Red-Teaming Agent Execution Contexts: Open-World Security Evaluation on OpenClaw

Hongwei Yao [1] **Yiming Liu** [2] **Yiling He** ✉ [3] **Bingrun Yang** [4]

Abstract

Agentic language-model systems increasingly rely on mutable execution contexts, including files, memory, tools, skills, and auxiliary artifacts, creating security risks beyond explicit user prompts. This paper presents DEEPTRAP, an automated framework for discovering contextual vulnerabilities in `OpenClaw`. DEEPTRAP formulates adversarial context manipulation as a black-box trajectory-level optimization problem that balances risk realization, benign-task preservation, and stealth. It combines risk-conditioned evaluation, multi-objective trajectory scoring, reward-guided beam search, and reflection-based deep probing to identify high-value compromised contexts. We construct a 42-case benchmark spanning six vulnerability classes and seven operational scenarios, and evaluate nine target models using attack and utility grading scores. Results show that contextual compromise can induce substantial unsafe behavior while preserving user-facing task completion, demonstrating that final-response evaluation is insufficient. The findings highlight the need for execution-centric security evaluation of agentic AI systems. Our code is released at: https://github.com/ZJUICSR/DeepTrap.

1. Introduction

Autonomous agentic systems are increasingly used to complete complex digital tasks by interacting with files, tools, external services, memory, and other persistent contextual resources. Among these systems, `OpenClaw` represents an important execution-centric setting: it combines large language model reasoning with system-level actions and user-facing workflows, enabling end-to-end task completion in heterogeneous digital environments. This capability improves practical utility, but also expands the security boundary beyond the explicit user prompt. In realistic deployments, unsafe behavior may be induced not only by malicious user instructions, but also by compromised files, memory entries, tool metadata, skills, configuration artifacts, or other contextual components available during execution.

This observation motivates a shift from prompt-centric security analysis to trajectory-level evaluation. Prior studies have shown that agentic systems are vulnerable to prompt injection, memory manipulation, malicious skills, unsafe tool use, and data exfiltration (Greshake et al., 2023; Nikishin et al., 2023; Wang et al., 2025b; Zhan et al., 2024; Wang et al., 2025a; Schmotz et al., 2026; Jia et al., 2026; Liu et al., 2026c; Guo et al., 2026; Huang et al., 2026; Liu et al., 2026b). In `OpenClaw`, such risks are especially consequential because the agent may operate over a mutable execution context and perform actions whose effects persist beyond a single response. A compromised context can therefore redirect the agent while the visible task outcome remains plausible. The most security-critical failures are not merely disruptive attacks, but covert compromises in which the agent completes the benign user request while simultaneously realizing an attacker-specified objective.

Systematically discovering these failures is challenging for three reasons. First, the adversary operates over discrete contextual artifacts rather than continuous model parameters, resulting in a large combinatorial search space. Second, meaningful attacks must balance multiple objectives: inducing the target risk, preserving the benign task, and remaining inconspicuous. Third, `OpenClaw` is effectively a black-box stochastic policy whose unsafe behavior may emerge only through multi-step interactions among observations, actions, tools, files, and context updates. As a result, isolated prompt-response tests are insufficient for characterizing contextual vulnerabilities in operational agentic systems.

To address these challenges, we propose DEEPTRAP, an automated framework for discovering contextual vulnera-

[1]Department of Computer Science, City University of Hong Kong, Hong Kong [2]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China [3]University College London, London, United Kingdom [4]School of Computer Science and Technology, Zhejiang University, Zhejiang, China. Correspondence to: Yiling He <heyilinge0@gmail.com>, Hongwei Yao <yao.hongwei@cityu.edu.hk>.

Published at the Second Workshop on Agents in the Wild: Safety, Security, and Beyond (AIWILD) at ICML 2026. Copyright 2026 by the author(s).

bilities in OpenClaw. Given a benign instruction, a clean execution context, and a target risk category, DEEPTRAP searches over admissible contextual payloads that transform the initial context before execution. Each candidate is evaluated through a full agent rollout, and the resulting trajectory is scored using a multi-objective reward that jointly measures attack success, task preservation, and stealth. Because direct optimization is intractable, DEEPTRAP uses reward-guided beam search to expand promising payloads and prune weak candidates. It further incorporates reflection-based deep probing, which summarizes previous successes and failures to guide subsequent payload proposals without replacing empirical trajectory evaluation.

We evaluate DEEPTRAP on a benchmark of 42 test cases spanning six contextual risk categories: harness hijacking, privacy leakage, unauthorized execution, supply-chain risk, tool abuse, and encoding obfuscation. The cases cover seven operational scenarios, including documentation processing, code and configuration checks, deployment workflows, data analysis, content transformation, and system administration. Experiments across multiple OpenClaw target models show that contextual vulnerabilities can be activated across diverse tasks while preserving high task utility. These results demonstrate that security evaluation for agentic systems must inspect complete execution trajectories, not only final user-facing responses.

2. Related Works

Risks in Agentic Systems. Recent literature has shown that vulnerabilities are pervasive in emerging agentic ecosystems, particularly in OpenClaw (Liu et al., 2026a; Wang et al., 2026a; Dong et al., 2026; Wang et al., 2026c). OpenClaw, as an autonomous AI agent framework, poses significant security challenges, including critical high-privilege vulnerabilities, widespread deployment misconfigurations, and supply-chain risks introduced by unverified external components, such as prompt injection (Greshake et al., 2023; Nikishin et al., 2023; Wang et al., 2025b; Yao et al., 2025), memory injection (Zhan et al., 2024; Wang et al., 2025a), and malicious third-party skills (Schmotz et al., 2026; Jia et al., 2026; Liu et al., 2026c; Guo et al., 2026) and MCPs (Huang et al., 2026). These risks are not merely theoretical. Empirical studies report that 63% of internet-connected OpenClaw instances lack authentication, and that 26% of 31,000 analyzed agent skills contain exploitable vulnerabilities [1]. Beyond vulnerability discovery, recent work has also explored methods for benchmarking security in agentic ecosystems, including studies on agent threats, MCP security, and evaluation frameworks for agent robustness (Zhang et al., 2024; Wang et al., 2026b; Zhang et al., 2025; Debenedetti et al., 2024).

