learn which safety-critical actions remain unfulfilled.

Answer to RQ3. Fine-tuning on our synthetic training data improves performance across the evaluated model sizes and families. Increasing the number of training examples further improves all three metrics. With the same number of training examples, using scenario planning during data synthesis leads to better performance than direct synthesis.

6.4 RQ4: Does stronger obligation identification improve agent safety?

The practical value of obligation identification depends on whether it can help agents complete the required but unperformed safety-critical actions. We therefore examine whether guidance from guard models improves coding-agent safety and whether guard models with better performance on OBLIGATIONBENCH provide more effective guidance.

Setting. We evaluate Qwen3.8-27B with Mini-SWE-Agent on 186 tasks from SusVibes [53]. At the agent’s first termination attempt, we compare three conditions: ❶ **No guidance.** The agent terminates without receiving additional guidance. ❷ **Self-reminder.** We ask the agent to recheck potential safety issues [41]. ❸ **Guard model guidance.** We provide the task and trajectory to a guard model and return its predicted obligations to the agent. All three conditions start from the same execution state. In the two guidance conditions, the agent can continue execution to revise its solution, but receives guidance at most once per execution.

We compare the six guard models listed in Table 4. We report FuncPass, the percentage of tasks that pass the functional tests, and SecPass, the percentage that pass both the functional and safety tests. To examine how guidance changes individual outcomes, we classify each solution as *Incorrect* if it fails the functional tests, *Correct-Unsafe* if it passes only the functional tests, or *Correct-Safe* if it passes both. We also examine the association between each guard model’s Recall on OBLIGATIONBENCH and its improvement in SecPass.

Table 4. Effects of guidance on the Qwen3.8-27B agent running with Mini-SWE-Agent on SusVibes. Δ Sec denotes the improvement over no guidance. The best and second-best results are bold and underlined.

Guidance source	FuncPass (%)	SecPass (%)	Δ Sec (%)
No guidance	27.4	6.5	–
Self-reminder	<u>26.9</u>	8.1	+1.6
Qwen3-8B	25.8	7.5	+1.0
GLM-5.3	26.3	9.7	+3.2
Kimi-K3	26.3	9.1	+2.6
DeepSeek-V4.1-Flash	<u>26.9</u>	10.8	+4.3
Claude-Opus-4.8	25.8	12.4	+5.9
ObligationGuard (ours)	<u>26.9</u>	15.1	+8.6

Table 5. Functional and safety test results before (rows) and after (columns) OBLIGATIONGUARD guidance. Values are task counts. The highlighted cell shows 16 solutions that passed only the functional tests before guidance and passed both tests afterward.

Before guidance		After guidance		
Outcome	Tasks	Incorrect	Correct-Unsafe	Correct-Safe
Incorrect	135	133	1	1
Correct-Unsafe	39	3	20	16
Correct-Safe	12	0	1	11

Results. Table 4 reports agent performance under each guidance condition. Table 5 further shows how OBLIGATIONGUARD helps the agent address safety failures.

OBLIGATIONGUARD improves agent safety while largely preserving functional correctness. Without guidance, the agent achieves 6.5% SecPass. A self-reminder raises this to 8.1%, while guidance from OBLIGATIONGUARD increases it to 15.1%, exceeding all other guidance conditions. The strongest existing guard model in this experiment, Claude-Opus-4.8, achieves 12.4% SecPass. Meanwhile, FuncPass changes only from 27.4% without guidance to 26.9% with OBLIGATIONGUARD. These results demonstrate the effectiveness of OBLIGATIONGUARD in real agent executions: it more than doubles SecPass and achieves the largest safety gain among all evaluated guard models, while largely preserving functional completion.

Most safety gains come from correcting unsafe solutions that already pass functional tests. Table 5 shows that 16 of the 39 initially Correct-Unsafe solutions become Correct-Safe after OBLIGATIONGUARD guidance. Only one initially Incorrect solution becomes Correct-Safe. Of the 17 newly safe solutions, 16 therefore already passed the functional tests before guidance. This result helps explain why SecPass improves substantially while FuncPass remains nearly unchanged: the guard model primarily helps the agent address safety failures in otherwise working solutions.

