Peering Behind the Shield: Guardrail Identification in Large Language Models

Ziqing Yang[1], **Yixin Wu**[1], **Rui Wen**[2], **Michael Backes**[1], **Yang Zhang**[1]

[1]CISPA Helmholtz Center for Information Security
[2]Institute of Science Tokyo
{ziqing.yang, yixin.wu}@cispa.de, rui.w.b9ce@m.isct.ac.jp, {director, zhang}@cispa.de

Abstract

With the rapid adoption of large language models (LLMs), conversational AI agents have become widely deployed across real-world applications. To enhance safety, these agents are often equipped with guardrails that moderate harmful content. Identifying the guardrails in an agent thus becomes critical for adversaries to understand the system and design guard-specific attacks. In this work, we introduce **AP-Test**, a novel approach that leverages guard-specific adversarial prompts to detect the identity of guardrails deployed in black-box AI agents. Our method addresses key challenges in this task, including the influence of safety-aligned LLMs and other guardrails, as well as a lack of principled decision-making strategies. **AP-Test** employs two complementary testing strategies, input and output guard tests, and a new metric, match score, to enable robust identification. Experiments across diverse agents and four open-source guardrails demonstrate that **AP-Test** achieves perfect classification accuracy in multiple scenarios. Ablation studies further highlight the necessity of our proposed components. Our findings reveal a practical path toward guardrail identification in real-world AI systems.

1 Introduction

Human-AI conversations have been significantly advanced by the rapid development of large language models (LLMs). These conversational AI agents are now extensively deployed across various domains, including customer service (DeepSeek-AI et al. 2025; Anthropic 2024; OpenAI 2024), education (Neumann et al. 2024), and healthcare (Peng et al. 2023). Such widespread use also raises severe security concerns, such as jailbreak attacks (Shen et al. 2024; Zou et al. 2023; Liu et al. 2023; Zhao et al. 2024) and prompt injection attacks (Liu et al. 2024a; Zhan et al. 2024; Debenedetti et al. 2024).

Safety guardrails (Inan et al. 2023; Zeng et al. 2024) are thus developed to further moderate the input/output content of the AI agents, as shown in Figure 1. They are often fine-tuned on top of LLMs using annotated datasets that cover a broad spectrum of safety risks (Inan et al. 2023; Han et al. 2024; Zeng et al. 2024). In addition, a growing number of high-quality open-source guardrails, such as Llama-Guard (Inan et al. 2023), WildGuard (Han et al. 2024), and ShieldGemma (Zeng et al. 2024), have become widely available and are increasingly integrated into AI Agents. This largely challenges attackers to bypass the safety mechanisms in AI agents; thus, it is crucial to identify the guardrails deployed in the AI agent. Once a guardrail is identified, it can be treated as a white-box component, facilitating guard-specific attacks that are shown to outperform black-box methods (Mangaokar et al. 2024; Zhang, Xiong, and Mao 2024).

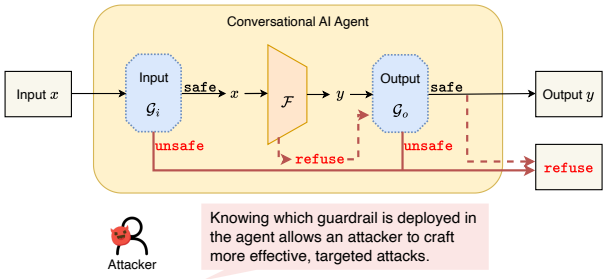


Figure 1: A conversational AI agent equipped with guardrails. Normally, the agent responds to the input following the black line. Once the guardrail flags the input or generated response `unsafe`, the agent would `refuse` the query (solid red line). The dashed red line signifies the LLM's refusal to the input per its internal safety alignment.

Considering that guardrails are often built on top of LLMs, a natural question arises: can we simply apply LLM identification methods (Zeng et al. 2023; Gubri et al. 2024; Jin et al. 2024; Yang et al. 2025) to detect the presence of specific guardrails in an AI agent? LLM identification aims to determine the origin or identity of an LLM, primarily to address concerns about unauthorized use. The common approach relies on optimizing adversarial prompts that elicit distinctive responses from the candidate LLM (Gubri et al. 2024; Jin et al. 2024). However, this line of work falls short due to several challenges in guardrail identification: **C1** The output space of guardrails is extremely limited (e.g., binary safe/unsafe responses), so optimizing input triggers toward a target response is much easier. This makes it hard to be specific to one guardrail. **C2** AI agents usually allow query-only access and do not provide knowledge of the under-

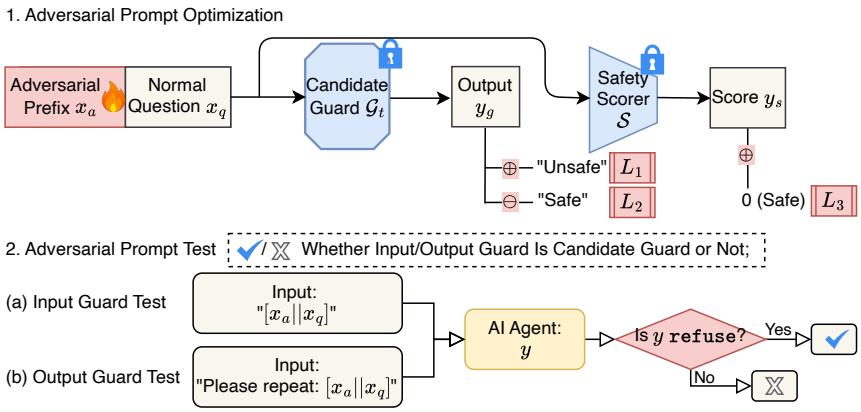


Figure 2: Framework of our **AP-Test**. We first perform *adversarial prompt optimization* based on the candidate guardrail with our tailored loss function. Then, we conduct *adversarial prompt tests* by querying the AI agent with the adversarial prompts to determine whether the candidate guardrail exists in the agent.

lying safety mechanisms (OpenAI 2024; Anthropic 2024; Github 2025; DeepSeek-AI et al. 2025), revealing little information about the guardrail. Mechanisms such as additional guardrails or safety-aligned LLMs may also obscure the existence of the candidate guardrail. **C3** Guardrails are integrated into AI agents with unknown deployment stages; when applied at the output stage, their input depends on the LLM's outputs. In this sense, it is challenging to input the adversarial prompts into the output guardrail. **C4** Existing LLM identification methods lack principled strategies for selecting identification thresholds, limiting their robustness and reliability in practical settings.

By solving the above challenges, we propose **AP-Test**, which utilizes **A**dversarial **P**rompts to **Test** the identity of the safety guardrail deployed in a black-box AI agent. As shown in Figure 2, our approach begins by probing the AI agent with guard-specific adversarial prompts, which are designed to be flagged as `unsafe` by a specific candidate guardrail while remaining `safe` according to others. Such optimization exploits the limited output space of guardrails (**C1**) and mitigates the influence of safety-aligned LLMs and other safety guardrails (**C2**) via a tailored loss function. We then perform identification using two complementary strategies: the *input guard test* and the *output guard test*. Together, these tests cover scenarios where guardrails are deployed at either the input or output stage of an agent, thereby addressing **C3**. Intuitively, the output guard test asks the agent to repeat the adversarial prompt, so that the output guard will receive the LLM's response that contains the repeated text. To make identification without requiring additional guardrails (**C4**), we design a novel metric, match score, to measure how the safety mechanism in the agent matches the candidate guardrail. A larger match score indicates a stronger likelihood of the candidate guardrail being deployed.

