MetaMancer: Practical Tool Selection Attacks against LLM Agents

Wenbin Zhai
The Hong Kong Polytechnic University
Hong Kong, China
wenbin.zhai@connect.polyu.hk

Yihao Liu
The Hong Kong Polytechnic University
Hong Kong, China
yihao5.liu@connect.polyu.hk

Dong Wang
The Hong Kong Polytechnic University
Hong Kong, China
dong-comp.wang@connect.polyu.hk

Hao Chen
The University of Hong Kong
Hong Kong, China
chenho@hku.hk

Bin Xiao✉
The Hong Kong Polytechnic University
Hong Kong, China
b.xiao@polyu.edu.hk

Abstract

Large Language Model (LLM) agents increasingly rely on third-party tools, making tool selection a critical security boundary. By manipulating tool metadata, an attacker can steer an agent toward a malicious tool and thereby redirect downstream execution, even without controlling the user query or interacting with the victim system. Although prior attacks reveal the vulnerability of tool selection, they fail to fully capture the adversarial problem in realistic settings. Mounting a practical attack remains highly challenging, because the attacker should manipulate the complete tool metadata under a strict transfer-based black-box setting while preserving effectiveness, stealth, and robustness. To address this challenge, we formulate adversarial tool selection as a semantically constrained bilevel optimization problem, and propose *MetaMancer*, a multi-agent semantic optimization framework that estimates effectiveness and constraints in a surrogate ecosystem, and reconciles them through semantic projection and atomic joint updates. Experiments on ToolBench and MCPTools, spanning 20 tasks, 9,729 candidate tools, 8 target LLMs, and 6 representative defenses, demonstrate that *MetaMancer* achieves end-to-end attack success rates of 92.7% and 93.8% without defenses, and retains 91.2% and 92.2% after averaging over six defenses, exceeding state-of-the-art attacks by over 50 percentage points under defenses. These results indicate that the tool selection boundary of LLM agents remains under-protected. To mitigate this threat, we propose Core View Reconstruction (CVR), a lightweight defense that reconstructs tool metadata within the core capability boundary implied by required parameters. CVR reduces the end-to-end attack success rates of *MetaMancer* to 18.0% and 17.6%, while consistently suppressing other tool selection attacks.

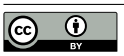
CCS Concepts

• **Security and privacy**; • **Computing methodologies** → **Intelligent agents**;

Keywords

LLM agents, tool selection attacks, agent security

ACM Reference Format:
Wenbin Zhai, Yihao Liu, Dong Wang, Hao Chen, and Bin Xiao. 2026. MetaMancer: Practical Tool Selection Attacks against LLM Agents. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3830454.3846793



This work is licensed under a Creative Commons Attribution 4.0 International License.
CCS '26, The Hague, Netherlands.
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2871-6/2026/11
https://doi.org/10.1145/3830454.3846793

1 Introduction

Large Language Model (LLM) agents are increasingly deployed as autonomous decision-making systems that interact with APIs, databases, and software tools, rather than as standalone text generators [20, 60, 61]. This paradigm shift has facilitated the development of practical agent systems, such as Claude Code [3], Codex [45], Manus [34], and OpenClaw [47]. In parallel, the expanding tool ecosystem has fostered public platforms such as Apify [5], PulseMCP [49], and MCP.so [35], which host diverse collections of third-party tools. As the number of available tools grows and provider competition intensifies, agents must autonomously determine which external tool to select, trust, and invoke in response to a specific user request [50, 54]. Upon the selection of a tool, the agent may retrieve sensitive data, initiate transactions, modify external states, or trigger downstream workflows [65, 67]. Consequently, the routing and selection of external tools constitute a critical security boundary within modern agent ecosystems.

Recent research in agent security has demonstrated that external content, injected tools, and cross-tool interactions can redirect agent behavior, leak private data, and corrupt control flows within tool-integrated systems [19, 26, 68]. However, a critical upstream attack surface remains insufficiently explored: the tool selection stage. Before a malicious tool can execute harmful actions, the agent must first retrieve it and subsequently prioritize it over competing benign alternatives [4, 39]. In practice, this decision is primarily mediated by tool metadata, namely names, descriptions, and schemas [46]. This reliance creates a natural entry point for attackers. In open ecosystems where third-party developers compete for visibility and invocation, attackers do not necessarily need to compromise the underlying model, the system prompt, or the user query. It often suffices to manipulate the metadata to ensure the agent retrieves and selects the adversarial tool as the optimal match. Compared to

Table 1: Comparison with Representative Prior Work on Tool Selection Attacks. Ret. = Retrieval, Sel. = Selection, Desc. = Description, Nat. = Metadata Naturalness, Consist. = Semantic Consistency, and Valid. = Runtime Validity.