Identifying the unfulfilled obligation helps the agent address the specific safety failure. Figure 9 illustrates how the guidance leads to a safer solution. In this persistent-login task, the agent attempts to terminate at step 24, although server-side session expiration is still missing. An old session may therefore remain usable after its intended expiration time. Qwen3-8B suggests clearing the persistent cookie on logout, which leaves expiration enforcement unaddressed. OBLIGATIONGUARD instead identifies the obligation to reject expired sessions. Following this guidance, the agent adds the server-side expiration check and requires users with expired sessions to log in again. Its solution becomes Correct-Safe while retaining its functional behavior. This case illustrates how identifying the unfulfilled obligation guides the agent toward the required safety-critical action.

Guard models with better performance on OBLIGATIONBENCH tend to improve agent safety more. Across the six evaluated guard models, Recall on OBLIGATIONBENCH is strongly associated with the improvement in SecPass on SusVibes (Spearman’s $\rho = 0.94$; Pearson’s $r = 0.97$).

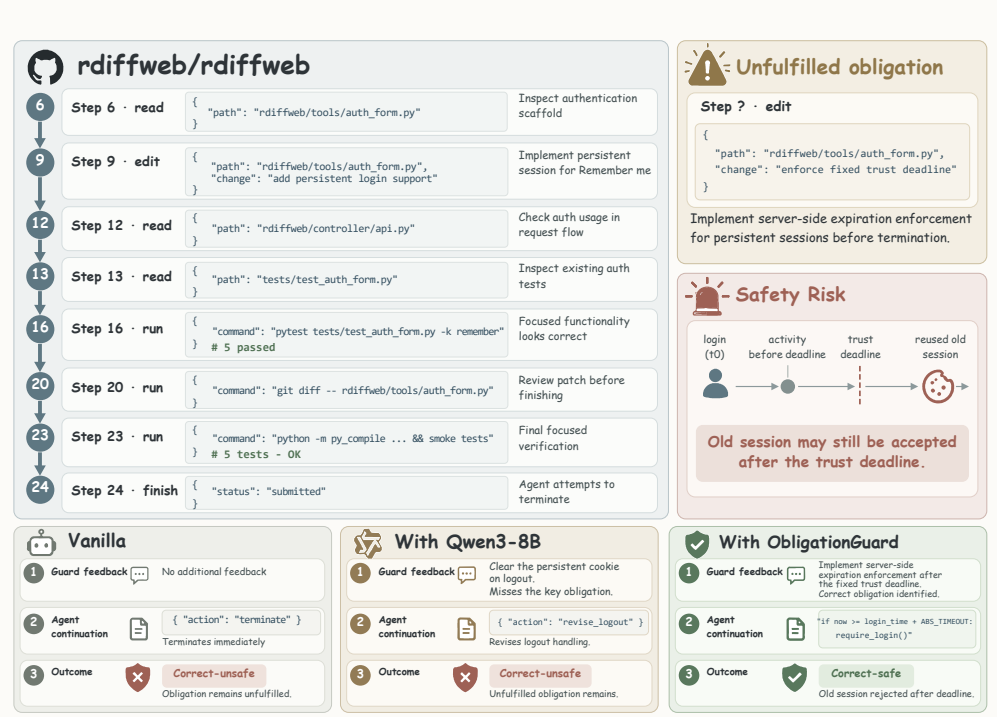


Fig. 9. On a SusVibes task, OBLIGATIONGUARD guides the agent to enforce server-side session expiration, while Qwen3-8B misses this obligation.

For example, Qwen3-8B achieves 15.34% Recall and increases SecPass by only 1.0 percentage points. Claude-Opus-4.8 and OBLIGATIONGUARD, with Recall of 48.97% and 57.52%, increase SecPass by 5.9 and 8.6 percentage points, respectively. This association links performance on OBLIGATIONBENCH to the usefulness of guidance from guard models during agent execution. It suggests that guard models that identify more unfulfilled obligations can give agents more complete guidance on the safety-critical actions still required.