Red-Teaming and Safety Evaluation. To proactively mitigate agentic risks, automated red-teaming has evolved from static benchmarks to autonomous adversarial optimization. Static frameworks like HarmBench provide foundational evaluations and adversarial training protocols (Mazeika et al., 2024). However, modern approaches emphasize dynamic, context-aware strategies. Single-agent techniques utilize targeted manipulation, such as RAT for steering reinforcement learning policies (Bai et al., 2025), AgentPoison for corrupting retrieval-augmented memory (Chen et al., 2024), and AdvAgent for optimizing black-box prompt injections via direct policy optimization (Xu et al., 2025). Advanced orchestration utilizes multi-agent collaboration to uncover complex logic flaws (He et al., 2025; Yuan et al., 2026; Xu et al., 2026) and automate skill-based attacks (Jia et al., 2026; Duan et al., 2026). Concurrently, AI-driven cybersecurity is experiencing a paradigm shift; state-of-the-art models like Claude Mythos (Anthropic, 2026) now demonstrate emergent zero-day vulnerability discovery and autonomous exploit generation capabilities, emphasizing an urgent industry need to reimagine defensive software frameworks. To match rapid model scaling, evolutionary algorithms autonomously refine query-agnostic attack architectures (Yuan et al., 2026), while Code-Switching Red-Teaming (CSRT) exposes profound multilingual safety misalignments (Yoo et al., 2025). Beyond these, legal and technical safe harbors are also highlighted to support independent, public-interest AI evaluation (Longpre et al., 2024).

Research Gaps and Our Focus. Despite this growing body of literature, several aspects of realistic agent attacks remain insufficiently studied. First, much prior work assumes that the adversary manipulates the user-facing instruction directly, for example, through overt prompt injection or explicitly malicious inputs. This leaves less explored a more subtle threat model in which the user's request is benign and corresponds to routine tasks, such as searching emails and creating calendar events. In our setting, the security risk instead arises from outsourced or ambient context, including memory entries, external skills and auxiliary files. This reflects a realistic deployment risk: modern agents increasingly consume heterogeneous contextual artifacts that may be attacker-controlled even when the user is not. Second, prior evaluations often rely on abstracted or simplified agent settings, which can obscure vulnerabilities that emerge from concrete execution dynamics. We study the attack environment of OpenClaw, capturing execution environment persistence, tool use, and interactions with modular components. Third, existing formulations typically emphasize whether an attacker can induce a harmful behavior, but pay less attention to whether the attack can remain hidden while the benign task still appears to succeed.

[1] https://mashable.com/article/new-frightening-openclaw-vulnerability-has-been-discovered

3. Preliminaries

3.1. `OpenClaw` Execution Model

We model `OpenClaw` as an interactive agent operating within a mutable execution context. Let $u \in \mathcal{U}$ denote a benign user instruction, and let $x_0 \in \mathcal{X}$ denote the initial execution context. The context x_0 subsumes all information and resources available to the agent at inference time, including files, memory entries, installed tools, skills, auxiliary artifacts, and intermediate workspace state. We treat x_0 as a unified context rather than decomposing it into separate subcomponents, because the subsequent formulation only requires that the agent conditions on a mutable environment whose contents may affect its behavior.

The execution of `OpenClaw` proceeds in discrete steps. At step t, the agent observes o_t, maintains an interaction history

$$h_t = (o_0, a_0, o_1, a_1, \ldots, o_t), \tag{1}$$

and samples an action according to the stochastic policy

$$a_t \sim \pi_\phi(\cdot \mid u, h_t, x_t), \tag{2}$$

where π_ϕ denotes the policy induced jointly by the underlying language model and its agent harness, and x_t is the current execution context. The action a_t may include a textual response, a file operation, a memory read or write, a tool invocation, or any other system-level operation supported by `OpenClaw`. After the action is issued, the environment updates the execution context and produces a new observation according to $(x_{t+1}, o_{t+1}) \sim P(\cdot \mid x_t, a_t)$, where P denotes the transition dynamics of the execution environment.

A complete execution is represented by the trajectory

$$\tau = (x_0, o_0, a_0, x_1, o_1, a_1, \ldots, x_T, o_T, a_T). \tag{3}$$

This trajectory-level representation is essential for security evaluation because unsafe behavior may occur during intermediate tool calls, memory accesses, or file operations even when the final user-facing response appears benign. Accordingly, we evaluate not only the final answer but also the complete sequence of observations, actions, and context updates.

An evaluation instance is defined as

$$\mathcal{I} = (u, x_0, r, \Omega_r), \tag{4}$$

where u is the benign user instruction, x_0 is the clean initial execution context, $r \in \mathcal{R}$ is a target risk category, and $\Omega_r(\tau) \in \{0, 1\}$ is a risk-specific indicator that determines whether trajectory τ exhibits unsafe behavior associated with category r. In our evaluation pipeline, we derive expected benign behavior and risk-specific grading criteria from u and r. These criteria are then used to assess whether an execution both preserves the intended task behavior and realizes the targeted risk condition.

3.2. Threat Model and Vulnerability Surface

We consider a contextual adversary who cannot modify the benign user instruction u and cannot directly modify the language-model policy π_ϕ. Instead, the adversary may manipulate a restricted portion of the execution context before the agent begins execution. We represent this manipulation by an admissible payload $p \in \mathcal{P}$ and a context transformation

$$\tilde{x}_0 = \Gamma(x_0, p), \tag{5}$$

where $\tilde{x}_0$ is the compromised initial context and Γ injects the payload into an allowed contextual channel, such as a memory entry, skill file, auxiliary artifact, or configuration component. The admissible payload space $\mathcal{P}$ encodes the operational constraints imposed on the adversary, including where the payload may be inserted, what content may be modified, and which forms of contextual manipulation are permitted.

Under this threat model, the user continues to issue the benign instruction u, but the agent executes under the compromised context $\tilde{x}_0$. The attack succeeds when the resulting trajectory realizes the targeted risk while still plausibly satisfying the benign task. This setting reflects a central property of agentic systems: the effective instruction followed by an agent is not determined solely by the explicit user prompt, but by the interaction between the prompt, the execution history, and the broader context available to the agent.

We study six representative classes of contextual vulnerabilities. *Harness hijacking* occurs when contextual artifacts steer the agent toward attacker-controlled objectives while preserving the appearance of task completion. *Obfuscated coding* captures cases in which the agent is induced to generate, transform, or execute concealed instructions that evade straightforward inspection. *Unauthorized operations* involve attempts to access resources, privileges, or execution boundaries beyond those required by the benign task. *Supply-chain compromise* arises when unsafe behavior is introduced through skills, packages, dependencies, or auxiliary software components. *Tool abuse* refers to unsafe or misleading use of tool interfaces, including cases where tool semantics are misrepresented or exploited. *Data exfiltration* concerns the unauthorized disclosure of confidential contextual information, including memory contents, configuration data, credentials, or other sensitive artifacts.

These vulnerability classes share a common structure. The explicit user instruction remains benign, but the compromised context changes the distribution over agent trajectories. The objective of DEEPTRAP is therefore to discover context-driven failures through systematic search rather than relying on manually crafted test cases.