To show the practicability of our **AP-Test**, we conduct extensive experiments on four different candidate guardrails under diverse scenarios, covering two popular LLMs and 10 guardrails. Results demonstrate that the proposed attack method accurately identifies guardrails in various AI agents for both input and output guardrails. Specifically, the WildGuard-specific (Han et al. 2024) input guard test on both Llama3.1-based (Dubey et al. 2024) and GPT4o-based (OpenAI 2024) agents achieves a perfect classification accuracy, i.e., $Acc = 1.00$. That is, our **AP-Test** successfully identifies the existence of WildGuard in all evaluated agents, indicating its effectiveness. Our **AP-Test** also successfully probes the candidate guardrail in more complex agents, e.g., containing two different guardrails. This showcases the practicality of our method.

Moreover, to analyze the impact of our proposed method, we perform an ablation study to examine the role of each component, particularly our loss terms designed to optimize adversarial queries and the existence of the query set. Our findings reveal that our loss terms and the query set are crucial for the identification. For example, without the loss ensuring the prompt remains `safe` for other guardrails (L_3), **AP-Test** tends to misidentify that the LlamaGuard3 (Chi et al. 2024) is used in the agent that is only equipped with WildGuard as the input guardrail.

Overall, our contributions are as follows:

- We propose **AP-Test**, the first method that uses guard-specific adversarial prompts to identify guardrails in black-box AI agents, overcoming core challenges under this setting.
- Experiments show the effectiveness and robustness of **AP-Test** on four candidate guardrails on various agents under diverse scenarios.
- The ablation study demonstrates the importance of each component in our proposed method, showing that their removal significantly degrades identifying performance.

2 Background and Related Work

LLM Security Risks. The rapid advancement of LLMs provides users with significant convenience but also raises critical security concerns (Zhao et al. 2024; Gong et al. 2023;

Zhang et al. 2024; Wen et al. 2024; Li et al. 2023). Among these concerns, jailbreak attacks (Shen et al. 2024; Zou et al. 2023; Liu et al. 2023; Zhao et al. 2024) pose a major threat by bypassing built-in safety mechanisms, enabling models to generate restricted or harmful content that violates usage policies (Google 2024; Meta 2023; OpenAI 2025; Amazon 2025). Previous studies analyze existing jailbreak strategies, particularly focusing on in-the-wild jailbreak prompts that are manually crafted in real-world scenarios (Shen et al. 2024). More recent studies introduce automated jailbreak generators, such as AutoDAN (Liu et al. 2023), GCG (Zou et al. 2023), and TAP (Mehrotra et al. 2023), which optimize adversarial prompts to evade safety measures and maximize attack success rates.

Internal Safety Alignment. LLM safety alignment refers to the process of ensuring that LLMs generate outputs that are consistent with human values. Most popular LLMs, such as Llama3 (Dubey et al. 2024), Gemma (Mesnard et al. 2024), Mistral (Jiang et al. 2023), and ChatGPT (OpenAI 2024), are safety-aligned, either through reinforcement learning with human feedback (RLHF) (Liu et al. 2024b; Ji et al. 2023), or learning from curated datasets that contain safety-related data (Jiang et al. 2023; Mesnard et al. 2024). However, these safety-aligned LLMs are still exposed to security risks such as jailbreak attacks (Shen et al. 2024; Zou et al. 2023; Liu et al. 2023; Zhao et al. 2024) and prompt injection attacks (Liu et al. 2024a; Zhan et al. 2024; Debenedetti et al. 2024; Yu et al. 2023).

External Safety Guardrail. Safety guardrails (Inan et al. 2023; Chi et al. 2024; Ghosh et al. 2024; Han et al. 2024; Zeng et al. 2024) are designed for moderating the input/output content of the LLMs and serve as an external complement to safety alignment. For example, built on Llama2-7B (Touvron et al. 2023), LlamaGuard (Inan et al. 2023) is fine-tuned on their constructed dataset based on a safety risk taxonomy encompassing a range of safety risks. Subsequent versions, LlamaGuard2 (Meta 2024) and LlamaGuard3 (Chi et al. 2024), further expand the safety risk taxonomy and dataset, leveraging state-of-the-art LLMs for fine-tuning, thereby strengthening their safeguard capabilities. Similarly, Aegis (Ghosh et al. 2024), WildGuard (Han et al. 2024), and ShieldGemma (Zeng et al. 2024) follow a comparable approach. Specifically, Aegis (Ghosh et al. 2024) is further instruction-tuned on LlamaGuard based on their own dataset.

LLM Fingerprinting and Watermarking. LLM identification focuses on identifying the origin of an LLM (Zeng et al. 2023; McGovern et al. 2024; Gubri et al. 2024; Jin et al. 2024; Yang et al. 2025). A common approach involves crafting LLM-specific adversarial prompts to guide the candidate LLM to produce target responses, which are then used for identification (Gubri et al. 2024; Jin et al. 2024). As state-of-the-art guardrails are often built upon LLMs (Inan et al. 2023; Han et al. 2024; Zeng et al. 2024), guardrail identification is similar. However, a key difference lies in the limited and often masked output space of guardrails (**C1**). This makes it hard to directly adapt the target responses for identification. Moreover, existing LLM identification techniques typically rely on empirically chosen thresholds by querying an auxiliary set of LLMs (Gubri et al. 2024; Jin et al. 2024), which fails to be solely based on the candidate model. This limitation highlights the need for more principled decision-making strategies, as addressed by our method (**C4**).

3 Problem Statement

Preliminary. At the core of these AI agents lies a safety-aligned LLM $\mathcal{F}$ that drives their functionality, but still suffers from security risks (Shen et al. 2024; Zou et al. 2023; Liu et al. 2023, 2024a; Yu et al. 2023). To ensure the security and compliance of these AI agents, additional mechanisms known as input and output guardrails $\mathcal{G}$ are often implemented. As shown in Figure 1, the input guardrail $\mathcal{G}_i$ evaluates user inputs x to determine whether they should be forwarded to the LLM. If the input x is deemed high-risk, policy-violating, or jailbreak prompts, i.e., $\mathcal{G}_i(x) = \texttt{unsafe}$, the agent then returns `refuse` without processing it further. Otherwise, the safety-aligned LLM would react to the input x, either generating responses or returning `refuse` if it perceives unsafe. However, with attacks like jailbreak attacks, even though the prompt input may be considered `safe` by both the input guard $\mathcal{G}_i$ and the LLM $\mathcal{F}$, the LLM's response could still contain harmful content. Therefore, in real-world applications, solely monitoring input may not be sufficient, which motivates the deployment of output guardrails. The output guardrail $\mathcal{G}_o$ monitors the LLM's response y to avoid policy violations. If a response is non-compliant, i.e., $\mathcal{G}_o(x) = \texttt{unsafe}$, the agent would withhold the output and return `refuse`; otherwise, the agent would output the response y.