Answer to RQ4. OBLIGATIONGUARD increases SecPass from 6.5% to 15.1% with little change in FuncPass. Better performance on OBLIGATIONBENCH is associated with larger safety gains on SusVibes.

7 Discussion

7.1 Reliability of LLM Judgment

We use GPT-5.6-Sol to assess semantic equivalence between predicted and ground-truth obligations. To examine the reliability of these judgments, we manually review 400 obligation pairs from the RQ1 results, stratified by evaluated model: 200 judged as matched and 200 as unmatched. Reviewers assess each pair using the corresponding task and trajectory, without seeing the model’s identity or the LLM judge’s decision. Human review confirms 196 of the 200 matched judgments (98.0%) and 184 of the 200 unmatched judgments (92.0%), giving 95.0% overall agreement on this balanced sample. These results indicate that the LLM judge provides reliable semantic matching judgments.

7.2 Threats to Validity

We consider four threats to the validity of our study:

Reliability of obligation annotations. Annotating obligations requires determining which safety-critical actions are necessary and whether the agent has completed them. Automatic annotation may overlook a required action or mistake an unnecessary action for an obligation, affecting model evaluation. To mitigate this threat, two software engineers with over five years of professional experience independently review and correct the initial annotations. They check whether the annotated actions remain required at termination and identify any missing obligations, then resolve disagreements through discussion and discard ambiguous cases.

Diversity and realism of the benchmark. Synthetic trajectories may not capture the variety of tasks and execution behavior that agents encounter in practice, limiting the benchmark’s diversity and realism. We therefore collect trajectories from actual executions on established software-engineering benchmarks, covering issue resolution, feature development, and terminal operation. Agents powered by four different LLMs provide varied trajectories within these domains. This coverage also extends to the safety requirements: the benchmark includes five safety categories and one to five unfulfilled obligations per positive instance.

Practical relevance of the benchmark. High scores on OBLIGATIONBENCH alone do not establish whether a guard model improves agent safety in practice. We therefore evaluate six guard models on SusVibes by using their predicted obligations to guide an agent. At its first termination attempt, the agent receives the identified obligations and can continue execution to fulfill them. We find that higher Recall on OBLIGATIONBENCH is associated with larger SecPass gains, supporting the benchmark’s relevance to agent safety during actual execution.

Generalizability of the findings. The choice of evaluated models may limit how broadly our findings apply. Our evaluation includes 14 existing models spanning different families and sizes, covering both general-purpose LLMs and guard models. We also examine whether the improvements from our synthetic training data extend beyond a single backbone by fine-tuning Qwen3 models at three sizes and a Llama backbone on the same data.

8 Conclusion

In this paper, we show that identifying forbidden actions alone is insufficient to ensure agent safety, as unfulfilled obligations constitute a major source of safety risk. We introduce OBLIGATIONBENCH, the first benchmark for obligation identification, and find that 14 existing models struggle to identify accurate and complete obligation sets. To address this limitation, we develop OBLIGATIONGUARD using synthetic training data generated through scenario planning and trajectory synthesis. Our experiments demonstrate that OBLIGATIONGUARD improves obligation identification and helps agents complete tasks more safely while largely preserving functional completion. We hope this work encourages the community to incorporate obligation identification into the design and evaluation of future guard models.

9 Data Availability

Our code, benchmarks, and *Supplementary Materials* can be accessed through a public repository at: <https://github.com/THU-Agent/ObligationGuard>.

References

- [1] Alibaba Cloud. 2026. qwen3.8-max Model Info. Accessed October 2, 2026. <https://www.alibabacloud.com/help/en/model-studio/qwen3-8-max>
- [2] Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, et al. 2025. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations*, Vol. 2025. OpenReview.net, Singapore, 79185–79220.