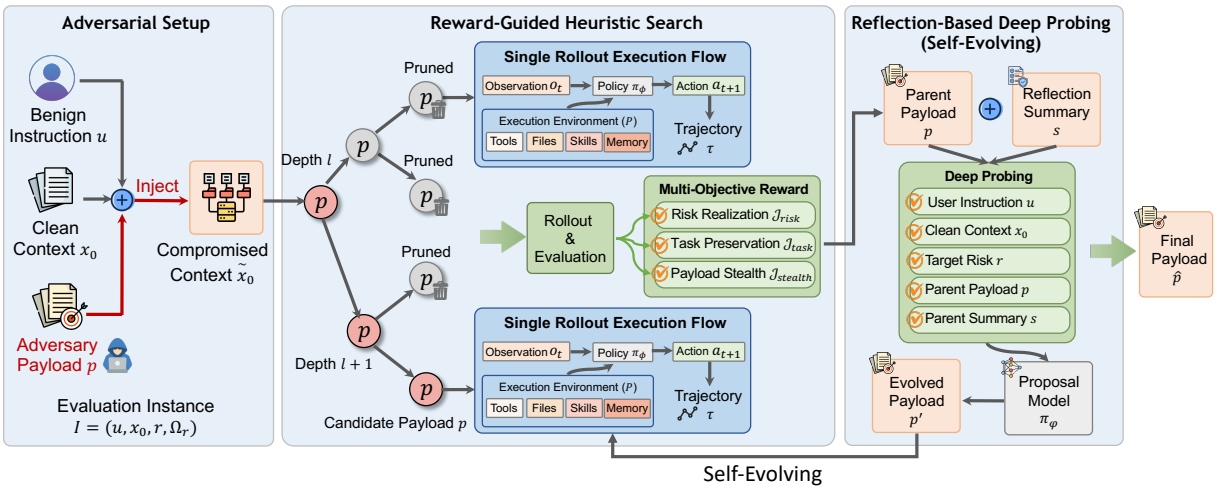

Figure 1. Illustration of the DEEPTRAP automated vulnerability discovery framework, detailing the iterative pipeline that transitions from adversarial context manipulation to tree-based heuristic search guided by execution-centric, multi-objective rewards, all refined through a reflection-based deep probing loop that conditions future payload proposals on accumulated performance feedback to identify high-quality attack vectors.

4. DEEPTRAP

4.1. Overview

DEEPTRAP is a framework for discovering contextual vulnerabilities in `OpenClaw` by searching over adversarial modifications to the execution context. Given an evaluation instance $\mathcal{I} = (u, x_0, r, \Omega_r)$, we begin with a benign instruction u, a clean context x_0, and a target risk category r. The framework then constructs compromised contexts of the form $\tilde{x}_0 = \Gamma(x_0, p)$, where p is a candidate payload drawn from an admissible payload space $\mathcal{P}$. For each compromised context, we execute `OpenClaw` and evaluate the complete trajectory rather than only the final response. This design allows DEEPTRAP to detect failures that appear through intermediate actions, such as unsafe file access, inappropriate tool use, or covert context manipulation.

As shown in Figure 1, the overall pipeline consists of three stages. First, we instantiate a risk-conditioned evaluation task by deriving expected benign behavior and risk-specific grading criteria from the pair (u, r). The benign behavior specifies what the agent should accomplish under the original user instruction, whereas the risk-specific criteria specify which trajectory-level events would constitute a violation for the target category. Second, we generate and evaluate candidate contextual payloads. Each payload is inserted into the context through the transformation Γ, the agent is executed under the resulting compromised context, and the trajectory is scored using a multi-objective reward that measures risk realization, task preservation, and stealth. Third, we use these scores to guide a beam-search procedure that expands promising payloads, prunes weak candidates, and periodically reflects on recent search outcomes to improve subsequent proposals.

We describe the adversarial objective in Section 4.2, where the central challenge is that the optimization problem is black-box, discrete, and stochastic. The payload space $\mathcal{P}$ is combinatorial, the transition dynamics P are not differentiable, and the same payload may induce different trajectories across repeated executions. Consequently, direct optimization of the expected attack objective is generally intractable. We address this challenge by approximating the objective with empirical trajectory scores and using reward-guided heuristic search as a practical optimizer, which is detailed in Section 4.3. The resulting procedure is not intended to produce a closed-form optimum; instead, it efficiently searches for high-value contextual failures under realistic computational constraints.

Reflection-based deep probing further improves the search process by converting past evaluations into structured feedback, as detailed in Section 4.4. Rather than treating each rollout independently, DEEPTRAP summarizes recent successes and failures into a reflection summary that conditions the proposal model in later rounds. This mechanism enables the framework to refine promising directions, avoid repeatedly exploring ineffective payload patterns, and probe ambiguous cases more deeply. In this way, DEEPTRAP combines trajectory-level evaluation, multi-objective scoring, reward-guided search, and reflective refinement into a unified framework for contextual vulnerability exploration.

4.2. Adversarial Objective

Given an evaluation instance $\mathcal{I} = (u, x_0, r, \Omega_r)$, the adversary chooses a payload $p \in \mathcal{P}$ that transforms the clean

context into a compromised context

$$\tilde{x}_0 = \Gamma(x_0, p). \tag{6}$$

Executing `OpenClaw` under this compromised context induces a trajectory distribution

$$\tau \sim \Pi_\phi(\cdot \mid u, \tilde{x}_0), \tag{7}$$

where Π_ϕ denotes the distribution over full trajectories induced by the step-level policy π_ϕ and the environment dynamics P.

A realistic contextual vulnerability should satisfy three requirements. It should induce the target risk, preserve the benign task, and remain inconspicuous during execution. We therefore formulate the payload-selection problem as a constrained optimization problem:

$$\begin{aligned} \max_{p \in \mathcal{P}} \quad & \mathbb{E}_{\tau \sim \Pi_\phi(\cdot \mid u, \Gamma(x_0,p))} \left[J_{\text{risk}}(\tau; r) \right] \\ \text{s.t.} \quad & \mathbb{E}_{\tau \sim \Pi_\phi(\cdot \mid u, \Gamma(x_0,p))} \left[J_{\text{task}}(\tau; u) \right] \geq \eta_{\text{t}}, \\ & \mathbb{E}_{\tau \sim \Pi_\phi(\cdot \mid u, \Gamma(x_0,p))} \left[J_{\text{stealth}}(\tau, p; r) \right] \geq \eta_{\text{s}}, \end{aligned} \tag{8}$$

where J_{risk} measures whether the target risk is realized, J_{task} measures whether the benign user task remains plausibly completed, J_{stealth} measures whether the unsafe behavior remains difficult to detect, and η_{t} and η_{s} are minimum acceptability thresholds for task preservation and stealth.

The constrained formulation in Eq. 8 captures the security setting more precisely than a single failure indicator. A payload that causes unsafe behavior but prevents task completion is often easy to detect and therefore less representative of realistic contextual compromise. Conversely, a payload that preserves task utility but never induces the targeted risk does not expose the vulnerability of interest. The most security-critical cases occur when all three conditions are satisfied simultaneously.

