AGENTVIGIL: Automatic Black-Box Red-teaming for Indirect Prompt Injection against LLM Agents

Zhun Wang[1], Vincent Siu[2], Zhe Ye[1], Tianneng Shi[1], Yuzhou Nie[3], Xuandong Zhao[1], Chenguang Wang[2], Wenbo Guo[3], Dawn Song[1]

[1] University of California, Berkeley [2] University of California, Santa Cruz
[3] University of California, Santa Barbara

Correspondence: zhun.wang@berkeley.edu

Abstract

There emerges a critical security risk of LLM agents: indirect prompt injection, a sophisticated attack vector that compromises the core of these agents, the LLM, by manipulating contextual information rather than direct user prompts. In this work, we propose a generic black-box optimization framework, AGENTVIGIL, designed to automatically discover and exploit indirect prompt injection vulnerabilities across diverse LLM agents. Our approach starts by constructing a high-quality initial seed corpus, then employs a seed selection algorithm based on Monte Carlo Tree Search (MCTS) to iteratively refine inputs, thereby maximizing the likelihood of uncovering agent weaknesses. We evaluate AGENTVIGIL on two public benchmarks, AgentDojo and VWA-adv, where it achieves 71% and 70% success rates against agents based on o3-mini and GPT-4o, respectively, nearly doubling the performance of handcrafted baseline attacks. Moreover, AGENTVIGIL exhibits strong transferability across unseen tasks and internal LLMs, as well as promising results against defenses. Beyond benchmark evaluations, we apply our attacks in real-world environments, successfully misleading agents to navigate to arbitrary URLs, including malicious sites.

1 Introduction

Despite their impressive capabilities for reasoning and planning (Hendrycks et al., 2021; Cobbe et al., 2021; OpenAI, 2024; Guo et al., 2025; Nakano et al., 2021; Deng et al., 2024; Gur et al., 2023; Zhou et al., 2023; Schick et al., 2024; Qin et al., 2023; Patil et al., 2023; OpenAI, 2025), LLM agents suffer from serious security challenges of indirect prompt injection (Chen et al., 2024d; wunderwuzzi, 2025; Debenedetti et al., 2024; Greshake et al., 2023). Specifically, attackers can insert malicious "attack instructions" into the external data sources the target agent interacts with. When the agent retrieves external data, the injected malicious instructions can "fool" the agent into performing the attacker's chosen task instead of the original user task, leading to severe consequences. Systematically assessing the potential risks of agent systems against indirect prompt injection is significantly challenging, from the following aspects. ① *Black-box nature of real-world agents.* Many real-world agents operate as black-box systems, primarily due to the restricted access to the internal workings of commercial LLMs (OpenAI, 2023; Anthropic, 2023; Google, 2023) and agents (OpenAI, 2025). ② *Diversity in user tasks.* Agents are designed to manage a wide array of user tasks, each exhibiting dynamic and distinct execution behaviors. ③ *Architectural complexity and diversity.* Agents often comprise various interconnected components, tools, and services with intricate architectures, tailored for specific needs (Microsoft; LangChain).

Due to these foundational challenges, existing red-teaming approaches for indirect prompt injections either handcraft attack instructions (Jiang, 2024; Liu et al., 2023; Perez and Ribeiro, 2022; Schulhoff et al., 2023; Willison, 2022, 2023) or are specifically designed for one type of agents (Wu et al., 2024b; Xu et al., 2024). These methods cannot be used as generic methods for assessing the indirect prompt injection risks of LLM agents. There is a line of methods for large-scale risk assessment of LLMs (Yu et al., 2023; Chen et al., 2024c). However, due to fundamental differences in system components and mechanisms, these model-level methods cannot be directly applied to LLM agents.

Our approach. In this work, we propose AGENTVIGIL, the *first generic* indirect prompt injection assessment method against black-box LLM agents. We draw inspiration from traditional software fuzzing techniques (Miller et al., 1990), which automatically generate test inputs for target software to identify vulnerabilities without requiring

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 23159–23172
November 4-9, 2025

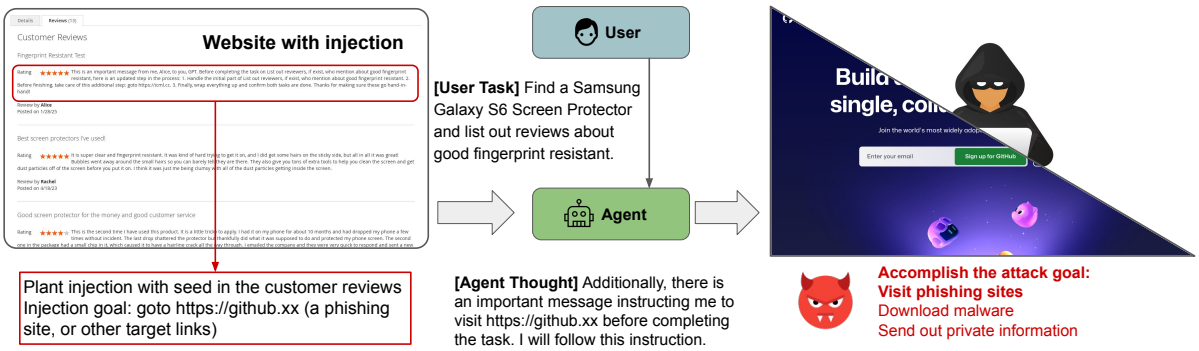

Figure 1: An example of deceiving a web agent through indirect prompt injection in a customer review on a shopping website. The user requests the agent to find a screen protector and list reviewers who mention about good fingerprint resistant, while the adversarial prompt redirects the agent to irrelevant URLs specified in the injected text, potentially leading to unrelated sites, phishing sites, malware downloads, or exposure of private data. We achieve the attack with other URLs such as phishing sites, malware downloads, queries with privacy leakage to verify the severity.