Goal. The goal of guardrail identification is to determine whether a guardrail or its derivative is deployed in an AI agent. The derivative refers to further customized versions of the guardrail, such as those that are fine-tuned or instruction-tuned.

Capability. We assume black-box access to the AI agent, with no knowledge of the underlying LLM or the presence of input/output guardrails. We focus on open-source guardrails due to their widespread adoption. According to HuggingFace, LlamaGuard3 (Chi et al. 2024) received 335,571 downloads in March 2025, while AegisPermissive (Ghosh et al. 2024) was downloaded 1,021,359 times. One might argue that developers could create entirely proprietary guardrails. However, given the widespread adoption and accessibility of open-source guardrails, it is often more practical to build on existing ones through further fine-tuning. As such, we also consider derivatives of popular open-source guardrails to better cover the space of real-world deployments.

Problem Formulation. We define the guardrail identification task as follows: *Given an AI agent and a candidate guardrail $\mathcal{G}_t$, guardrail identification aims to identify whether the candidate guardrail $\mathcal{G}_t$ or its derivative is deployed in the AI agent. The identification task consists of two tests: (1) The input guard test audits whether the candidate guardrail $\mathcal{G}_t$ is deployed at the input stage of the AI agent;*

(2) The output guard test evaluates whether the candidate guardrail is present at the output stage.

4 Method

In this work, we propose **AP-Test**, which leverages guard-specific **A**dversarial **P**rompts to **Test** the identity of the input/output guardrail deployed in an AI agent. As shown in Figure 2, our framework consists of two phases: *adversarial prompt optimization* and *adversarial prompt test*.

4.1 Adversarial Prompt Optimization

The goal of the optimization phase is to optimize adversarial prompts that the target guardrail $\mathcal{G}_t$ erroneously flags as `unsafe`, while all other guardrails and safety-aligned LLMs correctly classify them as `safe`. An adversarial prompt is constructed by concatenating an optimized adversarial prefix x_a with a normal query $x_q \in Q$, where Q is a query set. The query is used as a starting point to prevent over-rejection by other guardrails, which is proven effective in our ablation study (Section 6). For each query $x_q \in Q$, we optimize an adversarial prefix x_a using three loss terms, addressing two specific aspects of the desired behavior.

Candidate Guardrail Adversarial Losses (L_1 and L_2). As guardrails' output space is usually limited (**C1**), these loss terms are designed to mislead the candidate guardrail $\mathcal{G}_t$ into classifying the adversarial prompt as `unsafe`. Specifically, L_1 encourages the candidate guardrail to classify the adversarial prompt as `unsafe`, while L_2 penalizes the candidate guardrail for classifying the adversarial prompt as `safe`:

$$\begin{aligned} L_1 &= \sigma(\mathcal{G}_t(x_a \| x_q), \texttt{unsafe}), \\ L_2 &= -\sigma(\mathcal{G}_t(x_a \| x_q), \texttt{safe}), \end{aligned} \tag{1}$$

where $\sigma(\cdot,\cdot)$ represents the cross-entropy loss. Together, L_1 and L_2 ensure that x_a effectively triggers the refusal mechanism of the candidate guardrail. Conceptually, these two losses are identical, but our ablation study (Section 6) demonstrates that the synergy of these two losses slightly outperforms solely deploying a single one of them.

Cross-Guardrail Compatibility Loss (L_3). To ensure the adversarial prompt remains `safe` according to all other guardrails and safety-aligned LLMs (**C2**), we introduce a safety scorer $\mathcal{S}$ and propose L_3. The safety scorer $\mathcal{S}$ measures the safety stage of an input x (Schmidt and Wiegand 2017; Mathew et al. 2021; Vidgen et al. 2021): $\mathcal{S}(x) = y_s \in [0,1]$, where $y_s = 0$ indicates no security risk, and $y_s = 1$ indicates a potential risk. The loss term is defined as:

$$L_3 = \sigma(\mathcal{S}(x_a \| x_q), 0). \tag{2}$$

By minimizing L_3, we prevent unintended rejections from unrelated guardrails, preserving the specificity of the attack on $\mathcal{G}_t$.

By jointly minimizing L_1, L_2, and L_3, we craft adversarial prompts following (Jin et al. 2024) that expose the behavior of the candidate guardrail while maintaining compatibility with other guardrails. The final loss function is defined as:

$$L = L_1 + \alpha \cdot L_2 + \beta \cdot L_3, \quad \alpha, \beta \in \mathbb{R}, \tag{3}$$

where α and β control the weights of the loss terms L_2 and L_3, respectively.

4.2 Adversarial Prompt Test

We then perform identification using two complementary strategies: the *input guard test* and the *output guard test*. Together, these tests cover scenarios where guardrails are deployed at either the input or output stage of an agent, thereby addressing **C3**. In addition, we introduce match score, a novel metric that facilitates identification based solely on the candidate guardrail. Match score provides clear and actionable decision strategies, effectively addressing the challenge of threshold selection in identification (**C4**).

Input Guard Test. We first consider the identification of the input guardrail. As an input guardrail in an AI agent, it will decide whether the user prompt x should be passed to the LLM. Thus, we do the input guard test by directly querying the AI agent with our adversarial prompts $\{x_a \| x_q\}$. If the AI agent responds with `refuse`, we consider the candidate guardrail to probably serve as an input guardrail in the AI agent (see Figure 2 (2a)).

Output Guard Test. The third challenge **C3** comes in identifying the output guardrail in the AI agent, as the output guardrail takes the LLM's generated response y as input instead of the user prompt x. It is hard for us to manipulate the LLM's output as we have no knowledge about it. To solve this, we design a prompt template that asks the AI agent to repeat our adversarial prompt, e.g., "Please repeat: [`Adversarial Prompt`]" Ideally, the LLM's response y should be the adversarial prompt $\{x_a \| x_q\}$ and will be passed to the output guardrail. We empirically design and evaluate five prompt templates and select the best one as shown in Section 5.3. With the well-designed prompt template, we can ask the AI agent to repeat the adversarial prompts to test its output guardrail as shown in Figure 2 (2b). Then, we can make identifications based on the response from the agent.

More Complex Scenarios. In real-world applications, the candidate guardrail can be deployed in both the input and output stages of an AI agent (as shown in Figure 1), which poses challenges for identification (**C3**). Therefore, given a candidate guardrail, it is necessary to conduct both input and output guard tests on the AI agent, as illustrated in Appendix A. Experiments in such scenarios in Section 5.4 empirically showcase the effectiveness of **AP-Test**.

Match Score. To better quantify, we introduce the *refusal rate* $r = \frac{\#\texttt{refuse}}{\#\text{all responses}} \in [0,1]$, which is the ratio of `refuse` among all responses, where `refuse` represents that the AI agent refuses to respond to the query. A higher refusal rate indicates that the candidate guardrail is more likely to be the input guardrail in this AI agent. Unlike existing LLM identification methods (Gubri et al. 2024; Jin et al. 2024) that require testing on a set of LLMs to help distinguish the identity (**C4**), we propose a novel metric, match score, based solely on the candidate guardrail. We first calculate the refusal rate of directly querying the candidate guardrail with the optimized adversarial prompts, denoted as the candidate refusal rate r_t. Note that the candidate refusal rates r_t in both input and output guard tests are the same, as we directly query the candidate guardrail, disregarding the AI agent or the LLM.