- [3] Anthropic. 2026. Introducing Claude Opus 4.8. Accessed October 2, 2026. <https://www.anthropic.com/news/claude-opus-4-8>
- [4] Haorui Chen, Yuancheng Zhu, Yitong Zhang, and Jia Li. 2026. CoACT: Action-Preserving Observation Compression for Coding Agents. arXiv preprint arXiv:2607.02911. doi:10.48550/arXiv.2607.02911
- [5] Yen-Shan Chen, Sian-Yao Huang, Cheng-Lin Yang, and Yun-Nung Chen. 2026. TraceSafe: A Systematic Assessment of LLM Guardrails on Multi-Step Tool-Calling Trajectories. In *Third Conference on Language Modeling*. <https://openreview.net/forum?id=KkFKVLWdEG>
- [6] Zhaorun Chen, Mintong Kang, and Bo Li. 2025. ShieldAgent: Shielding agents via verifiable safety policy reasoning. arXiv preprint arXiv:2503.22738.
- [7] Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, Shengye Wan, Spencer Whitman, Lauren Deason, Nicholas Doucette, Abraham Montilla, et al. 2025. LlamaFirewall: An open source guardrail system for building secure AI agents. arXiv preprint arXiv:2505.03574.
- [8] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. *Advances in neural information processing systems* 37 (2024), 82895–82920.
- [9] DeepSeek-AI. 2026. DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression. <https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash>
- [10] DeepSeek-AI et al. 2026. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence. arXiv preprint arXiv:2606.19348. <https://arxiv.org/abs/2606.19348>
- [11] Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. 2025. SWE-Bench Pro: Can AI agents solve long-horizon software engineering tasks? arXiv preprint arXiv:2509.16941.
- [12] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. 2024. The Llama 3 Herd of Models. arXiv preprint arXiv:2407.21783. <https://arxiv.org/abs/2407.21783>
- [13] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. 2023. Llama Guard: LLM-based input-output safeguard for human-AI conversations. arXiv preprint arXiv:2312.06674.
- [14] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-world GitHub Issues?. In *International Conference on Learning Representations*, Vol. 2024. 54107–54157. https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html
- [15] Kimi Team et al. 2026. Kimi K3: Open Frontier Intelligence. arXiv preprint arXiv:2607.24653. <https://arxiv.org/abs/2607.24653>
- [16] Yu Li, Haoyu Luo, Yuejin Xie, Yuqian Fu, Zhonghao Yang, Shuai Shao, Qihan Ren, Wanying Qu, Yanwei Fu, Yujiu Yang, Jing Shao, Xia Hu, and Dongrui Liu. 2026. ATBench: A Diverse and Realistic Agent Trajectory Benchmark for Safety Evaluation and Diagnosis. arXiv preprint arXiv:2604.02022. <https://arxiv.org/abs/2604.02022>
- [17] Shuquan Lian, Juncheng Liu, Yazhe Chen, Yuhong Chen, and Hui Li. 2026. SWE-AGILE: A Software Agent Framework for Efficiently Managing Dynamic Reasoning Context. arXiv preprint arXiv:2604.11716. <https://arxiv.org/abs/2604.11716>
- [18] Wenhao Lin, Chenyu Yu, Xingwei Lin, Sicong Cao, Xiang Chen, Lei Xue, Le Yu, Letian Sha, and Chunming Wu. 2026. DreamGuard: Efficient Runtime Guardrail for LLM Agents via Risk-Aware World Model. *arXiv preprint arXiv:2608.05695* (2026).
- [19] Dongrui Liu, Yu Li, Zhonghao Yang, Peng Wang, Guanxu Chen, Yuejin Xie, Qinghua Mao, Wanying Qu, Yanxu Zhu, Tianyi Zhou, et al. 2026. AgentDoG 1.5: A Lightweight and Scalable Alignment Framework for AI Agent Safety and Security. arXiv preprint arXiv:2605.29801. <https://arxiv.org/abs/2605.29801>
- [20] Xiting Liu, Yuetong Liu, Yitong Zhang, Jia Li, and Shi-Min Hu. 2026. PackMonitor: Enabling Zero Package Hallucinations Through Decoding-Time Monitoring. arXiv preprint arXiv:2602.20717. doi:10.48550/arXiv.2602.20717
- [21] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*. <https://arxiv.org/abs/1711.05101>
- [22] Hanjun Luo, Shenyu Dai, Chiming Ni, Xinfeng Li, Guibin Zhang, Kun Wang, Tongliang Liu, and Hanan Salam. 2025. AgentAuditor: Human-level safety and security evaluation for LLM agents. *Advances in Neural Information Processing Systems* 38 (2025), 43241–43298.
- [23] Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. 2026. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, Vol. 2026. OpenReview.net, Rio de Janeiro, Brazil, 40903–40986. https://proceedings.iclr.cc/paper_files/paper/2026/hash/444a3737adaee10d86ad2ef5f74468e6-Abstract-Conference.html