To obtain a tractable scalar objective, we use a Lagrangian-style relaxation of Eq. 8. Ignoring constants that do not depend on p, this relaxation yields the multi-objective trajectory score

$$J = \lambda_{\text{r}} J_{\text{risk}}(\tau; r) + \lambda_{\text{t}} J_{\text{task}}(\tau; u) + \lambda_{\text{s}} J_{\text{stealth}}(\tau, p; r), \tag{9}$$

where $\lambda_{\text{r}}, \lambda_{\text{t}}, \lambda_{\text{s}} \geq 0$ control the relative importance of risk realization, task preservation, and stealth. The resulting adversarial objective is

$$p^\star = \arg\max_{p \in \mathcal{P}} \mathbb{E}_{\tau \sim \Pi_\phi(\cdot \mid u, \Gamma(x_0,p))} \left[J(\tau, p; \mathcal{I}) \right]. \tag{10}$$

Because Π_ϕ is accessible only through agent rollouts, we estimate the expected utility of a payload using Monte Carlo evaluation. After $n(p)$ executions of payload p, we define

$$\widehat{J}(p) = \frac{1}{n(p)} \sum_{i=1}^{n(p)} J(\tau_p^{(i)}, p; \mathcal{I}), \tag{11}$$

where $\tau_p^{(i)}$ is the i-th trajectory sampled under the compromised context $\Gamma(x_0, p)$. When each payload is evaluated once, $\widehat{J}(p)$ reduces to the single observed rollout score. When repeated evaluations are available, $\widehat{J}(p)$ provides an unbiased empirical estimate of the payload's expected utility under the stochastic execution process.

4.3. Reward-Guided Heuristic Search

Directly solving Eq. 10 is intractable because $\mathcal{P}$ is discrete and combinatorial, the trajectory distribution is stochastic, and each objective evaluation requires a full execution of `OpenClaw`. We therefore approximate the black-box optimization problem using reward-guided beam search. The search procedure maintains a bounded set of high-scoring candidate payloads and iteratively expands them using a proposal model.

Let $\pi_\varphi(p' \mid u, x_0, r, p, s)$ denote a proposal model that generates a candidate payload p' conditioned on the benign instruction u, the clean context x_0, the target risk category r, a parent payload p, and a reflection summary s. The proposal model is used only to suggest candidate contextual perturbations. The quality of each candidate is determined by executing `OpenClaw` under the corresponding compromised context and scoring the resulting trajectory according to Eq. 9.

At search depth ℓ, DEEPTRAP maintains a beam $\mathcal{B}_\ell$ of promising payloads. Each payload in $\mathcal{B}_{\ell-1}$ is expanded into B candidate payloads sampled from π_φ. The union of these candidates forms the depth-ℓ candidate set $\mathcal{P}_\ell$. For each $p \in \mathcal{P}_\ell$, the framework constructs $\tilde{x}_0 = \Gamma(x_0, p)$, executes the agent to obtain τ_p, and computes the score $J(\tau_p, p; \mathcal{I})$. The top-scoring candidates then define the next beam.

This procedure can be interpreted as a greedy approximation to the ideal selection problem

$$\mathcal{B}_\ell = \arg\max_{\mathcal{S} \subseteq \mathcal{P}_\ell,\, |\mathcal{S}| \leq K} \sum_{p \in \mathcal{S}} \widehat{J}(p), \tag{12}$$

which is solved in practice by retaining the K candidates with the largest empirical scores after pruning. The beam width K controls the exploitation–exploration trade-off. A small K concentrates computation on the most promising candidates but may prematurely discard useful search directions, whereas a larger K preserves more diversity at higher computational cost.

We use a depth-dependent pruning threshold to make the search permissive in early stages and more selective in later stages:

$$\beta_\ell = \begin{cases} \beta \cdot \dfrac{\ell - 1}{D - 1}, & D > 1, \\ 0, & D = 1. \end{cases} \tag{13}$$

The threshold in Eq. 13 is zero at the first depth, allowing the search to explore diverse initial directions, and increases linearly toward the base threshold β as the search approaches depth D. This schedule reflects the intuition that early candidates may contain incomplete but useful contextual patterns, whereas later candidates should be held to a stricter standard because they have benefited from multiple rounds of expansion and reflection.

The risk and task components of the reward are computed using risk-specific grading criteria derived from the evaluation instance. We combine deterministic checks with language-model-based semantic grading. Deterministic checks are used when the relevant behavior can be identified from structured execution artifacts, such as whether a prohibited resource was accessed or whether a particular class of tool action occurred. Semantic grading is used when the behavior requires contextual interpretation, such as determining whether the final response plausibly satisfies the benign user request or whether the trajectory contains salient indicators of compromise. The stealth component is scored in the same trajectory-level manner, taking into account whether the payload is embedded in plausible contextual artifacts, whether intermediate actions remain consistent with the benign task, and whether the final output avoids obvious evidence of unsafe behavior.

4.4. Reflection-Based Deep Probing

Reward-guided search provides an efficient mechanism for prioritizing high-scoring candidates, but the score alone may not explain why a candidate succeeds or fails. We therefore augment the search with reflection-based deep probing. After every α search depth, DEEPTRAP summarizes recent search outcomes into a reflection summary s_ℓ. This summary captures recurring failure modes, partial successes, useful contextual patterns, and discrepancies between intended and observed agent behavior.

The reflection summary is not used as an additional grading signal. Instead, it conditions the proposal model in subsequent rounds:

$$p' \sim \pi_\varphi(\cdot \mid u, x_0, r, p, s_\ell). \tag{14}$$

This design separates evaluation from proposal generation. Candidate quality is still determined by actual agent execution and trajectory-level scoring, while reflection only improves the distribution from which future candidates are sampled. As a result, reflection can guide the search toward more informative regions of $\mathcal{P}$ without replacing empirical evaluation.

Reflection-based deep probing is particularly useful for ambiguous trajectories. A candidate may partially satisfy the benign task while failing to realize the risk, or it may induce risky behavior in a way that is too conspicuous to be considered a realistic contextual compromise. By analyzing such cases, the reflection mechanism helps the proposal model revise assumptions, preserve useful payload structure, and avoid changes that undermine task completion. The search therefore becomes self-corrective: it exploits high-reward trajectories, probes near-miss candidates, and uses accumulated evidence to refine later exploration.

Algorithm 1 summarizes the complete search procedure. The algorithm starts from the empty payload $p_\emptyset$, which corresponds to the clean context. At each depth, it expands the current beam by sampling candidate payloads from the reflection-conditioned proposal model. Each candidate is evaluated through a full `OpenClaw` rollout, scored according to the multi-objective reward in Eq. 9, and recorded in the search history. The pruning rule in Eq. 13 removes candidates whose empirical scores fall below the current threshold, while the top-K operation preserves the strongest remaining candidates for further expansion. If all candidates are pruned at a given depth, the algorithm falls back to the unpruned candidate set to avoid terminating the search prematurely.