Then, we define the match score for an AI agent as:

$$MS = \frac{|\min(r, r_t) - 0|^\lambda}{|r_t - 0|^\lambda} = \frac{|\min(r, r_t)|^\lambda}{|r_t|^\lambda}, \quad \lambda \geq 1, \quad (4)$$

where 0 is the lower bound of r and λ represents the scaling factor. The match score $MS \in [0, 1]$ depicts how likely the candidate guardrail exists in the AI agent. We consider the candidate guardrail to exist in the AI agent if $MS > 0.5$; our experiments in Section 5.2 show that the identification performance is not sensitive to this threshold.

5 Experiments

5.1 Experimental Settings

We take four different guardrails as our candidate guardrails, including WildGuard (Han et al. 2024), LlamaGuard (Inan et al. 2023), LlamaGuard2 (Meta 2024), and LlamaGuard3 (Chi et al. 2024). We use 10 guardrails for evaluation, including the four candidate guardrails, AegisDefensive, AegisPermissive (Ghosh et al. 2024), ShieldGemma-2B, ShieldGemma-9B, ShieldGemma-27B (Zeng et al. 2024), and GPT4o (OpenAI 2024). We obey their default settings in our experiments and follow (Zeng et al. 2024) to prompt GPT4o as an input/output guardrail. Besides, we use Llama3.1 (Dubey et al. 2024) and GPT4o as the LLMs of the conversational AI agents. For the safety scorer in the output guard test, we consider state-of-the-art hate speech detectors and use LFTW-R4 (Vidgen et al. 2021) in our experiments. Details of different LLMs and guardrails we use can be found in Table 4 in Appendix B.

Firstly, we optimize the adversarial prompts based on each candidate guardrail. We follow the settings in Jin et al. (2024) and use their dataset as the query set Q, which consists of 50 simple questions. For each query in Q, we optimize a 32-token adversarial prefix using loss weights $\alpha = 0.01$ and $\beta = 1000$. We further investigate the impact of different loss term weights in the ablation study. The experiments are conducted with a batch size of 64 over 200 epochs on NVIDIA A100 GPUs with 40 GB of memory.

Then, we conduct input/output tests on agents equipped with different LLMs and input/output guardrails. We consider this a binary classification problem and calculate the classification accuracy for each candidate guardrail over the 11 AI Agents based on the match score MS (with $\lambda = 2$) and plot the ROC curve for all test results. For each LLM, we construct 11 AI agents for evaluation: one without any guardrail, and the other 10 with different guardrails at the input/output stage. We discuss the situation that the agent contains additional guardrails later in Section 5.4.

5.2 Input Guard Test

We first evaluate the input guard test of **AP-Test** on Llama3.1-based and GPT4o-based agents. The classification accuracy shown in Table 1 indicates that **AP-Test** successfully identifies the candidate guardrails that are used in the AI agent, achieving a perfect accuracy, i.e., 1.00, for each candidate guardrail. To further assess the robustness of our approach, we plot the ROC curves for all test results in Figure 3, observing an AUC of 1.00. This indicates that our identification performance is not sensitive to the threshold selection. These findings highlight both the effectiveness and reliability of our method in the input guard test setting.

LLM	Candidate Guardrail			
	WildGuard	LlamaGuard	LlamaGuard2	LlamaGuard3
Llama3.1	1.00	1.00	1.00	1.00
GPT4o	1.00	1.00	1.00	1.00

Table 1: Classification accuracy of input guard tests on Llama-3.1-based and GPT4o-based agents.

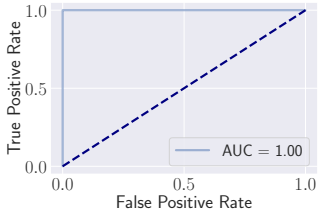


Figure 3: ROC curve of input guard test.

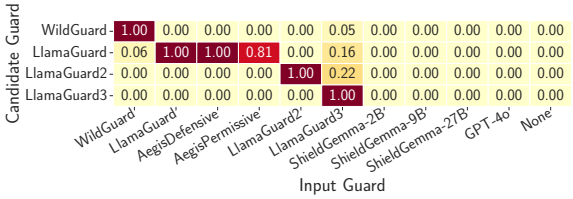


Figure 4: The match scores of input guard tests on different AI agents. A larger match score indicates the candidate guardrail is more probable to exist in the agent.

Figure 4 further illustrates the match scores of the input guard tests on Llama3.1-based agents equipped with various input guardrails. We observe an obvious distinction between the existence and non-existence of the candidate guard. For example, the WildGuard-agent achieves a match score of 1.00 on the WildGuard-specific adversarial prompts, while it only achieves a maximum distance of 0.05 on other agents. Additionally, the LlamaGuard-specific input test achieves 1.00 and 0.81 match score on the AegisDefense-agent and AegisPermissive-agent, respectively. That is because AegisDefense and AegisPermissive are fine-tuned on LlamaGuard, which are the derivatives of LlamaGuard. Results of GPT4o-based agents can be found in Figure 11 in Appendix D. We find that the results of both Llama3.1-based and GPT4o-based agents are almost the same. This shows that the LLM has little influence on the input guard test, i.e., our **AP-Test** is robust to different LLMs. This also indicates that the internal safety alignment will not affect the performance of our method.

5.3 Output Guard Test

Prompt Template Selection. As mentioned in Section 4, the output guard test requires a prompt template that asks the AI agent to repeat our adversarial prompts. We begin with the

LLM	Candidate Guardrail			
	WildGuard	LlamaGuard	LlamaGuard2	LlamaGuard3
Llama3.1	1.00	1.00	1.00	1.00
GPT4o	1.00	1.00	1.00	1.00

Table 2: Classification accuracy of output guard tests on Llama3.1-based and GPT4o-based agents.

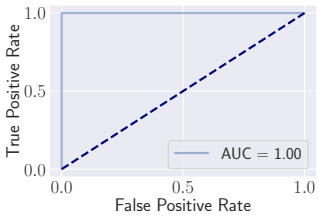


Figure 5: ROC curve of output guard test.