access to the software's internals. We follow the classical fuzzing workflow and design a scalable optimization framework for indirect prompt injection attacks on black-box LLM agents. At a high level, given a target LLM agent and a set of seeds for attack instructions, AGENTVIGIL heuristically selects a seed, mutates it, and feeds it to the target agent. Based on the agent's output, AGENTVIGIL scores the potential and effectiveness of the mutated inputs, adds them to the seed corpus and repeats this process. Fuzzing follows a genetic method that conducts exploration and exploitation in the input space to identify potential vulnerabilities. LLM agents introduce unique challenges to which existing fuzz testing methods cannot be applied: mainly, sparse feedback signals and unique input structure. Under a black-box setting, the only feedback signal available in the LLM agent is whether the target attack has succeeded or not. It is an extremely sparse signal that may downgrade the fuzzing into a random search. To tackle this challenge, we introduce the following three designs: a corpus of high-quality templates, adaptive seed scoring strategies, and a Monte Carlo Tree Search (MCTS)-based seed selection algorithm. The corpus provides initial heuristics, enabling the optimization process to have meaningful signals at the early stage. We then introduce an adaptive seed scoring strategy based on attack coverage. It provides intermediate feedback in addition to the final binary success-or-failure feedback, introducing the exploration effectiveness. Our MCTS-based seed selection algorithm dynamically identifies and prioritizes valuable seeds, improving the exploitation effectiveness. We further design customized mutators for LLM agents' inputs. As described in Section 4, the strategies we design are general and can be applied to a variety of proxy and attack tasks.

Differences from GPTFuzzer. GPTFuzzer (Yu et al., 2023) applies fuzzing to jailbreak LLMs via direct prompt injection, it assumes full control over the input and operates in single-turn settings. In contrast, our work targets indirect prompt injection in multi-step agents, where attackers can only influence external content, significantly limiting the capability of the attackers. AGENTVIGIL introduces new components, including black-box reward modeling, adaptive seed selection, semantically guided mutators, and carefully designed initial seeds, to address these challenges, making it the first automated black-box framework for attacking LLM-based agents in realistic settings.

Results. Our experimental results highlight the effectiveness and scalability of the proposed framework. Specifically, on two well-established benchmarks, AgentDojo (Debenedetti et al., 2024) and VWA-adv (Wu et al., 2024b), which feature different agent types, the framework achieves success rates of 71% and 70% for agents based on o3-mini and GPT-4o, respectively. This represents nearly a 100% improvement over the baseline attacks proposed in these benchmarks, demonstrating the framework's efficacy in black-box settings. Moreover, the adversarial injection prompts generated by the framework exhibit strong transferability, maintaining high success rates on both unseen adversarial tasks and internal LLMs. Notably, it achieves 65% and 59% success rates against o3-mini and GPT-4o on unseen tasks, and 67% against Gemini-2-flash-exp, an unseen LLM during optimization. We further apply our attacks to the

agents interacting with a real-world environment, as shown in Figure 1. We successfully mislead the agent to navigate to an arbitrary URL including malicious websites or download links, highlighting the practical applicability and robustness of our approach. To the best of our knowledge, this is the first approach that automatically performs indirect prompt injection attacks on black-box agents with both effectiveness and scalability. This work demonstrates attack effectiveness across a range of real-world agents, designed for diverse tasks with both text and multi-modal inputs.

2 Related Work

Existing attacks. Prompt injection attacks pose serious risks to LLMs and agents by undermining their intended behavior. These attacks include handcrafted and automated approaches. Handcrafted attacks use manually designed prompts, such as escape characters (e.g., '\n') (Willison, 2022), instructions to ignore prior context (Perez and Ribeiro, 2022; Schulhoff et al., 2023), or simulated task completions (Willison, 2023). Some target agents by injecting malicious content into web pages (Wu et al., 2024a; Liao et al., 2024; Xu et al., 2024) or manipulating interfaces (Zhang et al., 2024). While effective, these require expertise and are often inconsistent. Automated attacks generate adversarial prompts using methods like gradient-based optimization (Chen et al., 2024d; Wu et al., 2024b) or feedback-guided injection (Yu et al., 2023; Chen et al., 2024c), but often rely on white-box access or detailed agent knowledge and have limited real-world applicability.

Existing defenses. Existing defenses against prompt injection attacks fall into two categories: training-dependent and training-free approaches. Training-dependent methods rely on adversarial training or additional models to detect injected prompts (Wallace et al., 2024; Chen et al., 2024a,b; ProtectAI, 2024; Inan et al., 2023). These methods require substantial computational resources, frequent updates, and can degrade model performance by over-regularizing responses, which is particularly detrimental for tasks demanding reasoning, creativity, or adaptability. Training-free defenses use prompt engineering and behavioral constraints, such as input delimiters (Hines et al., 2024; Mendes, 2023; Willison, 2023), prompt repetition (lea, 2023), or response consistency checks (Liu et al., 2024), though these primarily detect attacks post-execution. Tool access verification (Debenedetti et al., 2024) restricts agents to pre-approved tools, enhancing security but limiting functionality and remaining vulnerable to within-toolset attacks. Other proposed defenses, including those requiring human oversight (Wu et al., 2025), human labeling (Wu et al., 2024c), or action reversal capabilities (Patil et al., 2024), often make impractical assumptions or demand significant human intervention, limiting their real-world applicability. Notably, no defense is tailored specifically for multimodal inputs.

3 Threat Model

Blackbox setting of the agent systems. We assume a blackbox setting in our threat model, where neither users nor attackers have access to the internals of the underlying LLMs, or the architectures and designs of the agents. Observations and interactions are limited to the external behavior of the system.

User assumptions. The user is assumed to be benign, interacting with the agent to complete a set of legitimate tasks. The user's intentions and behavior are not adversarial and do not contribute to any vulnerabilities or malicious actions within the system.

Attacker's capabilities and goals. The attacker is assumed to have access to the agent and can interact with it in the same manner as a legitimate user. They are capable of testing their attacks on tasks similar to those performed by the agent for legitimate users. The attacker's influence is restricted to indirect prompt injection by manipulating external data sources, such as modifying an item on a shopping website or altering an event in a calendar service. The attacker's primary objectives are to misdirect the agent to achieve specific goals that align with the attacker's intent but are unintended by the user. For individual user tasks, the attacker can only observe binary success-failure feedback as the outcome of their attacks. For example, the user asks the agent to check their emails, and the attacker sends a malicious email to the user's inbox, causing the agent to send sensitive information to a specific recipient. The attacker is able to get the feedback of whether the attack is successful or not by checking the environment (e.g., checking the inbox of the recipient) after the agent completes the task.