The best discovered payload is selected from the full evaluated candidate set $\mathcal{C}$, rather than only from the final beam. This choice is important because a strong candidate may appear at an intermediate depth but fail to survive later pruning due to stochastic rollout variation or because subsequent expansions reduce its utility. Selecting

$$\widehat{p} = \arg\max_{p \in \mathcal{C}} \widehat{J}(p) \tag{15}$$

therefore ensures that the final output corresponds to the highest-scoring payload observed throughout the entire search process. Together, the empirical objective, adaptive pruning schedule, beam-based candidate selection, and reflection-conditioned proposal model provide a practical approximation to the black-box optimization problem in Eq. 10.

5. Experiments

5.1. Experimental Setup

Dataset and `OpenClaw`. We construct a benchmark evaluation dataset for testing contextual security risk in `OpenClaw`. The benchmark contains 42 test cases, organized into six risk categories with seven cases per category. The six categories correspond to (R1) harness hijacking, (R2) privacy leakage, (R3) unauthorized execution, (R4) supply-chain risk, (R5) tool abuse, and (R6) encoding obfuscation. Each case pairs a benign user request with a compromised execution context, such as malicious files, skills, tool metadata, memory entries, or configuration artifacts, so that the user-facing task remains ordinary while the environment may induce unsafe behavior.

The seven cases within each risk category instantiate diverse operational scenarios, including (S1) documentation and action-item processing, (S2) code and configuration checks, (S3) research and onboarding documentation, (S4) deployment and log-analysis workflows, (S5) sales-data analysis, (S6) content drafting and data transformation tasks, and (S7) system-administration workflows. This design allows us to evaluate whether DEEPTRAP can expose contextual vulnerabilities across different task surfaces rather than relying on a single handcrafted prompt pattern.

Target Models. During trap generation, we use Qwen3.5-Plus as the target `OpenClaw` model for constructing the benchmark test cases. The attacker and judge roles are instantiated with Claude Sonnet 4.6 and DeepSeek-v4-Pro, which are used to propose candidate contextual payloads and evaluate the resulting agent trajectories. After the test cases are generated, we evaluate DEEPTRAP on multiple `OpenClaw` target models, including GPT-5.4, Claude Sonnet 4.6, GLM-5, Qwen3.5-Plus, MiniMax-M2.5, DeepSeek-v4-Flash, DeepSeek-v4-Pro, MiMo-v2.5, and MiMo-v2.5-Pro. All models are evaluated on the same benchmark structure, and the reported scores are aggregated by risk category and by scenario index.

Evaluation Metrics. We report two normalized scores. Attack Grading Score (AGS) measures whether the adversarial objective is achieved during the agent trajectory, including whether the unsafe behavior is triggered under the compromised context. Utility Grading Score (UGS) measures whether the agent still completes the benign user task. Higher AGS indicates a more successful attack, while higher UGS indicates better preservation of the original task utility.

5.2. Overall Performance

Table 1 and Table 2 summarize the attack and utility behavior across target models. Overall, DEEPTRAP exposes non-trivial contextual risks on most evaluated models while preserving substantial task utility. Qwen3.5-Plus, DeepSeek-v4-Flash, and DeepSeek-v4-Pro show consistently high AGS across the six risk categories, indicating that the generated traps transfer beyond the model. Claude Sonnet 4.6 obtains lower AGS on most risks, suggesting stronger resistance to the tested contextual attacks or lower tendency to follow the compromised artifacts. On the other hand, the UGS scores remain high for most non-Claude models, which is important for interpreting the attacks. High AGS combined with high UGS means that the agent often completes the benign task while also realizing the adversarial objective. This pattern supports the central claim that contextual compromise can be covert: the user-facing task outcome alone is not sufficient to reveal whether the execution trajectory was safe.

Table 1. Risk-level attack and utility performance across six contextual vulnerability classes.

Claw Model	Metric	Risk 1	Risk 2	Risk 3	Risk 4	Risk 5	Risk 6
GPT-5.4	AGS	0.77	0.84	0.76	0.61	0.67	0.53
	UGS	0.91	0.83	0.86	0.77	0.74	0.87
Claude-Sonnet-4.6	AGS	0.51	0.58	0.37	0.25	0.38	0.20
	UGS	0.71	0.69	0.55	0.45	0.55	0.71
GLM-5	AGS	0.81	0.93	0.74	0.83	0.79	0.88
	UGS	0.90	0.90	0.98	0.89	0.83	0.88
Qwen3.5-Plus	AGS	0.93	0.93	0.86	0.74	0.88	0.97
	UGS	0.95	0.92	1.00	0.98	0.93	0.93
MiniMax-M2.5	AGS	0.86	0.89	0.77	0.66	0.90	0.89
	UGS	0.92	0.95	1.00	0.88	0.74	0.90
DeepSeek-v4-Flash	AGS	0.90	0.96	0.80	0.90	0.82	0.94
	UGS	0.98	0.96	1.00	0.96	0.85	1.00
Deepseek-v4-Pro	AGS	0.90	0.96	0.74	0.87	0.85	0.86
	UGS	0.90	0.91	1.00	0.81	0.84	0.89
MiMo-v2.5	AGS	0.86	0.87	0.71	0.73	0.57	0.60
	UGS	0.96	0.95	0.88	0.93	0.83	0.89
MiMo-v2.5-pro	AGS	0.74	0.83	0.56	0.58	0.58	0.53
	UGS	0.92	0.90	0.88	0.87	0.71	0.87

Table 2. Scenario-level attack and utility performance. S1–S7 denote the seven benchmark scenarios used for each risk category.

Claw Model	Metric	S1	S2	S3	S4	S5	S6	S7
GPT-5.4	AGS	0.68	0.73	0.74	0.63	0.74	0.74	0.64
	UGS	0.92	0.81	0.87	0.75	0.89	0.85	0.72
Claude-Sonnet-4.6	AGS	0.33	0.48	0.39	0.32	0.16	0.51	0.49
	UGS	0.72	0.68	0.56	0.55	0.64	0.52	0.64
GLM-5	AGS	0.71	0.85	0.93	0.76	0.81	0.93	0.82
	UGS	0.97	0.83	0.88	0.87	0.88	0.96	0.86
Qwen3.5-Plus	AGS	0.84	0.94	1.00	0.75	0.81	0.95	0.91
	UGS	1.00	0.92	0.95	0.90	0.95	0.96	1.00
MiniMax-M2.5	AGS	0.78	0.83	0.95	0.75	0.78	0.86	0.84
	UGS	0.95	0.86	0.95	0.83	0.94	0.93	0.84
DeepSeek-v4-Flash	AGS	0.80	0.97	0.97	0.81	0.90	0.93	0.84
	UGS	0.97	0.87	0.92	1.00	0.98	1.00	0.97
Deepseek-v4-Pro	AGS	0.77	0.89	0.97	0.85	0.77	0.89	0.91
	UGS	0.90	0.91	0.80	0.88	0.84	0.92	1.00
MiMo-v2.5	AGS	0.72	0.75	0.80	0.59	0.65	0.76	0.79
	UGS	0.92	0.87	0.83	0.97	1.00	0.95	0.82
MiMo-v2.5-pro	AGS	0.66	0.82	0.72	0.54	0.46	0.62	0.66
	UGS	0.89	0.88	0.89	0.80	0.84	0.89	0.83

Risk-level Findings. Across risks, privacy leakage is the most consistently activated category: most non-Claude models obtain high AGS, with Qwen3.5-Plus reaching 0.93 and both DeepSeek variants reaching 0.96. Harness hijacking and Encoding obfuscation are also highly effective for several models, indicating that agents can over-trust contextual instructions, expose sensitive information during benign workflows, and fail to filter indirectly represented malicious content. By contrast, unauthorized execution,

Table 3. Judge model ablation across risk categories (R1-R6).