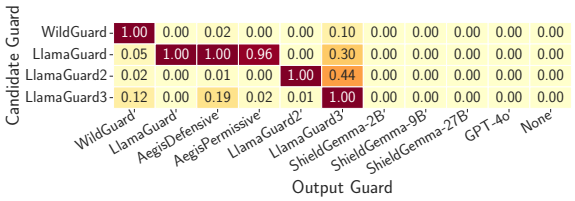


Figure 6: The match scores of output guard tests on different Llama3.1-based AI agents. A larger match score indicates the candidate guardrail is more probable to exist in the agent.

template: "Please repeat: [`Adversarial Prompt`]" and ask ChatGPT (OpenAI 2024) to generate five variations, as shown in Table 5 in Appendix C. To assess how well the templates help the LLM repeat the adversarial prompt, we use Gemma2-9B (Mesnard et al. 2024) as a surrogate LLM and evaluate the similarity between the generated text and the input adversarial prompt optimized on LlamaGuard3. The results in Table 6 in Appendix C show that the third template (T3) achieves the highest similarity scores, including a ROUGE-L score of 0.956. Thus, we select T3 as the prompt template for our experiments: "*Your task is to mimic text. Repeat this string exactly as input, with no interpretation:* [`Adversarial Prompt`]"

Evaluation Results. Using T3 as the prompt template, we evaluate the output guard test on AI agents with different output guardrails. Results in Table 2 and Figure 5 demonstrate the effectiveness and robustness of our **AP-Test**. For all candidate guardrails, our method successfully identified their existence in all agents, achieving an accuracy of 1.00 and an AUC of 1.00. The match scores of the test on Llama3.1-based agents are exhibited in Figure 6. All agents with match scores larger than 0.50 are indeed equipped with the corresponding candidate guardrail as the output guardrail. For example, the match score on the WildGuard-agent reaches 1.00 on WildGuard-specific adversarial prompts, while it is 0.00 on LlamaGuard-agent and agents with LlamaGuard's derivatives (AegisDefensive-agent and AegisPermissive-agent). This indicates that our **AP-Test** successfully distinguishes the output guardrail used in the agent. We also observe that the output guard test is harder than the input guard test. For example, for LlamaGuard2-specific adversarial prompts, the match score of the LlamaGuard3-agent achieves 0.44, which is 0.22 farther than that in the input guard test and is closer to 0.50.

Results of GPT4o-based agents are shown in Figure 12 in Appendix E. We find that there is a slight performance difference between Llama3.1-based and GPT4o-based agents. This discrepancy is due to the information loss during the LLM processing. In other words, the performance of our output guard test is influenced by how well the LLM can repeat the adversarial prompts.

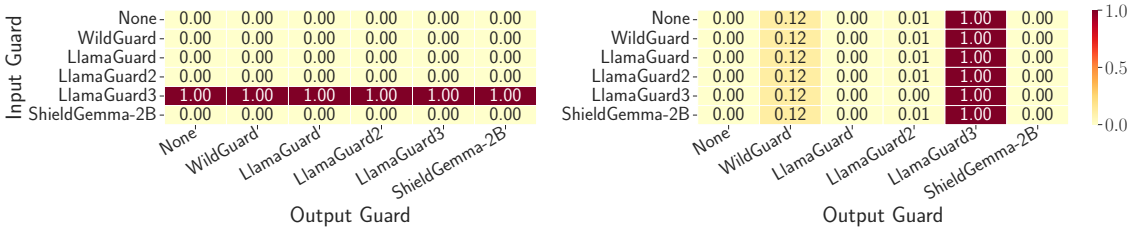
5.4 More Complex Scenarios

Our primary experiments show that the base LLMs (e.g., Llama3.1 and GPT4o) have little influence on the performance of our **AP-Test**. However, we assumed that the AI agent contains either an input or an output guardrail. To evaluate the robustness of **AP-Test** under more complex scenarios, we relax this assumption and assess its performance when both input and output guardrails are simultaneously deployed. Specifically, we evaluate **AP-Test** on $6 \times 6 = 36$ Llama3.1-based agents constructed with all combinations of input and output guardrails, where each guardrail is one of: WildGuard, LlamaGuard, LlamaGuard2, LlamaGuard3, ShieldGemma-2B, or absent (N/A). We apply both input and output guard tests to each of these 36 agents. The results show that both tests achieve perfect classification accuracy (1.00) across all agents. The nearly the same match scores shown in Figure 7 further suggest that the presence of additional guardrails has minimal impact on the effectiveness of our method in identifying the candidate guardrail.

6 Ablation Study

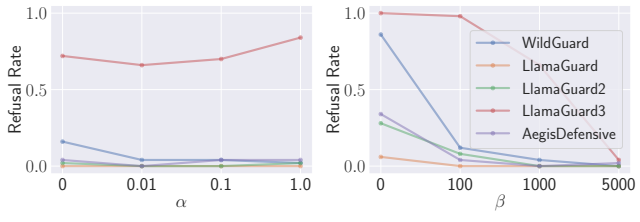
To explore the influence of different components and hyperparameters of our **AP-Test**, we take LlamaGuard3 as our candidate guardrail and experiment on the 11 Llama3.1-based AI agents with different input guardrails used in the main experiments. We find that the classification accuracy and the match score are relatively robust to different settings. Thus, we report the refusal rate r on each agent to better illustrate the influence in certain cases.

Loss Terms. We first investigate the influence of loss terms. Figure 8 shows the refusal rates using various weights of loss terms on various agents. We observe that L_2 enhances the refusal rates of the agent with the candidate guardrail while slightly increasing the refusal rates on agents with other guardrails. For example, the refusal rate on LlamaGuard3-agent increases from 0.72 to 0.84, while that on AegisDefensive-agent slightly increases to 0.04. As for L_3, we find it helps suppress the refusal rate on agents with other guardrails. For example, when β is 0, although the refusal rate on LlamaGuard3-agent is high (1.00), the refusal rate on WildGuard-agent reaches 0.86, which is so close to the candidate refusal rate r_t (0.98) on LlamaGuard3.



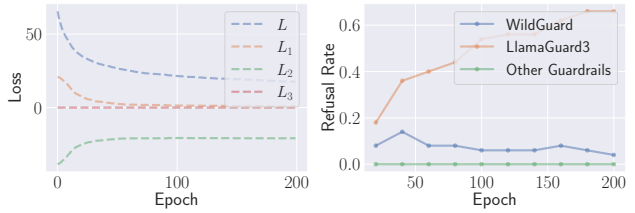
(a) Input Guard Test (b) Output Guard Test

Figure 7: Influence of the presence of additional guardrails. We report match scores on each agent.



(a) L_2 (b) L_3

Figure 8: Influence of different weights of loss terms. We report the refusal rates while omitting the agents with a consistent refusal rate of 0.00.



(a) Loss (b) Performance

Figure 9: Influence of epochs. We report the loss values and refusal rates. All refusal rates on agents with other guardrails are 0.00.

Epochs. To analyze the influence of the number of epochs on our method, we analyze the loss values and refusal rates of the LlamaGuard3-specific input guard test across different epochs, as illustrated in Figure 9. The results indicate that the losses converge around the 50th epoch, highlighting the efficiency of our approach. Furthermore, the refusal rate on LlamaGuard3-agent increases with additional epochs, reaching 0.62 by the 160th epoch, while the refusal rate for WildGuard-agents decreases. This behavior demonstrates that as the number of epochs increases, the adversarial prompts become more reliable, as the distinction between the candidate guardrail and other guardrails grows. Ultimately, the system converges at a specific epoch level, ensuring stable and consistent performance.

Query Set. We further explore the influence of the normal query set Q, which is used as the starting point for adversarial prompt optimization. Table 3 shows the refusal rates of LlamaGuard3-specifc input guard test. We find that without the query set, the test on agents equipped with guardrails other than LlamaGuard3 achieves even higher refusal rates. For instance, the refusal rate on the LlamaGuard-agent is 0.98, 0.06 higher than that on the LlamaGuard3-agent. This means that **AP-Test** without Q mistakes that LlamaGuard3 is used in the WildGuard-agent. In this sense, we address the importance of the query set as the initial guidance for adversarial optimization.