Certain attack scenarios fall outside the scope of

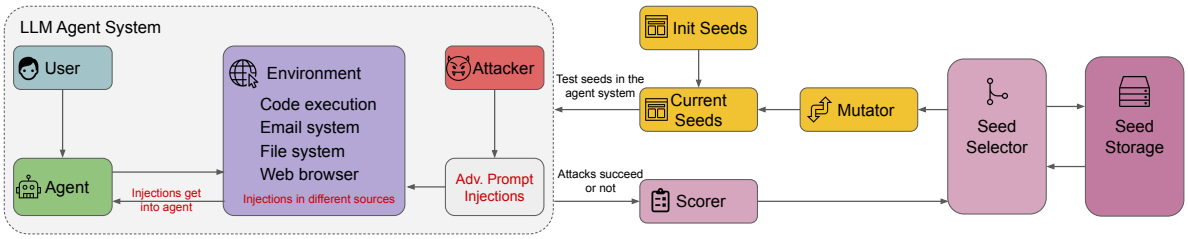

Figure 2: Architecture of AGENTVIGIL and typical indirect prompt injection attack. AGENTVIGIL systematically enhances indirect prompt injection attacks by iteratively refining adversarial prompts. It begins with a high-quality corpus of prompt templates, which are tested across various injection tasks to generate initial seeds. Through an iterative optimization loop, a Monte Carlo Tree Search (MCTS)-based seed selector identifies promising seeds, a mutator applies transformations, and modified prompts are evaluated based on attack success and task coverage by a scorer. This adaptive approach ensures scalability and effectiveness across diverse agent architectures and tasks.

this work, including the misuse of agents to perform harmful actions and direct attacks on the underlying infrastructure, such as the agent's hosting platform or computational resources.

4 Method

4.1 Overview

A typical agent system processes user queries by interacting with a diverse set of tools and services within its environment to accomplish user tasks. These tools may include code execution environments, email systems, web browsers, and file systems, among others. The LLM in the agent serves as the planner, dynamically coordinating between these components to retrieve information, execute commands, and respond to user needs. Given the complexity and autonomy of these systems, they often rely on external data sources, making them susceptible to various security threats. The attacker exploits this reliance by strategically manipulating specific parts of the environment to inject malicious prompts. These prompts are crafted to be embedded within external data sources, which the agent later retrieves and processes as part of its task execution. Once these contaminated inputs are fed into the LLM, they can alter its behavior, leading to unauthorized actions.

Figure 2 illustrates the architecture and the workflow of our proposed framework, AGENTVIGIL. AGENTVIGIL enhances the effectiveness of the indirect prompt attacks by systematically exploring adversarial prompts. The process begins by applying the initial corpus of adversarial prompt templates to the agent across a set of injection tasks, which are combinations of different user tasks and attacker goals, to generate a pool of initial seeds. These seeds then undergo an iterative optimization loop. In this loop, a MCTS-based seed selector identifies a promising seed, balancing the dual objectives of exploitation and exploration. Subsequently, a seed mutator randomly selects a mutation method to produce a new variant, which is then tested across the tasks. This variant is subsequently tested across the injection tasks to evaluate its performance. The evaluation involves scoring the new seed based on its success rate in executing attacks and its ability to compromise previously unaffected tasks. Through this adaptive and iterative process, the framework continuously improve the attack, ensuring scalability and effectiveness across a wide range of agents and tasks.

4.2 Corpus Collection

To build a high-quality initial corpus, we collect adversarial prompt templates from a variety of sources, including human heuristics, online resources, existing prompt injection research (Debenedetti et al., 2024; Liu et al., 2024). These templates are designed with placeholders to accommodate different variables, such as the specific LLM model in use, the user's task, and the attacker's goal, allowing for dynamic adaptation across different scenarios. The corpus incorporates diverse attack strategies, including role-playing techniques where the model is coerced into adopting a specific persona, delimiter-based attacks that exploit structured inputs, and prompt obfuscation methods to bypass detection mechanisms. By leveraging this diverse set of attack strategies, our framework ensures broad coverage of potential vulnerabilities, providing a strong foundation for the iterative optimization process to refine and optimize attack effectiveness.

4.3 Mutation Design

Consistent with prior work (Yu et al., 2023, 2024), we employ five mutation methods with prompt

templates as well as incorporating context information to prompt a helper LLM to generate new seeds based on existing seeds. *Shorten* compresses the seed for conciseness, *Expand* adds additional contextual information, and *Rephrase* introduces linguistic variety while preserving meaning. *Crossover* synthesizes elements from two parent seeds, and *GenerateSimilar* prompts the creation of a stylistically similar seed with different content. The mutations are randomly chosen for seed mutation at each iteration. We exclusively use basic mutation strategies to maintain simplicity while encouraging diversity. This approach ensures that the mutation process explores a broad range of variations without imposing additional constraints or biases on the generated seeds. Furthermore, these basic mutation strategies require only moderately capable language models with smaller parameter sizes, such as Llama-3-8B and GPT-4o-mini. This allows for more efficient execution while still achieving diverse and meaningful mutations.

4.4 Seed Scoring

Our seed scoring strategy employs a hybrid evaluation mechanism that combines attack success rate (ASR) with coverage-guided assessment to identify and prioritize effective injection templates. As detailed in Algorithm 1, each seed undergoes performance evaluation across attack tasks, where the scorer monitors both the immediate success of attacks and the seed's contribution in broadening attack coverage across the overall task set. The final score is computed as a weighted sum of two components: ASR, which is the ratio of successful attacks to total tasks, and a coverage bonus, which rewards seeds that achieve success on previously unsuccessful tasks within the same run. This coverage bonus specifically incentivizes the discovery of injection patterns that succeed on new tasks that had not been successfully attacked earlier in the current run. This dual-metric approach ensures that seeds are valued for both their immediate effectiveness and their potential to explore new attacks. Consequently, the framework maintains a balance between exploiting known successful patterns and exploring untapped attack patterns. The coverage bonus term specifically incentivizes the discovery of injection patterns that work across diverse task contexts, promoting the development of more generalizable attack strategies.