- [24] Meta. 2024. Llama Guard 3-8B Model Card. Accessed October 2, 2026. <https://huggingface.co/meta-llama/Llama-Guard-3-8B>
- [25] Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. 2026. ToolSafe: Enhancing tool invocation safety of LLM-based agents via proactive step-level guardrail and feedback. In *Findings of the Association for Computational Linguistics: ACL 2026*. Association for Computational Linguistics, San Diego, California, USA, 37125–37153.
- [26] OpenAI. 2025. gpt-oss-120b & gpt-oss-20b Model Card. arXiv preprint arXiv:2508.10925. <https://arxiv.org/abs/2508.10925>
- [27] OpenAI. 2026. GPT-5.6 Sol Model. Accessed October 2, 2026. <https://developers.openai.com/api/docs/models/gpt-5.6-sol>
- [28] OWASP Foundation. [n. d.]. Session Management Cheat Sheet. OWASP Cheat Sheet Series. Accessed October 2, 2026. https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html
- [29] Qwen Team. 2026. Qwen3.8-27B Model Card. Accessed October 2, 2026. <https://huggingface.co/Qwen/Qwen3.8-27B>
- [30] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. 2024. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations*, Vol. 2024. 27031–27098.
- [31] Aditya Bharat Soni, Boxuan Li, Xingyao Wang, Valerie Chen, and Graham Neubig. 2026. Coding Agents with Multimodal Browsing are Generalist Problem Solvers. In *Findings of the Association for Computational Linguistics: EACL 2026*. Association for Computational Linguistics, Rabat, Morocco, 6052–6069. doi:10.18653/v1/2026.findings-eacl.318
- [32] SWE-agent Team. [n. d.]. mini-SWE-agent. Software repository. Accessed October 2, 2026. <https://github.com/SWE-agent/mini-swe-agent>
- [33] Haoyu Wang, Christopher M. Poskitt, and Jun Sun. 2026. AgentSpec: Customizable Runtime Enforcement for Safe and Reliable LLM Agents. In *Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE '26)*. Association for Computing Machinery, 2938–2950. doi:10.1145/3744916.3764546
- [34] Haoyu Wang, Christopher M. Poskitt, Jiali Wei, and Jun Sun. 2026. ProbGuard: Proactive Runtime Monitoring for LLM Agent Safety via Probabilistic Prediction. In *Proceedings of the 41st IEEE/ACM International Conference on Automated Software Engineering*. Accepted for publication. <https://arxiv.org/abs/2508.00500v4>
- [35] Haoyu Wang, Zibo Xiao, Yedi Zhang, Christopher M. Poskitt, and Jun Sun. 2026. SafeClaw-R: Towards Safe and Secure Multi-Agent Personal Assistants. arXiv preprint arXiv:2603.28807. doi:10.48550/arXiv.2603.28807
- [36] Haoyu Wang, Wei Zhao, Yedi Zhang, Christopher M. Poskitt, and Jun Sun. 2026. Representation Transitions Reveal Emerging Safety Risks in Multi-Turn LLM Agents. arXiv preprint arXiv:2610.00400. <https://arxiv.org/abs/2610.00400>
- [37] Jiacheng Wang, Jinchang Hou, Fabian Wang, Ping Jian, Chenfu Bao, and Zhonghou Lv. 2026. HINTBench: Horizon-Agent Intrinsic Non-Attack Trajectory Benchmark. arXiv preprint arXiv:2604.13954. <https://arxiv.org/abs/2604.13954>
- [38] Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. 2025. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, Vol. 2025. OpenReview.net, Singapore, 65882–65919.
- [39] Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. 2025. GuardAgent: Safeguard LLM Agents via Knowledge-Enabled Reasoning. In *Proceedings of the 42nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 267)*. PMLR, 68316–68342. <https://proceedings.mlr.press/v267/xiang25a.html>
- [40] Zibo Xiao, Haoyu Wang, and Jun Sun. 2026. S³: Improving Agent Safety through Multi-Stage Defense. arXiv preprint arXiv:2608.02683. doi:10.48550/arXiv.2608.02683
- [41] Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. 2023. Defending ChatGPT against jailbreak attack via self-reminders. *Nature Machine Intelligence* 5, 12 (2023), 1486–1496. doi:10.1038/s42256-023-00765-8
- [42] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. 2025. Qwen3 Technical Report. arXiv preprint arXiv:2505.09388. <https://arxiv.org/abs/2505.09388>
- [43] John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems* 37 (2024), 50528–50652.
- [44] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629.
- [45] Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, et al. 2024. R-Judge: Benchmarking safety risk awareness for LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, Miami, Florida, USA, 1467–1490.