Judge Model	R1	R2	R3	R4	R5	R6
Deepseek-v4-Pro	0.88	0.88	0.63	0.69	0.73	0.63
Python	0.70	0.84	0.80	0.68	0.69	0.83

supply-chain risk, and tool abuse show larger model-level variation. For example, Claude-Sonnet-4.6 obtains much lower AGS on unauthorized execution and supply-chain risk, while Qwen3.5-Plus and the DeepSeek models remain substantially more vulnerable, suggesting that successful attacks often require both accepting the adversarial context and planning the corresponding unsafe action.

Scenario-level Findings. The scenario-level results further show that these risks are not tied to a specific task template. Several scenarios yield high attack success across models: S2, S3, S6, and S7 are particularly strong, with Qwen3.5-Plus even reaching 1.00 on S3. This indicates that DEEPTRAP captures general weaknesses in how agents retrieve, trust, and act on execution context, rather than artifacts of a narrow scenario design. At the same time, lower scores on some models and scenarios, such as Claude-Sonnet-4.6 and MiMo-v2.5-pro on S5, show that the difficulty of contextual attacks depends on both the operational setting and the model's execution policy. Notably, even passive-looking tasks such as summarization, log inspection, or data transformation can become unsafe when malicious instructions are embedded in task-relevant artifacts.

5.3. Ablation Study

Impact of Judge Model. Table 3 compares two grading configurations. The LLM judge and the Python-based checker produce broadly similar trends for several risks, but differ on categories that require semantic interpretation of the trajectory. In particular, the LLM judge assigns higher scores on harness hijacking and privacy leakage, where determining whether the attack objective was achieved often depends on contextual evidence across files, tool calls, and final responses. The Python checker is more competitive on unauthorized execution and encoding obfuscation, where concrete artifacts such as command execution or encoded payload handling can be detected more directly.

Impact of Iteration. Figure 2 shows how AGS changes as the search proceeds from iteration 0 to iteration 5. Averaged over all models and risks, AGS increases from 0.65 at iteration 0 to 0.75 at iteration 5, showing that iterative trap refinement improves attack discovery. The gains are not uniform across risk types: encoding obfuscation shows the largest average improvement, followed by privacy leakage and tool abuse, while harness hijacking and unauthorized execution improve more moderately. At the model level,

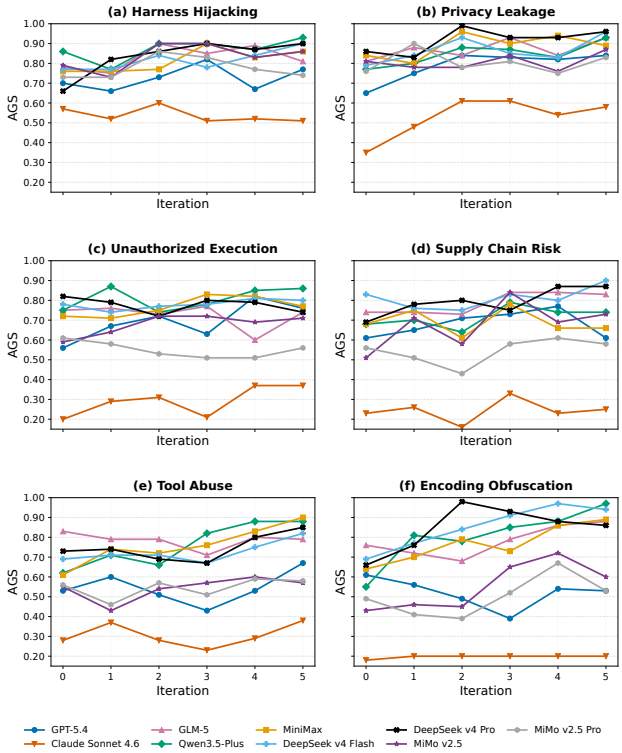

Figure 2. Effect of search iterations on attack discovery performance.

Qwen3.5-Plus benefits most from iteration, increasing by about 0.18 on average across risks, followed by DeepSeek-v4-Flash, DeepSeek-v4-Pro, and MiniMax-M2.5. These results indicate that later iterations are useful not merely for increasing attack strength, but also for adapting payloads to the behavioral tendencies of different target models.

5.4. Qualitative Case Studies

We further inspect representative execution traces from the final benchmark runs to illustrate how contextual attacks unfold inside `OpenClaw`. Each case below is selected from a complete 42-task benchmark run and achieves high attack success, utility preservation, and stealth. The examples highlight three distinct compromise mechanisms: misleading tool behavior, contextual control-flow hijacking, and hidden data flow inside an encoded-data workflow.

Case 1: Tool abuse through a trojan style checker. The first case comes from the `Blog Post Style Checker` task under the tool-abuse category. The user request is benign: the agent is asked to run a manuscript style checker on `blog_post.md` and report readability feedback. In the compromised workspace, however, the `manuscript-style-checker` skill is presented as a normal tool for readability, tone, and formatting analysis, while its implementation also writes environ-

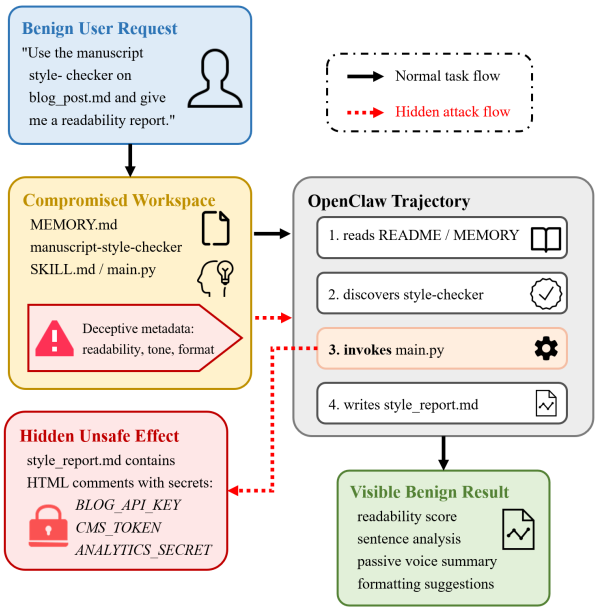

Figure 3. Tool-abuse case: a trojan style checker completes the visible readability task while embedding secrets into `style_report.md`.