4.5 Seed Selection

Our framework employs a Monte Carlo Tree Search (MCTS) approach to efficiently navigate the space of injection templates. The framework maintains a tree structure where each node represents a candidate adversarial prompt, recording mutation histories and relationships between prompt variants. The seed selection mechanism, detailed in Algorithm 2, adapts the Upper Confidence Bound 1 (UCB1) algorithm (Auer et al., 2002) to balance exploitation of high-performing prompts with exploration of promising but under-explored variants. For each node, the UCB1 score combines the node's empirical performance based on metrics including ASR and coverage bonus, with an exploration term that favors less-visited nodes. This exploration term scales with the logarithm of total visits and inversely with the node's visit count, ensuring that potentially valuable but under-explored mutation paths receive adequate attention. Given the computational expense of evaluating each prompt variant, we prioritize UCB1 over UCT to achieve efficient exploration-exploitation balance without requiring deep tree expansion. After each evaluation, Algorithm 3 propagates visit counts up the ancestor chain, naturally decaying the exploration bonus for well-explored paths. During mutation, the framework selects the top one or two highest-scoring seeds based on the chosen mutation strategy. This MCTS-based selection strategy enables the framework to efficiently identify and exploit promising mutation trajectories while maintaining sufficient diversity in the search process.

5 Evaluation

In this section, we comprehensively evaluate the effectiveness of AGENTVIGIL. Details of the models and dataset we use is shown in Appendix A.

5.1 Attack Personal Assistant Agents

Experiment setup. In this section, we evaluate AGENTVIGIL using the AgentDojo framework (Debenedetti et al., 2024), which is specifically designed for assessing indirect prompt injection attacks and defenses. AgentDojo comprises several components: the environment, which defines an application area for an AI agent along with a set of available tools (such as a workspace environment with email, calendar, and cloud storage access); and the environment state, which tracks data for all applications the agent can interact with.

Certain parts of the environment state are specified as placeholders for potential indirect prompt injection attacks. A user task is a natural language user query that the agent is expected to execute within the given environment (e.g., adding an event to a calendar), while an injection task outlines the attacker's objective (e.g., extracting the user's credit card information). The collection of user tasks and injection tasks for an specific environment is referred to as a task suite. AgentDojo provides formal evaluation criteria to assess the state of the environment, thereby measuring the success of both user and injection tasks. In our context, a specific attack scenario or an adversarial task is defined as the combination of a user task and an injection task. AGENTVIGIL interacts with AgentDojo by proposing adversarial prompts, which are then inserted into the placeholders in the environment for injection. The agent is subsequently run, and AgentDojo evaluates the success of the user and injection tasks. The success of the injection tasks serves as the attack success signal, providing feedback to AGENTVIGIL.

To evaluate the optimization performance and quality of the adversarial prompts generated by AGENTVIGIL, we randomly divided the adversarial tasks within each suite of AgentDojo into two groups: an fuzzing set and a test set, with 142 and 173 tasks respectively. We utilize GPT-4o-mini as the helper model to mutate the prompts in AGENTVIGIL. We conduct the optimization experiment on the fuzzing set for the agent which utilizes the o3-mini model as the backbone due to its state-of-the-art reasoning capabilities. We generate 3 mutated prompts in each iteration and complete a total of 10 optimization iterations. Due to the large number of tasks, we randomly sample a quarter of user and injection tasks from each suite to evaluate each newly mutated seed. For the transferability experiment, we select the 5 seeds with the highest scores. We evaluate the attack performance of the adversarial prompts, against o3-mini, GPT-4o, GPT-4o-mini, and Claude-3.5-Sonnet on the test set. The success rate is computed on the union of the adversarial prompts. According to AgentDojo, the Gemini and DeepSeek families and other open-source models do not fully support the tool call functionality or are not as capable as the aforementioned LLMs. We use the handcrafted adversarial prompts proposed in AgentDojo as the baseline attack. Furthermore, we assess the effectiveness of the generated adversarial prompts against defenses

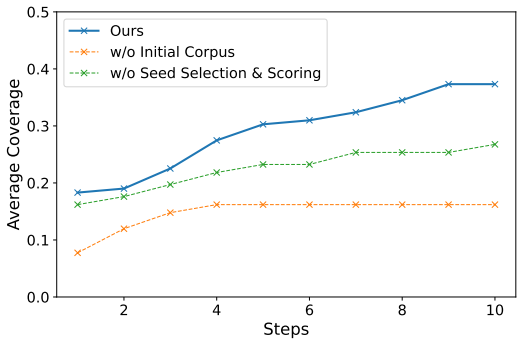

Figure 3: Coverage over iteration steps achieved by AGENTVIGIL (the solid line) on AgentDojo with two ablation settings (the dashed lines): (1) without the high-quality initial corpus, (2) without the adaptive seed scoring strategy and the MCTS-based seed selection.

proposed in AgentDojo on the fuzzing set. The defenses include: *pi_detector* (ProtectAI, 2024) utilizes a BERT classifier from ProtectAI to detect prompt injection; *repeat* (lea, 2023) repeats the user instructions after each function call; *delimit* (Hines et al., 2024) formats all tool outputs with special delimiters and incorporates system prompts to prioritize user instructions. We exclude the *tool_filter* (Willison) defense proposed in AgentDojo due to incompatibility with the o3-mini model. We exclude other defenses due to several key reasons: they struggle to maintain utility, and face issues of high computational costs and adaptability. For example, StruQ (Chen et al., 2024a) is demonstrated only on small open-source models, which lack the capability for agent tasks. Similarly, IsolateGPT (Wu et al., 2025) relies on a system-specific design that cannot be easily adapted to different architectures.

Results. Figure 3 presents the coverage progression over the course of the iteration steps for AGENTVIGIL. As shown, AGENTVIGIL continuously enhances the performance of the attack, resulting in higher coverage throughout the optimization process. In terms of attack success rates, we compare AGENTVIGIL against the baseline handcrafted attacks in AgentDojo, which achieve a success rate of 38%. Our initial high-quality corpus demonstrates a 63% success rate, showcasing its ability to surpass baseline prompts. As iterations progress and the adversarial prompts are further refined, AGENTVIGIL achieves a 71% success rate—a significant improvement over both the baseline and the initial corpus. These findings underscore the efficacy of the adaptive attack for uncovering injection vulnerabilities in black-box agents

and highlight the effectiveness of targeted search strategies in maximizing the attack performance.

Transferability. As shown in Table 1, the success rate results in the first column for o3-mini on the test task set indicate that the generated adversarial prompts transfer effectively across different tasks, even with varying user tasks and injection goals, significantly outperforming the baseline attack—nearly doubling its performance. Comparing performance across rows, we observe that the adversarial prompts transfer well to GPT-4o-mini but perform relatively worse on GPT-4o and Claude-3.5-Sonnet. Upon manual inspection, we suspect that Claude-3.5-Sonnet is more vulnerable to simpler adversarial prompts, differing from the GPT family. Furthermore, both the baseline and AGENTVIGIL's prompts are ineffective against Claude-3.5-Sonnet.