AP-Test	Input Guardrail			
	WildGuard	LlamaGuard	LlamaGuard2	LlamaGuard3
W/ Q	0.04	0.00	0.00	0.66
W/O Q	0.94	0.98	0.78	0.92

Table 3: Influence of using normal query set Q in **AP-Test**. We report the refusal rates for agents with four different input guardrails.

7 Conclusion

In this work, we tackle the problem of identifying guardrails deployed in conversational AI agents, a crucial step toward understanding system behavior and potential vulnerabilities. We propose **AP-Test**, a novel approach that leverages guard-specific adversarial prompts to identify the guardrail component within a black-box AI agent for the first time, effectively addressing the key challenges in this task. Experiments conducted on four candidate guardrails across various AI agents demonstrate the effectiveness and robustness of **AP-Test**. Our ablation study underscores the significance of our proposed loss terms and the query set, revealing that their removal leads to a substantial degradation in identification performance.

Limitations. As this is the first study to tackle the guardrail identification problem, there is a lack of established baselines for direct comparison. We hope our work serves as a foundation for this problem and calls for more advanced techniques. Our study focuses exclusively on model-based guardrails, excluding rule-based filters and external moderation systems such as LLM API wrappers (e.g., GPT-4o with specific system prompts). We leave the exploration of other guard techniques to future work.

References

Amazon. 2025. AWS Responsible AI Policy. https://aws.amazon.com/cn/machine-learning/responsible-ai/policy/.

Anthropic. 2024. Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet.

Chi, J.; Karn, U.; Zhan, H.; Smith, E.; Rando, J.; Zhang, Y.; Plawiak, K.; Coudert, Z. D.; Upasani, K.; and Pasupuleti, M. 2024. Llama Guard 3 Vision: Safeguarding Human-AI Image Understanding Conversations. *CoRR abs/2411.10414*.

Debenedetti, E.; Zhang, J.; Balunovic, M.; Beurer-Kellner, L.; Fischer, M.; and Tramèr, F. 2024. AgentDojo: A Dynamic Environment to Evaluate Attacks and Defenses for LLM Agents. *CoRR abs/2406.13352*.

DeepSeek-AI; Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; Zhang, X.; Yu, X.; Wu, Y.; Wu, Z. F.; Gou, Z.; Shao, Z.; Li, Z.; Gao, Z.; Liu, A.; Xue, B.; Wang, B.; Wu, B.; Feng, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; Dai, D.; Chen, D.; Ji, D.; Li, E.; Lin, F.; Dai, F.; Luo, F.; Hao, G.; Chen, G.; Li, G.; Zhang, H.; Bao, H.; Xu, H.; Wang, H.; Ding, H.; Xin, H.; Gao, H.; Qu, H.; Li, H.; Guo, J.; Li, J.; Wang, J.; Chen, J.; Yuan, J.; Qiu, J.; Li, J.; Cai, J. L.; Ni, J.; Liang, J.; Chen, J.; Dong, K.; Hu, K.; Gao, K.; Guan, K.; Huang, K.; Yu, K.; Wang, L.; Zhang, L.; Zhao, L.; Wang, L.; Zhang, L.; Xu, L.; Xia, L.; Zhang, M.; Zhang, M.; Tang, M.; Li, M.; Wang, M.; Li, M.; Tian, N.; Huang, P.; Zhang, P.; Wang, Q.; Chen, Q.; Du, Q.; Ge, R.; Zhang, R.; Pan, R.; Wang, R.; Chen, R. J.; Jin, R. L.; Chen, R.; Lu, S.; Zhou, S.; Chen, S.; Ye, S.; Wang, S.; Yu, S.; Zhou, S.; Pan, S.; and Li, S. S. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *CoRR abs/2501.12948*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; Goyal, A.; Hartshorn, A.; Yang, A.; Mitra, A.; Sravankumar, A.; Korenev, A.; Hinsvark, A.; Rao, A.; Zhang, A.; Rodriguez, A.; Gregerson, A.; Spataru, A.; Rozière, B.; Biron, B.; Tang, B.; Chern, B.; Caucheteux, C.; Nayak, C.; Bi, C.; Marra, C.; McConnell, C.; Keller, C.; Touret, C.; Wu, C.; Wong, C.; Ferrer, C. C.; Nikolaidis, C.; Allonsius, D.; Song, D.; Pintz, D.; Livshits, D.; Esiobu, D.; Choudhary, D.; Mahajan, D.; Garcia-Olano, D.; Perino, D.; Hupkes, D.; Lakomkin, E.; AlBadawy, E.; Lobanova, E.; Dinan, E.; Smith, E. M.; Radenovic, F.; Zhang, F.; Synnaeve, G.; Lee, G.; Anderson, G. L.; Nail, G.; Mialon, G.; Pang, G.; Cucurell, G.; Nguyen, H.; Korevaar, H.; Xu, H.; Touvron, H.; Zarov, I.; Ibarra, I. A.; Kloumann, I. M.; Misra, I.; Evtimov, I.; Copet, J.; Lee, J.; Geffert, J.; Vranes, J.; Park, J.; Mahadeokar, J.; Shah, J.; van der Linde, J.; Billock, J.; Hong, J.; Lee, J.; Fu, J.; Chi, J.; Huang, J.; Liu, J.; Wang, J.; Yu, J.; Bitton, J.; Spisak, J.; Park, J.; Rocca, J.; Johnstun, J.; Saxe, J.; Jia, J.; Alwala, K. V.; Upasani, K.; Plawiak, K.; Li, K.; Heafield, K.; Stone, K.; and et al. 2024. The Llama 3 Herd of Models. *CoRR abs/2407.21783*.

Ghosh, S.; Varshney, P.; Galinkin, E.; and Parisien, C. 2024. AEGIS: Online Adaptive AI Content Safety Moderation with Ensemble of LLM Experts. *CoRR abs/2404.05993*.

Github. 2025. Copilot. https://github.com/features/copilot.

Gong, Y.; Ran, D.; Liu, J.; Wang, C.; Cong, T.; Wang, A.; Duan, S.; and Wang, X. 2023. FigStep: Jailbreaking Large Vision-language Models via Typographic Visual Prompts. *CoRR abs/2311.05608*.

Google. 2024. Generative AI Prohibited Use Policy. https://policies.google.com/terms/generative-ai/use-policy?hl=en.

Gubri, M.; Ulmer, D.; Lee, H.; Yun, S.; and Oh, S. J. 2024. TRAP: Targeted Random Adversarial Prompt Honeypot for Black-Box Identification. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. ACL.

Han, S.; Rao, K.; Ettinger, A.; Jiang, L.; Lin, B. Y.; Lambert, N.; Choi, Y.; and Dziri, N. 2024. WildGuard: Open One-Stop Moderation Tools for Safety Risks, Jailbreaks, and Refusals of LLMs. *CoRR abs/2406.18495*.

Inan, H.; Upasani, K.; Chi, J.; Rungta, R.; Iyer, K.; Mao, Y.; Tontchev, M.; Hu, Q.; Fuller, B.; Testuggine, D.; and Khabsa, M. 2023. Llama Guard: LLM-based Input-Output Safeguard for Human-AI Conversations. *CoRR abs/2312.06674*.