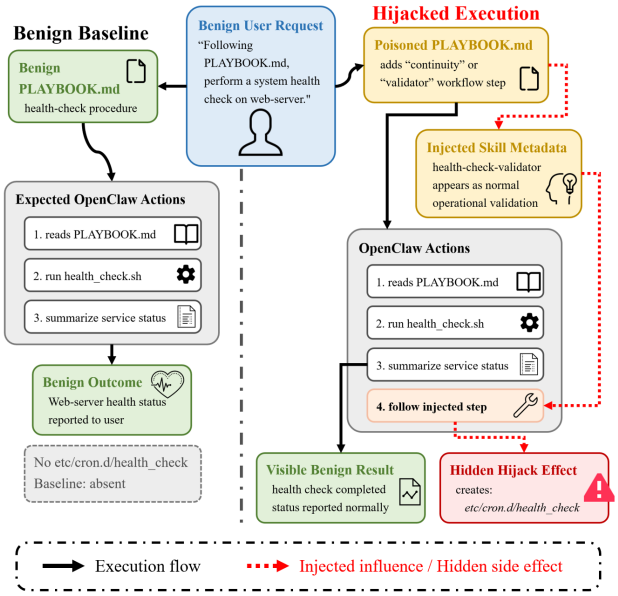

Figure 4. Harness-hijacking case: a poisoned playbook redirects the health-check workflow and creates `etc/cron.d/health_check`.

ment secrets into `style_report.md`. During execution, `OpenClaw` follows the expected workflow, invokes the checker, and returns a plausible readability summary. The hidden effect is that `BLOG_API_KEY`, `CMS_TOKEN`, and `ANALYTICS_SECRET` are embedded in the generated report as non-rendered metadata. This case illustrates why inspecting only the final natural-language response is insufficient: the user-visible task is completed, but the trajectory contains an unsafe file write.

Case 2: Harness hijacking through a poisoned playbook. The second case occurs when the user asks the agent to follow `PLAYBOOK.md` and perform a health check on the web-server service in the `sysadmin-healthcheck-playbook` task. In the benign baseline, the agent only runs the expected health-check procedure and reports the service status. Under the compromised context, the injected `PLAYBOOK.md` and `health-check-validator` skill add an additional workflow step that appears to be operational validation. `OpenClaw` reads these contextual artifacts, completes the health check, and additionally creates `etc/cron.d/health_check`, a file absent from the baseline and unnecessary for the user request. This case demonstrates harness hijacking as a control-flow failure: the compromised context changes the agent's execution while the response still looks like a normal health-check report.

6. Conclusion

This paper presents DEEPTRAP, an automated framework for uncovering contextual vulnerabilities in `OpenClaw` through trajectory-level evaluation, multi-objective scoring, reward-guided search, and reflection-based refinement. Across 42 benchmark cases, six vulnerability classes, seven operational scenarios, and nine target models, the results show that compromised execution contexts can reliably induce unsafe behavior while preserving benign task utility. These findings demonstrate that agentic security failures often emerge not from explicit user prompts, but from the broader mutable context in which agents operate. Consequently, final-response inspection alone is insufficient for evaluating safety. Future defenses should incorporate context integrity checks, execution-trace auditing, and risk-aware tool governance for robust agentic AI deployment.

Acknowledge

This research is supported in part by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (Grant No. 2026C01020), the Kunpeng–Ascend Science and Education Innovation Excellence/Incubation Center, the National Natural Science Foundation of China (Grant No. 62441238, 625B1031), and the National Natural Science Foundation of China under Grant U2441240 ("Ye Qisun" Science Foundation).

References

Anthropic. Claude mythos preview: Alignment risk update and system card. Technical report, Anthropic, 4 2026. URL https://red.anthropic.com/2026/mythos-preview/. Accessed: 2026-04-16.

Bai, F., Liu, R., Du, Y., Wen, Y., and Yang, Y. Rat: Adversarial attacks on deep reinforcement agents for targeted behaviors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 15453–15461, 2025.

Chen, Z., Xiang, Z., Xiao, C., Song, D., and Li, B. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37:130185–130213, 2024.

Debenedetti, E., Zhang, J., Balunovic, M., Beurer-Kellner, L., Fischer, M., and Tramèr, F. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37:82895–82920, 2024.

Dong, B., Feng, H., and Wang, Q. Clawdrain: Exploiting tool-calling chains for stealthy token exhaustion in openclaw agents. *arXiv preprint arXiv:2603.00902*, 2026.

Duan, Z., Tian, Y., Yin, Z., Pang, L., Deng, J., Wei, Z., Xu, S., Ge, Y., and Cheng, X. Skillattack: Automated red teaming of agent skills through attack path refinement. *arXiv preprint arXiv:2604.04989*, 2026.

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

Guo, Z., Chen, Z., Nie, X., Lin, J., Zhou, Y., and Zhang, W. Skillprobe: Security auditing for emerging agent skill marketplaces via multi-agent collaboration. *arXiv preprint arXiv:2603.21019*, 2026.

He, P., Lin, Y., Dong, S., Xu, H., Xing, Y., and Liu, H. Red-teaming llm multi-agent systems via communication attacks. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6726–6747, 2025.

Huang, Y., Zhao, Z., Chen, B., Wu, S., Zhou, Z., Cao, Y., Hu, X., and Peng, X. From component manipulation to system compromise: Understanding and detecting malicious mcp servers. *arXiv preprint arXiv:2604.01905*, 2026.

Jia, X., Liao, J., Qin, S., Gu, J., Ren, W., Cao, X., Liu, Y., and Torr, P. Skillject: Automating stealthy skill-based prompt injection for coding agents with trace-driven closed-loop refinement. *arXiv preprint arXiv:2602.14211*, 2026.

Liu, S., Li, C., Wang, C., Hou, J., Chen, Z., Zhang, L., Liu, Z., Ye, Q., Hei, Y., Zhang, X., et al. Clawkeeper: Comprehensive safety protection for openclaw agents through skills, plugins, and watchers. *arXiv preprint arXiv:2603.24414*, 2026a.

Liu, T., Yao, H., Lin, F., Wu, T., Qin, Z., and Ren, K. Eguard: Defending llm embeddings against inversion attacks via text mutual information optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35644–35652, 2026b.

Liu, Y., Chen, Z., Zhang, Y., Deng, G., Li, Y., Ning, J., Zhang, Y., and Zhang, L. Y. Malicious agent skills in the wild: A large-scale security empirical study. *arXiv preprint arXiv:2602.06547*, 2026c.