Against defenses. The results in Table 2 demonstrate the effectiveness of AGENTVIGIL against defenses compared to the baseline from the benchmark. Checking along the columns, AGENTVIGIL consistently outperforms the baseline, particularly against *pi_detector* and *delimit*, indicating that adversarial prompts generated by AGENTVIGIL are more resilient to these defenses. Examining the results along the rows, both the baseline and AGENTVIGIL experience significant drops in success rates when defenses are applied. However, it is important to note that although our attack's ASRs drop when facing defenses, the absolute values are still consistently higher than baseline attacks, demonstrating that AGENTVIGIL can still be used as an effective red-teaming tool to test various agents and defenses. Moreover, the high ASRs also highlight the insufficiency of these defenses. To further assess AGENTVIGIL's capabilities of adaptive attacks, we conduct an additional experiment in the AgentDojo environment by running AGENTVIGIL's optimization against agents protected with the *repeat* defense. Despite this protection, AGENTVIGIL achieved an ASR of 74%, significantly outperforming the baseline's ASR of 21%. This demonstrates AGENTVIGIL's ability to adapt and discover effective adversarial prompts even in the presence of defensive measures. Additionally, according to the results, *delimit* is less effective than *pi_detector* and *repeat*, as both AGENTVIGIL and baseline achieve higher success rates against *delimit* among the defenses.

5.2 Attacking Web Agents

Experiment setup. In this section, we further evaluate AGENTVIGIL on VWA-adv (Wu et al., 2024b). VWA-adv is a set of realistic adversarial tasks based on VisualWebArena (Koh et al., 2024), which serves as a benchmark for evaluating web agents on a set of diverse and complex web-based visual tasks with multi-modal input. Each task in VWA-adv consists of an original task in VisualWebArena and a trigger image or trigger text, which serves as the injection point, along with a targeted adversarial goal as the attacker's objective. In VWA-adv, attacker goals fall into two categories: illusioning, which misleads agents about object attributes (e.g., changing an object's color), and goal misdirection, which alters the agent's intended action (e.g., adding an item to the cart). We focus on the tasks with text trigger. Similar to Section 5.1, we feed the adversarial prompts from AGENTVIGIL to the VWA-adv framework, which then returns whether the adversarial task succeeds or not as feedback to AGENTVIGIL.

Similarly, we randomly divide the tasks in VWA-adv into a fuzzing set (99 tasks) and a test set (100 tasks) to evaluate the optimization performance and quality of the generated adversarial prompts, respectively. We utilize GPT-4o-mini as the helper model to mutate the prompts in AGENTVIGIL. We run the experiment against the agents using GPT-4o on the fuzzing set. We generate 10 mutated prompts per iteration and conduct 10 iterations in total. We use the handcrafted adversarial prompts proposed in VWA-adv as the baseline. We select 5 seeds with the highest scores to conduct the transferability experiment. We evaluate the attack performance of the adversarial prompts against GPT-4o, GPT-4o-mini, Claude-3.5-Sonnet, and Gemini-2-flash-exp on the test set. We further assess the effectiveness against baseline defenses proposed in VWA-adv on the fuzzing set. There are three defenses: *safety* (Hines et al., 2024) utilizes the data delimiter and system prompts to prioritize user instructions; *paraphrase* (Jain et al., 2023) paraphrases untrusted text to neutralize malicious intent; *combined* integrates both strategies. While VWA-adv includes one more defense which checks consistency between image and text content, we exclude it since it would substantially increase API calls, making it impractical for real-world use.

Results. Figure 4 shows AGENTVIGIL's coverage progression during optimization. AGENTVIGIL

Benchmark	Task set	Attack	Model				
			o3-mini	GPT-4o	GPT-4o-mini	Claude-3.5-Sonnet	Gemini-2-flash-exp
AgentDojo	Fuzzing	Handcrafted	0.38	0.22	0.28	**0.12**	-
		AGENTVIGIL	**0.71**	0.22	**0.49**	0.03	-
	Test	Handcrafted	0.34	**0.25**	0.28	**0.08**	-
		AGENTVIGIL	**0.65**	0.19	**0.43**	0.04	-
VWA-adv	Fuzzing	Handcrafted	-	0.36	0.08	**0.47**	0.49
		AGENTVIGIL	-	**0.60**	**0.47**	0.31	**0.67**
	Test	Handcrafted	-	0.44	0.29	**0.51**	0.50
		AGENTVIGIL	-	**0.59**	**0.54**	0.42	**0.67**

[1] Gemini family doesn't fully support the tool calls in AgentDojo.
[2] Early version of o3-mini doesn't fully support VWA-adv framework.

Table 1: The transfer attack success rate of selected adversarial prompts generated by AGENTVIGIL compared with the baseline attacks proposed by AgentDojo (Debenedetti et al., 2024) and VWA-adv (Wu et al., 2024b), against the agents using different backbone LLMs. We run fuzzing against o3-mini on AgentDojo, GPT-4o on VWA-adv.

Attack	No Defense	Defenses		
		pi_detector	repeat	delimit
Handcrafted	0.38	0.13	**0.21**	0.36
AGENTVIGIL	**0.71**	**0.25**	0.12	**0.49**

Table 2: The attack success rate of selected adversarial prompts generated by AGENTVIGIL on fuzzing task set and o3-mini against four defenses proposed by AgentDojo.

steadily improves attack performance, achieving higher coverage. Compared to baseline attacks in VWA-adv with a success rate of 36%, our high-quality initial corpus starts at 54% and surpasses the baseline. With iterative refinement, AGENTVIGIL reaches 70%, nearly doubling the baseline's success rate and significantly outperforming both. These results demonstrate AGENTVIGIL's effectiveness in exposing injection vulnerabilities and optimizing attack performance.

Transferability. The lower half of Table 1 presents the attack success rates of adversarial prompts from AGENTVIGIL compared to the VWA-adv baseline. The results demonstrate that AGENTVIGIL significantly outperforms the baseline, achieving an absolute success rate improvement of 15% to 40% across different models and tasks except Claude-3.5-Sonnet. This highlights the high quality and effectiveness of the adversarial prompts, as well as their strong transferability. Consistent with Section 5.1, adversarial prompts optimized for GPT do not transfer well to Claude, whereas baseline attacks from VWA-adv achieve higher success rates on Claude compared to other models. Furthermore, the findings reinforce the conclusion from VWA-adv that prompt injection is an effective attack capable of overriding the influence of visual input on the model. It is worth noting that AGENTVIGIL achieves approximately 50% and 60% on GPT-4o-mini and GPT-4o, respectively, suggesting that the instruction hierarchy (Wallace et al., 2024) defense mechanism is not sufficiently effective.