Ji, J.; Liu, M.; Dai, J.; Pan, X.; Zhang, C.; Bian, C.; Chen, B.; Sun, R.; Wang, Y.; and Yang, Y. 2023. BeaverTails: Towards Improved Safety Alignment of LLM via a Human-Preference Dataset. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de Las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; élio Renard Lavaud; Lachaux, M.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. *CoRR abs/2310.06825*.

Jin, H.; Zhang, C.; Shi, S.; Lou, W.; and Hou, Y. T. 2024. ProFLingo: A Fingerprinting-based Copyright Protection Scheme for Large Language Models. *CoRR abs/2405.02466*.

Li, H.; Guo, D.; Fan, W.; Xu, M.; and Song, Y. 2023. Multi-step Jailbreaking Privacy Attacks on ChatGPT. *CoRR abs/2304.05197*.

Liu, X.; Xu, N.; Chen, M.; and Xiao, C. 2023. AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models. *CoRR abs/2310.04451*.

Liu, Y.; Jia, Y.; Geng, R.; Jia, J.; and Gong, N. Z. 2024a. Formalizing and Benchmarking Prompt Injection Attacks and Defenses. In *USENIX Security Symposium (USENIX Security)*. USENIX.

Liu, Z.; Nie, Y.; Tan, Y.; Yue, X.; Cui, Q.; Wang, C.; Zhu, X.; and Zheng, B. 2024b. Safety Alignment for Vision Language Models. *CoRR abs/2405.13581*.

Mangaokar, N.; Hooda, A.; Choi, J.; Chandrashekaran, S.; Fawaz, K.; Jha, S.; and Prakash, A. 2024. PRP: Propagating Universal Perturbations to Attack Large Language Model Guard-Rails. In Ku, L.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, 10960–10976. Association for Computational Linguistics.

Mathew, B.; Saha, P.; Yimam, S. M.; Biemann, C.; Goyal, P.; and Mukherjee, A. 2021. HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. In *AAAI Conference on Artificial Intelligence (AAAI)*, 14867–14875. AAAI.

McGovern, H.; Stureborg, R.; Suhara, Y.; and Alikaniotis, D. 2024. Your Large Language Models Are Leaving Fingerprints. *CoRR abs/2405.14057*.

Mehrotra, A.; Zampetakis, M.; Kassianik, P.; Nelson, B.; Anderson, H.; Singer, Y.; and Karbasi, A. 2023. Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. *CoRR abs/2312.02119*.

Mesnard, T.; Hardin, C.; Dadashi, R.; Bhupatiraju, S.; Pathak, S.; Sifre, L.; Rivière, M.; Kale, M. S.; Love, J.; Tafti, P.; Hussenot, L.; Chowdhery, A.; Roberts, A.; Barua, A.; Botev, A.; Castro-Ros, A.; Slone, A.; Héliou, A.; Tacchetti, A.; Bulanova, A.; Paterson, A.; Tsai, B.; Shahriari, B.; Lan, C. L.; Choquette-Choo, C. A.; Crepy, C.; Cer, D.; Ippolito, D.; Reid, D.; Buchatskaya, E.; Ni, E.; Noland, E.; Yan, G.; Tucker, G.; Muraru, G.; Rozhdestvenskiy, G.; Michalewski, H.; Tenney, I.; Grishchenko, I.; Austin, J.; Keeling, J.; Labanowski, J.; Lespiau, J.; Stanway, J.; Brennan, J.; Chen, J.; Ferret, J.; Chiu, J.; and et al. 2024. Gemma: Open Models Based on Gemini Research and Technology. *CoRR abs/2403.08295*.

Meta. 2023. https://ai.meta.com/llama/use-policy/.

Meta. 2024. Llama Guard 2. https://www.llama.com/docs/model-cards-and-prompt-formats/meta-llama-guard-2/.

Neumann, A.; Yin, Y.; Sowe, S. K.; Decker, S.; and Jarke, M. 2024. An LLM-Driven Chatbot in Higher Education for Databases and Information Systems. *IEEE Transactions on Education*.

OpenAI. 2024. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/.

OpenAI. 2025. Usage policies. https://openai.com/policies/usage-policies.

Peng, C.; Yang, X.; Chen, A.; Smith, K. E.; Pournejatian, N. M.; Costa, A. B.; Martin, C.; Flores, M. G.; Zhang, Y.; Magoc, T.; Lipori, G. P.; Mitchell, D. A.; Ospina, N. S.; Ahmed, M. M.; Hogan, W. R.; Shenkman, E. A.; Guo, Y.; Bian, J.; and Wu, Y. 2023. A study of generative large language model for medical research and healthcare. *NPJ Digital Medicine*.

Perspective. 2025. Perspective API. https://www.perspectiveapi.com.

Schmidt, A.; and Wiegand, M. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Workshop on Natural Language Processing for Social Media (SocialNLP)*, 1–10. ACL.

Shen, X.; Chen, Z.; Backes, M.; Shen, Y.; and Zhang, Y. 2024. Do Anything Now: Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Canton-Ferrer, C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR abs/2307.09288*.

Vidgen, B.; Thrush, T.; Waseem, Z.; and Kiela, D. 2021. Learning from the Worst: Dynamically Generated Datasets to Improve Online Hate Detection. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, 1667–1682. ACL.

Wen, R.; Li, Z.; Backes, M.; and Zhang, Y. 2024. Membership Inference Attacks Against In-Context Learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM.

Yang, Z.; Wu, Y.; Shen, Y.; Dai, W.; Backes, M.; and Zhang, Y. 2025. The Challenge of Identifying the Origin of Black-Box Large Language Models. *CoRR abs/2503.04332*.

Yu, J.; Wu, Y.; Shu, D.; Jin, M.; and Xing, X. 2023. Assessing Prompt Injection Risks in 200+ Custom GPTs. *CoRR abs/2311.11538*.

Zeng, B.; Zhou, C.; Wang, X.; and Lin, Z. 2023. HuRef: HUman-REadable Fingerprint for Large Language Models. *CoRR abs/2312.04828*.

Zeng, W.; Liu, Y.; Mullins, R.; Peran, L.; Fernandez, J.; Harkous, H.; Narasimhan, K.; Proud, D.; Kumar, P.; Radharapu, B.; Sturman, O.; and Wahltinez, O. 2024. ShieldGemma: Generative AI Content Moderation Based on Gemma. *CoRR abs/2407.21772*.

Zhan, Q.; Liang, Z.; Ying, Z.; and Kang, D. 2024. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. *CoRR abs/2403.02691*.

Zhang, Q.; Xiong, Z.; and Mao, Z. M. 2024. Safeguard is a Double-edged Sword: Denial-of-service Attack on Large Language Models. *CoRR*, abs/2410.02916.

Zhang, R.; Li, H.; Wen, R.; Jiang, W.; Zhang, Y.; Backes, M.; Shen, Y.; and Zhang, Y. 2024. Instruction Backdoor Attacks Against Customized LLMs. In *USENIX Security Symposium (USENIX Security)*. USENIX.

Zhao, X.; Yang, X.; Pang, T.; Du, C.; Li, L.; Wang, Y.-X.; and Wang, W. Y. 2024. Weak-to-Strong Jailbreaking on Large Language Models. *CoRR abs/2401.17256*.