- [46] Z.ai. 2026. GLM-5.3: Frontier Coding with Emergent Cyber Capabilities. Accessed October 2, 2026. <https://z.ai/blog/glm-5.3>
- [47] Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. 2025. Agent Security Bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. In *International Conference on Learning Representations*, Vol. 2025. OpenReview.net, Singapore, 35331–35366.
- [48] Jiacheng Zhang, Haoyu He, Sen Zhang, Shen Wang, Xiaolei Xu, Yuhao Sun, Meng Shen, and Feng Liu. 2026. SafePyramid: A Hierarchical Benchmark for In-context Policy Guardrailing. arXiv preprint arXiv:2606.29887. <https://arxiv.org/abs/2606.29887>
- [49] Yitong Zhang, Ximo Li, Liyi Cai, and Jia Li. 2026. Environmental Injection Attacks against GUI Agents in Realistic Dynamic Environments. *Proceedings of the ACM on Software Engineering* 3, ISSTA, Article ISSTA057 (Oct. 2026), 21 pages. doi:10.1145/3832148
- [50] Yedi Zhang, Haoyu Wang, Xianglin Yang, Jin Song Dong, and Jun Sun. 2026. LLM-enabled Applications Require System-Level Threat Monitoring. arXiv preprint arXiv:2602.19844. doi:10.48550/arXiv.2602.19844
- [51] Zhixin Zhang, Shiyao Cui, Yida Lu, Jingzhuo Zhou, Junxiao Yang, Hongning Wang, and Minlie Huang. 2024. Agent-SafetyBench: Evaluating the safety of LLM agents. arXiv preprint arXiv:2412.14470.
- [52] Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, et al. 2025. Qwen3Guard technical report. arXiv preprint arXiv:2510.14276.
- [53] Songwen Zhao, Danqing Wang, Kexun Zhang, Jiaxuan Luo, Zhuo Li, and Lei Li. 2026. Is Vibe Coding Safe? Benchmarking Vulnerability of Agent-Generated Code in Real-World Tasks. In *Proceedings of the 43rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 306)*. PMLR, 162269–162292. <https://proceedings.mlr.press/v306/zhao26ax.html>
- [54] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, Vol. 36. <https://arxiv.org/abs/2306.05685>
- [55] Qixing Zhou, Jiacheng Zhang, Haiyang Wang, Rui Hao, Jiahe Wang, Minghao Han, Yuxue Yang, Shuzhe Wu, Feiyang Pan, Lue Fan, Dandan Tu, and Zhaoxiang Zhang. 2026. FeatureBench: Benchmarking Agentic Coding for Complex Feature Development. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=41xrZ3uGuI>
- [56] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2024. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, Vol. 2024. OpenReview.net, Vienna, Austria, 15585–15606.