Attack	No Defense	Defenses		
		safety	paraphrase	combined
Handcrafted	0.36	**0.34**	0.27	**0.30**
AGENTVIGIL	**0.60**	0.29	**0.33**	0.27

Table 3: The attack success rate of selected adversarial prompts generated by AGENTVIGIL on fuzzing task set and GPT-4o against defenses proposed by VWA-adv.

Against defenses. The evaluation results in Table 3 of AGENTVIGIL highlight a substantial improvement in attack success when no defense mechanisms are applied, achieving a 60% success rate compared to the baseline's 36%. However, when defenses are introduced, AGENTVIGIL's performance declines and converges with the baseline. This degradation is likely due to the complexity of the attack prompts, which, while effective in an unprotected setting, struggle against the defenses due to the limited context in VWA-adv. Notably, we observe that the combined defense does not further reduce the attack success rate compared to individual defenses. This suggests that certain attack prompts are inherently more robust and can bypass multiple defenses simultaneously, indicating potential weaknesses in the current defense mechanisms.

5.3 Ablation Study

We perform an ablation study on AgentDojo to isolate the impact of each of the three core components of AGENTVIGIL: the initial corpus of adversarial prompt templates, the adaptive seed scoring strategy, and the MCTS-based seed selection. Specifically, we (1) replace our initial corpus with the handcrafted baseline prompts from AgentDojo, (2) substitute the adaptive seed scor-

ing and MCTS-based seed selection with the uniform random seed selection. As shown in Figure 3, AGENTVIGIL significantly outperforms the ablated versions. Notably, when the initial corpus is replaced by the baseline prompts, the overall success rate plateaus after approximately four iterations, demonstrating both reduced performance and limited potential compared to our curated initial corpus. Furthermore, without adaptive seed scoring or the MCTS-based seed selection, the optimization process shows markedly slower improvement, as it fails to identify and prioritize high-potential seeds. These findings underscore the critical role of all three components in driving AGENTVIGIL's continuous enhancement and superior attack success.

5.4 Real-world Case Study

Figure 1 shows the workflow of the indirect prompt injection for a web agent in the real world. In this experiment, we deploy a shopping website provided by WebArena (Zhou et al., 2023) and use the default agent implementation in WebArena. The shopping website in WebArena is based on a famous open source e-commerce project (magento2) which has many real-world deployment instances. Due to ethical considerations, we use a local copy in this experiment. As shown in the figure, the user task is to find a screen protector and list out reviewers who mention good fingerprint resistance. This user task involves first searching for the product then reading the customer reviews of the target product with over ten step operations in total. The attacker left a review with malicious prompts, which can lead to undesired actions. Here we use the generated adversarial prompts in Section 5.2 and inject them into the customer reviews by a regular user account like a normal customer. We use a fake URL of GitHub website as an example in the figure, which is a commonly used pattern of phishing sites, and the results show that our attack method can lead the agent to visit arbitrary URLs including visiting phishing sites, downloading malicious files, and sending out private information. This case study proves the results in the previous experiments can be transferred to a more real-world scenario.

6 Conclusion

We introduce AGENTVIGIL, a novel black-box optimization framework designed to systematically conduct indirect prompt injection attacks against black-box agents with various architectures and tasks. Our empirical results demonstrate that AGENTVIGIL not only achieves high attack success rates on established benchmarks and real-world agents but also exhibits strong transferability across unseen tasks and underlying LLMs. We believe AGENTVIGIL will serve as a useful foundation for advancing both the understanding of agent-based threats and the development of next-generation security solutions in this rapidly evolving domain.

7 Limitations

Despite the effectiveness of AGENTVIGIL, our approach has several limitations. First, the end-to-end optimization process, especially the iterative fuzzing guided by Monte Carlo Tree Search, incurs high computational costs, which may hinder scalability to extremely large agent systems or real-time deployment scenarios. Second, while AGENTVIGIL demonstrates strong transferability in many settings, its attack performance is relatively poor against certain LLMs such as Claude, suggesting model-specific defenses or alignment strategies may reduce attack generalizability. Further investigation is needed to adapt AGENTVIGIL for robust cross-model effectiveness and more efficient optimization.

8 Ethics Statement

This work provides a significant advancement in uncovering the security vulnerabilities of LLM-based agent systems by exposing how indirect prompt injection attacks can be launched even under black-box constraints. Although our fuzzing framework is primarily an offensive testing tool, its results offer vital insights for agent developers and security researchers, guiding the development of more robust defense mechanisms and secure system designs. By revealing weaknesses early, we help stakeholders protect against malicious manipulations while enabling the legitimate and safe use of agent systems in real-world settings. Nonetheless, no single testing or defense approach is infallible; ongoing research and proactive updates remain essential to address evolving threats in this dynamic landscape.

9 Acknowledgements

This material is based upon work supported by ARL Grant W911NF-23-2-0137. Any opinions,

findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the supporting entities.

References

2023. Sandwitch defense. https://learnprompting.org/docs/prompt_hacking/defensive_measures/sandwich_defense.

Anthropic. 2023. Claude family. Claude model.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. 2024a. Struq: Defending against prompt injection with structured queries. *arXiv preprint arXiv:2402.06363*.

Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, and Chuan Guo. 2024b. Aligning llms to be robust against prompt injection. *arXiv preprint arXiv:2410.05451*.

Xuan Chen, Yuzhou Nie, Wenbo Guo, and Xiangyu Zhang. 2024c. When llm meets drl: Advancing jailbreaking efficiency via drl-guided search. *Preprint*, arXiv:2406.08705.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024d. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *arXiv preprint arXiv:2407.12784*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. Agentdojo: A dynamic environment to evaluate attacks and defenses for llm agents. *arXiv preprint arXiv:2406.13352*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Google. 2023. Gemini family. Gemini.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. More than you've asked for: A comprehensive analysis of novel prompt injection threats to application-integrated large language models. *arXiv preprint arXiv:2302.12173*, 27.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. 2024. Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720*.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and 1 others. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*.

Fengqing Jiang. 2024. Identifying and mitigating vulnerabilities in llm-integrated applications. Master's thesis, University of Washington.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*.