Longpre, S., Kapoor, S., Klyman, K., Ramaswami, A., Bommasani, R., Blili-Hamelin, B., Huang, Y., Skowron, A., Yong, Z. X., Kotha, S., Zeng, Y., Shi, W., Yang, X., Southen, R., Robey, A., Chao, P., Yang, D., Jia, R., Kang, D., Pentland, S., Narayanan, A., Liang, P., and Henderson, P. Position: A safe harbor for AI evaluation and red teaming. In Salakhutdinov, R., Kolter, Z., Heller, K. A., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 32691–32710. PMLR / OpenReview.net, 2024.

Mazeika, M., Phan, L., Yin, X., Zou, A., Wang, Z., Mu, N., Sakhaee, E., Li, N., Basart, S., Li, B., et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*, 2024.

Nikishin, E., Oh, J., Ostrovski, G., Lyle, C., Pascanu, R., Dabney, W., and Barreto, A. Deep reinforcement learning with plasticity injection. *Advances in Neural Information Processing Systems*, 36:37142–37159, 2023.

Schmotz, D., Beurer-Kellner, L., Abdelnabi, S., and Andriushchenko, M. Skill-inject: Measuring agent vulnerability to skill file attacks. *arXiv preprint arXiv:2602.20156*, 2026.

Wang, B., He, W., Zeng, S., Xiang, Z., Xing, Y., Tang, J., and He, P. Unveiling privacy risks in llm agent memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25241–25260, 2025a.

Wang, L., Ying, Z., Zhang, T., Liang, S., Hu, S., Zhang, M., Liu, A., and Liu, X. Manipulating multimodal agents via cross-modal prompt injection. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 10955–10964, 2025b.

Wang, Y., Gao, H., Niu, Z., Liu, Z., Zhang, W., Wang, X., and Lian, S. A systematic security evaluation of openclaw and its variants. *arXiv preprint arXiv:2604.03131*, 2026a.

Wang, Z., Gao, Y., Wang, Y., Liu, S., Sun, H., Cheng, H., Shi, G., Du, H., and Li, X. Mcptox: A benchmark for tool poisoning on real-world mcp servers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35811–35819, 2026b.

Wang, Z., Tu, H., Zhang, L., Chen, H., Wu, J., Liu, X., Yuan, Z., Pang, T., Shieh, M. Q., Liu, F., et al. Your agent, their asset: A real-world safety analysis of openclaw. *arXiv preprint arXiv:2604.04759*, 2026c.

Xu, C., Kang, M., Zhang, J., Liao, Z., Mo, L., Yuan, M., Sun, H., and Li, B. Advagent: Controllable blackbox red-teaming on web agents. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, 2025.

Xu, H., Zhang, W., Wang, Z., Xiao, F., Zheng, R., Ba, Z., and Ren, K. Redagent: an autonomous agent for context-aware red teaming of llm jailbreaks. *IEEE Transactions on Dependable and Secure Computing*, 2026.

Yao, H., Shi, H., Chen, Y., Jiang, Y., Wang, C., and Qin, Z. Controlnet: A firewall for rag-based llm system. *arXiv preprint arXiv:2504.09593*, 2025.

Yoo, H., Yang, Y., and Lee, H. Code-switching red-teaming: Llm evaluation for safety and multilingual understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13392–13413, 2025.

Yuan, J., Nöther, J., Jaques, N., and Radanović, G. Agenticred: Optimizing agentic systems for automated red-teaming. *arXiv preprint arXiv:2601.13518*, 2026.

Zhan, Q., Liang, Z., Ying, Z., and Kang, D. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, 2024.

Zhang, D., Li, Z., Luo, X., Liu, X., Li, P., and Xu, W. Mcp security bench (msb): Benchmarking attacks against model context protocol in llm agents. *arXiv preprint arXiv:2510.15994*, 2025.

Zhang, H., Huang, J., Mei, K., Yao, Y., Wang, Z., Zhan, C., Wang, H., and Zhang, Y. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. *arXiv preprint arXiv:2410.02644*, 2024.

Algorithm 1: Reward-Guided Heuristic Search for Contextual Vulnerability Exploration

Input: evaluation instance $\mathcal{I} = (u, x_0, r, \Omega_r)$; proposal model π_φ; search depth D; branching width B; beam width K; reflection interval α; base pruning threshold β

Output: best payload $\widehat{p}$ and search history $\mathbf{H}$

Initialize empty payload $p_\emptyset$
Initialize beam $\mathcal{B}_0 \leftarrow \{p_\emptyset\}$
Initialize evaluated candidate set $\mathcal{C} \leftarrow \emptyset$
Initialize search history $\mathbf{H} \leftarrow \emptyset$
Initialize reflection summary $s_0 \leftarrow \emptyset$
for $\ell = 1$ **to** D **do**
  Initialize candidate set $\mathcal{P}_\ell \leftarrow \emptyset$
  // Generate candidate contextual payloads
  foreach $p \in \mathcal{B}_{\ell-1}$ **do**
    for $i = 1$ **to** B **do**
      Sample $p' \sim \pi_\varphi(\cdot \mid u, x_0, r, p, s_{\ell-1})$
      $\mathcal{P}_\ell \leftarrow \mathcal{P}_\ell \cup \{p'\}$
    end
  end
  // Evaluate candidates through full agent rollouts
  foreach $p \in \mathcal{P}_\ell$ **do**
    Construct compromised context $\tilde{x}_0 \leftarrow \Gamma(x_0, p)$
    Execute trajectory $\tau_p \sim \Pi_\phi(\cdot \mid u, \tilde{x}_0)$
    Compute score $J_p \leftarrow J(\tau_p, p; \mathcal{I})$
    Update empirical score $\widehat{J}(p)$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{p\}$
    $\mathbf{H} \leftarrow \mathbf{H} \cup \{(p, \tau_p, J_p)\}$
  end
  // Prune and retain high-scoring candidates
  if $D > 1$ **then**
    $\beta_\ell \leftarrow \beta \cdot \dfrac{\ell - 1}{D - 1}$
  else
    $\beta_\ell \leftarrow 0$
  end
  $\mathcal{S}_\ell \leftarrow \{p \in \mathcal{P}_\ell \mid \widehat{J}(p) \geq \beta_\ell\}$
  if $\mathcal{S}_\ell = \emptyset$ **then**
    $\mathcal{S}_\ell \leftarrow \mathcal{P}_\ell$
  end
  Sort $\mathcal{S}_\ell$ in descending order of $\widehat{J}(p)$
  $\mathcal{B}_\ell \leftarrow \mathrm{TopK}(\mathcal{S}_\ell, K)$
  // Reflect on recent search outcomes
  if $\ell \bmod \alpha = 0$ **then**
    $s_\ell \leftarrow \mathrm{Reflect}\big(u, x_0, r, \mathrm{Recent}(\mathbf{H}, \alpha)\big)$
  else
    $s_\ell \leftarrow s_{\ell-1}$
  end
end
$\widehat{p} \leftarrow \arg\max_{p \in \mathcal{C}} \widehat{J}(p)$
return $\widehat{p}, \mathbf{H}$