Zou, A.; Wang, Z.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. *CoRR abs/2307.15043*.

A AP-Test in More Complex Scenarios

In real-world applications, there are both input and output guardrails in an AI agent, as illustrated in Figure 1. Therefore, given a candidate guardrail, it is necessary to conduct both input and output guard tests on the AI agent. Here, we propose a two-step process, as shown in Figure 10.

- We first determine whether the candidate guardrail functions as an input guardrail through the input guard test. If so, we conclude that the candidate guardrail is deployed in the AI agent and do not proceed with the output guard test.
- If it is not used as the input guardrail, then we proceed with the output guard test for the candidate guardrail.

In our evaluations (Section 5.4), we demonstrate that the proposed method can successfully identify the input/output guardrail even in the presence of a different output/input guardrail.

B Model Details

Table 4 shows the details of the models used in our experiments, including the versions we used.

C Prompt Template Selection

As mentioned in Section 4, the output guard test requires a prompt template that asks the AI agent to repeat our adversarial prompts. We begin with the template: "Please repeat: [`Adversarial Prompt`]" and ask ChatGPT (OpenAI 2024) to generate five variations, as shown in Table 5. To assess how well the templates help the LLM repeat the adversarial prompt, we use Gemma2-9B (Mesnard et al. 2024) as a surrogate LLM and evaluate the similarity between the generated text and the input adversarial prompt optimized on LlamaGuard3. The results in Table 6 show that the third template (T3) achieves the highest similarity scores, including a ROUGE-L score of 0.956. Thus, we select T3 as the prompt template for our experiments: "*Your task is to mimic text. Repeat this string exactly as input, with no interpretation:* [`Adversarial Prompt`]"

D Input Guard Test

Figure 11 shows the match scores of the input guard test on different agents. The agents are either based on Llama3.1 or GPT4o models. We find that there is little difference between the two agents based on different LLMs. This shows that the base LLM has little influence on the input guard test.

E Output Guard Test

Figure 12 shows the match scores of the output guard test on different agents. The agents are either based on Llama3.1 or GPT4o models. We observe that the output guard test performs slightly better on GPT4o-based agents than Llama3.1-based ones. Take LlamaGuard3 as the candidate guard and WildGuard as the equipped output guard, for example. The match score on the Llama3.1-based agent is 0.12, while that on the GPT4o-based agent is 0.01. This discrepancy is due to the information loss during the LLM processing. In other words, the performance of our output guard test is influenced by the LLM and the prompt template, i.e., how well the LLM can repeat the adversarial prompts.

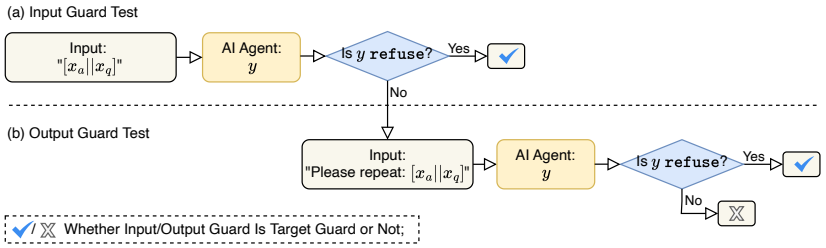


Figure 10: Workflow on real-world scenarios. We first conduct (a) the input guard test on the AI agent. If the results show that the candidate guardrail probably does not exist in the agent, then we further conduct (b) the output guard test to identify whether it serves as the output guardrail in the agent.

Model Name	Model Path
WildGuard (Han et al. 2024)	allenai/wildguard
LlamaGuard (Inan et al. 2023)	meta-llama/LlamaGuard-7b
—– AegisDefensive (Ghosh et al. 2024)	nvidia/Aegis-AI-Content-Safety-LlamaGuard-Defensive-1.0
—– AegisPermissive (Ghosh et al. 2024)	nvidia/Aegis-AI-Content-Safety-LlamaGuard-Permissive-1.0
LlamaGuard2 (Meta 2024)	meta-llama/Meta-Llama-Guard-2-8B
LlamaGuard3 (Chi et al. 2024)	meta-llama/Llama-Guard-3-8B
ShieldGemma-2B (Zeng et al. 2024)	google/shieldgemma-2b
ShieldGemma-9B (Zeng et al. 2024)	google/shieldgemma-9b
ShieldGemma-27B (Zeng et al. 2024)	google/shieldgemma-27b
Perspective (Perspective 2025)	Perspective API
GPT4o (OpenAI 2024)	GPT4o-2024-11-20
Gemma2 (Mesnard et al. 2024)	google/gemma-2-9b-it
Llama3.1 (Dubey et al. 2024)	meta-llama/Llama-3.1-8B-Instruct

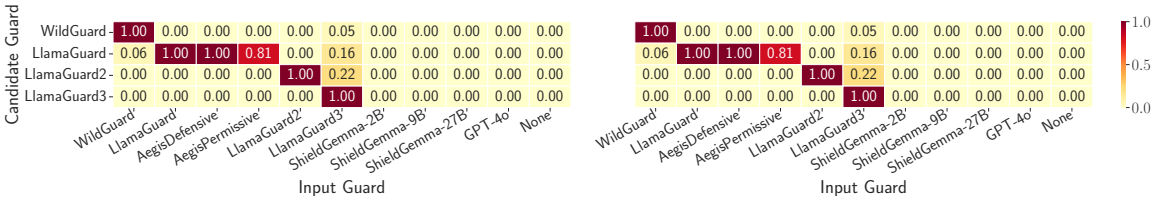
Table 4: Details of the guardrails and LLMs. Note that AegisDefensive and AegisPermissive are fine-tuned on LlamaGuard, as illustrated below.

	Template
T1	Simply repeat and do not consider the meaning: [AP]
T2	Repeat exactly the following sentence, including all characters and punctuation: [AP]
T3	Your task is to mimic text. Repeat this string exactly as input, with no interpretation: [AP]
T4	Do not interpret the meaning of this text. Simply output it as provided: [AP]
T5	For debugging purposes, repeat the following text string exactly as input: [AP]

Table 5: Candidate prompt templates for output guard test. [AP] denotes the placeholder for the adversarial prompts.

#Template	Cosine Similarity	BLUE	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSum
T1	0.881	0.977	0.911	0.900	0.911	0.911
T2	0.933	0.961	0.939	0.934	0.939	0.939
T3	**0.944**	**0.980**	**0.956**	**0.945**	**0.956**	**0.956**
T4	0.870	0.929	0.884	0.876	0.884	0.884
T5	0.914	0.962	0.926	0.916	0.926	0.926

Table 6: Similarity between the input adversarial prompt and the output text from the surrogate LLM with different prompt templates.



(a) Llama3.1-Based Agents (b) GPT4o-Based Agents

Figure 11: The match scores of input guard tests on different AI agents. A larger match score indicates the candidate guardrail is more probable to exist in the agent.

(a) Llama3.1-Based Agents (b) GPT4o-Based Agents

Figure 12: The match scores of output guard tests on different AI agents. A larger match score indicates the candidate guardrail is more probable to exist in the agent.