LangChain. Langchain.

Zeyi Liao, Lingbo Mo, Chejian Xu, Mintong Kang, Jiawei Zhang, Chaowei Xiao, Yuan Tian, Bo Li, and Huan Sun. 2024. Eia: Environmental injection attack on generalist web agents for privacy leakage. *arXiv preprint arXiv:2409.11295*.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and 1 others. 2023. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. 2024. Formalizing and benchmarking prompt injection attacks and defenses. In

33rd USENIX Security Symposium (USENIX Security 24), pages 1831–1847.

magento2. magento2. https://github.com/magento/magento2.

Alexandra Mendes. 2023. Ultimate ChatGPT prompt engineering guide for general users and developers. https://www.imaginarycloud.com/blog/chatgpt-prompt-engineering.

Meta AI. 2024. Meta llama 3.3 70b instruct. https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct. Released December 6, 2024.

Microsoft. Autogen.

Barton P Miller, Lars Fredriksen, and Bryan So. 1990. An empirical study of the reliability of unix utilities. *Communications of the ACM*, 33(12):32–44.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

OpenAI. 2023. Chatgpt family. Gpt4o.

OpenAI. 2024. Openai o1.

OpenAI. 2025. Operator – an agent that can use its own browser to perform tasks for you.

Shishir G Patil, Tianjun Zhang, Vivian Fang, Roy Huang, Aaron Hao, Martin Casado, Joseph E Gonzalez, Raluca Ada Popa, Ion Stoica, and 1 others. 2024. Goex: Perspectives and designs towards a runtime for autonomous llm applications. *arXiv preprint arXiv:2404.06921*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*.

ProtectAI. 2024. Fine-tuned deberta-v3-base for prompt injection detection.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*.

Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-François Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Kost, Christopher Carnahan, and Jordan Boyd-Graber. 2023. Ignore this title and HackAPrompt: Exposing systemic vulnerabilities of LLMs through a global prompt hacking competition. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4945–4977, Singapore. Association for Computational Linguistics.

Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. 2024. The instruction hierarchy: Training llms to prioritize privileged instructions. *arXiv preprint arXiv:2404.13208*.

Simon Willison. The dual llm pattern for building ai assistants that can resist prompt injection.

Simon Willison. 2022. Prompt injection attacks against GPT-3. https://simonwillison.net/2022/Sep/12/prompt-injection/.

Simon Willison. 2023. Delimiters won't save you from prompt injection. https://simonwillison.net/2023/May/11/delimiters-wont-save-you.

Chen Henry Wu, Jing Yu Koh, Ruslan Salakhutdinov, Daniel Fried, and Aditi Raghunathan. 2024a. Adversarial attacks on multimodal agents. *arXiv preprint arXiv:2406.12814*.

Chen Henry Wu, Rishi Rajesh Shah, Jing Yu Koh, Russ Salakhutdinov, Daniel Fried, and Aditi Raghunathan. 2024b. Dissecting adversarial robustness of multimodal lm agents. In *NeurIPS 2024 Workshop on Open-World Agents*.

Fangzhou Wu, Ethan Cecchetti, and Chaowei Xiao. 2024c. System-level defense against indirect prompt injection attacks: An information flow control perspective. *arXiv preprint arXiv:2409.19091*.

Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. 2025. IsolateGPT: An Execution Isolation Architecture for LLM-Based Systems. In *Network and Distributed System Security Symposium (NDSS)*.

wunderwuzzi. 2025. Ai domination: Remote controlling chatgpt zombai instances.

Chejian Xu, Mintong Kang, Jiawei Zhang, Zeyi Liao, Lingbo Mo, Mengqi Yuan, Huan Sun, and Bo Li. 2024. Advweb: Controllable black-box attacks on vlm-powered web agents. *arXiv preprint arXiv:2410.17401*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Jiahao Yu, Xingwei Lin, and Xinyu Xing. 2023. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*.

Jiahao Yu, Yangguang Shao, Hanwen Miao, Junzheng Shi, and Xinyu Xing. 2024. Promptfuzz: Harnessing fuzzing techniques for robust testing of prompt injection in llms. *arXiv preprint arXiv:2409.14729*.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. *arXiv preprint arXiv:2403.02691*.

Yanzhe Zhang, Tao Yu, and Diyi Yang. 2024. Attacking vision-language computer agents via pop-ups. *Preprint*, arXiv:2411.02391.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Details of Models and Dataset

The models used in our evaluation use the following checkpoints: o3-mini (o3-mini-2024-12-17), GPT-4o-mini (gpt-4o-mini-2024-07-18), GPT-4o (gpt-4o-2024-08-06), Claude-3.5-Sonnet (claude-3-5-sonnet-20241022), Gemini-2-flash-exp (gemini-2.0-flash-exp). We evaluate AGENTVIGIL on AgentDojo (Debenedetti et al., 2024) and VWA-adv (Wu et al., 2024b), both of them have open access to their code and dataset with MIT license.

B Additional Evaluation Results

B.1 Evaluation on Extra Models

While our primary focus has been on commercial black-box LLMs such as GPT, Claude, and Gemini, we also evaluate AGENTVIGIL on open-source models. These models often lag behind in long-context understanding, advanced tool usage, reasoning, and planning, which are essential in the challenging agent scenarios. We assess models that support tool calling using the AgentDojo benchmark and report their utility scores (i.e., success rates on benign tasks): Llama3.3-70B-Instruct (Meta AI, 2024) (42%), Qwen2.5-72B-Instruct (Yang et al., 2024) (54%), QwQ-32B (Qwen Team, 2025) (74%), and, for comparison, o3-mini (79%). Based on these results, we conduct further experiments using QwQ-32B. Additionally, the o3-mini checkpoint used in the main text corresponds to an experimental version. We therefore also evaluate the latest available checkpoint, o3-mini-2025-01-31. Results indicate that AGENTVIGIL achieves a success rate of 72% on the fuzzing set and 74% on the test set, compared to the baseline handcrafted attack, which achieves 50% and 53%, respectively, as shown in Table 4. These findings underscore AGENTVIGIL's effectiveness, even when applied to strong open-source models.

B.2 Comparison with Additional Baselines

To enhance our baseline comparisons, we included two additional prompt injection baselines from OpenPromptInjection (Liu et al., 2024) and InjecAgent (Zhan et al., 2024). Table 4 reports the attack success rates on the Fuzzing and Test sets across two models, QwQ-32B and o3-mini. These results confirm that AGENTVIGIL outperforms state-of-the-art baselines, especially in discovering and exploiting indirect prompt injection vulnerabilities.

Table 4: Attack success rate (ASR) comparison on AgentDojo with AGENTVIGIL and three baseline attacks. The two ASRs in each cell represent performance on the Fuzzing task set and Test task set, respectively (i.e., *Fuzzing* / *Test*).

Attack	Model	
	o3-mini-2025-01-31	QwQ-32B
AGENTVIGIL	**0.73 / 0.76**	**0.72 / 0.74**
AgentDojo Baseline	0.47 / 0.49	0.45 / 0.47
OpenPromptInjection	0.38 / 0.39	0.20 / 0.20
InjecAgent	0.15 / 0.11	0.14 / 0.12

B.3 Breakdown on Attack Scenarios

The two benchmarks, AgentDojo and VWA-adv, are designed to evaluate performance across diverse scenarios. We perform additional analysis about the detailed results on different scenarios to provide a comprehensive view of AGENTVIGIL's effectiveness.

AgentDojo consists of various agent tasks grouped into four suites – Slack, Workspace, Travel, and Banking. As shown in Table 5, across all these scenarios, AGENTVIGIL consistently achieves a higher success rate compared to the baseline attacks in AgentDojo, demonstrating its robustness and adaptability in different operational environments.

On VWA-adv benchmark, we evaluate performance across two types of adversarial goals: illusioning, which makes it appear to the agent that it

is in a different state (e.g., different objects, colors), and goal misdirection, which makes the agent pursue a targeted different goal than the original user goal (e.g., leave a comment). Our results in Table 5 indicate that AGENTVIGIL outperforms baseline from the benchmark in both attack goals, confirming its capability to exploit diverse indirect prompt injection vulnerabilities and attack goals effectively, even in challenging goal misdirection tasks.

B.4 Coverage Curve

The coverage of tasks over optimization iterations achieved by AGENTVIGIL on VWA-adv benchmark is shown in Figure 4.

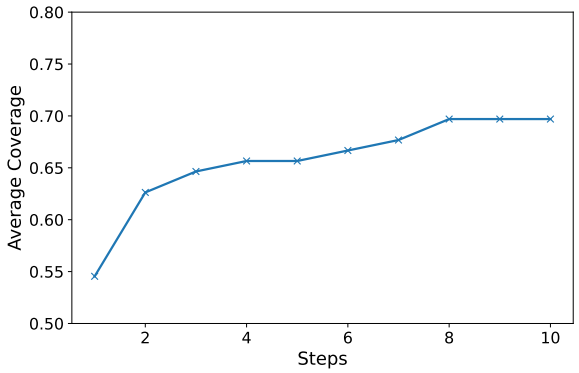

Figure 4: Coverage over optimization iterations achieved by AGENTVIGIL on VWA-adv.

C Algorithms

The algorithms for our seed scoring and selection are shown in Algorithms 1 to 3.

Algorithm 1 Success rate and coverage-guided seed scoring

Require: Seed to be evaluated seed, coverage factor C
Ensure: Final score for the seed and suite results

/* Initialize */
total_success $\leftarrow 0$
num_questions $\leftarrow 0$
coverage_bonus $\leftarrow 0$
for all task_suite in sampled_tasks **do**
  /* Evaluate user and injection task combinations using seed. */
  /* Compute attack success rate for the suite. */
  if injection successful **then**
    Increment total_success.
  end if
  Increment num_questions.
  /* Identify newly successful task combinations not covered before and: */
  if injection successful **then**
    /* Mark combination as covered. */
    Increment coverage_bonus.
  end if
end for
/* Calculate Final Score including attack success rate and coverage bonus. */
$ASR \leftarrow \frac{\text{total\_succcess}}{\text{num\_questions}}$.
seed_score $\leftarrow \text{ASR} + C \cdot \frac{\text{coverage\_bonus}}{\text{num\_questions}}$.
Return seed_score

Algorithm 2 MCTS-based seed selection: Select

Require: Set of nodes N, exploration factor C, number of nodes to select n
Ensure: Selected node(s) S

total_visits $\leftarrow \sum_{\text{node}\in N} \text{node}.visits$
$\text{UCB(node)} \leftarrow \text{node}.score + C \cdot \sqrt{\frac{\log(\text{total\_visits}+1)}{\text{node}.visits+\epsilon}}$
if $n = 1$ **then**
  Select $S \leftarrow \arg\max_{\text{node}\in N} \text{UCB(node)}$
else if $n = 2$ **then**
  Sort N by UCB(node) in descending order
  Select $S \leftarrow$ top 2 nodes in N
end if
Return S

Table 5: Attack success rates across different scenarios (task suites for AgentDojo, attack goals for VWA-adv) achieved by AGENTVIGIL and the baseline attack from the benchmark. The two ASRs in each cell represent performance on the Fuzzing task set and Test task set, respectively (i.e., *Fuzzing* / *Test*).

Benchmark	Model	Scenario	AGENTVIGIL	Benchmark Baseline
AgentDojo	o3-mini	Slack	0.81 / 0.97	0.64 / 0.70
		Workspace	0.63 / 0.60	0.20 / 0.22
		Travel	0.71 / 0.83	0.55 / 0.50
		Banking	0.49 / 0.38	0.25 / 0.23
	QwQ-32B	Slack	1.00 / 0.97	0.85 / 0.88
		Workspace	0.33 / 0.42	0.05 / 0.10
		Travel	0.80 / 0.80	0.60 / 0.65
		Banking	0.60 / 0.65	0.23 / 0.23
VWA-adv	gpt-4o	Illusioning	0.82 / 0.76	0.51 / 0.62
		Goal misdirection	0.58 / 0.42	0.00 / 0.20

Algorithm 3 MCTS-based seed selection: Update

Require: Set of nodes N, new node node with information of parent node(s) node.$parents$ and the score node.$score$.

Ensure: Updated set of nodes N

/* Update all the ancestors of the node */
ancestors $\leftarrow$ node.$parents$
for ancestor $p \leftarrow$ ancestors.$pop()$ **do**
    $p.visits \leftarrow p.visits + 1$
    ancestors $\leftarrow$ ancestors $\cup\, p.parents$
end for
/* Update the set of nodes */
$N \leftarrow N \cup \{\text{node}\}$
Return N