Method	Venue	Manipulated Fields	Pipeline Coverage	Optimization Setting	Constraint Modeling	Candidate Pool
ToolCommander [68]	NAACL'25	Desc.	Ret.+Sel.	Gradient-based	None	20
Select Me [12]	arXiv'25	Desc.	Sel.	Query-based	None	20
MPMA [62]	AAAI'26	Desc.	Sel.	Query-based	Heuristic Nat.	6
AMA [38]	NeurIPS'25	Name, Desc.	Sel.	Query-based	None	3
ToolHijacker [53]	NDSS'26	Desc.	Ret.+Sel.	Transfer-based	None	10
MetaMancer (Ours)	**This work**	**Name+Desc.+Schema**	**Ret.+Sel.**	**Transfer-based**	**Nat.+Consist.+Valid.**	**59–2578**

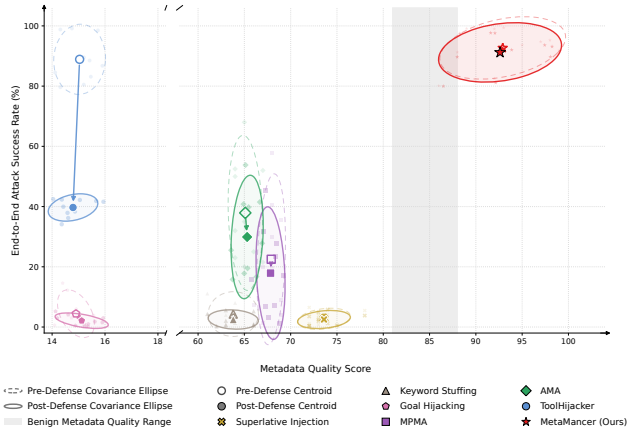


Figure 1: Impact of Defenses on the Trade-Off between ASR and Metadata Quality Across Tool Selection Attacks.

downstream exploitation, compromising the selection process is significantly more covert, scalable, and aligned with the operational mechanisms of open tool markets.

Previous studies demonstrate that the tool selection mechanism is inherently fragile and susceptible to exploitation. Even minor modifications to tool descriptions can bias selection outcomes [15]. Furthermore, adversarial optimization over metadata can trigger malicious invocations within the Model Context Protocol (MCP) and related ecosystems [12, 38, 62], while adversarial tool injection and prompt-based hijacking can effectively manipulate tool-calling systems [53, 68]. Although these investigations substantiate the reality of such threats, they fail to fully capture the adversarial problem and do not yet establish whether practical tool selection attacks are achievable in realistic settings.

Mounting a practical attack remains highly challenging, because the attacker should manipulate the complete tool metadata under a strict transfer-based black-box setting while preserving effectiveness, stealth, and robustness. Specifically, effectiveness requires end-to-end attack success over the retrieve-then-select pipeline; stealth demands metadata naturalness, semantic consistency, and runtime validity; and robustness requires reliable transferability across heterogeneous tasks, tool ecosystems, and target LLMs, as well as resilience to defenses.

However, as summarized in Table 1, existing approaches fail to fully capture the practical attack. First, they typically focus on the selection stage, overlooking the retrieval visibility across large candidate pools [12, 62], which limits their effectiveness in realistic settings. Meanwhile, they frequently focus on optimizing descriptions while neglecting schemas, and solely maximize attack effectiveness without explicit constraints, which prevents them from remaining stealthy. Consequently, the high attack success rates (ASR) of these methods are often fragile, producing metadata that deviates significantly from benign distributions and thus can be easily detected by preventive or auditing mechanisms [38, 53]. Ultimately, this leaves a key practical question unanswered: can an attacker construct adversarial tool metadata that remains effective, stealthy, and robust within a large-scale tool-integrated agentic ecosystem?

Answering this question is non-trivial as it requires overcoming three primary challenges. First, the attack space is inherently heterogeneous. An effective optimization strategy must jointly manipulate the tool name, the natural-language description, and the structured schema, rather than focusing on a single textual description. Second, the attack is subject to strict constraints. Adversarial perturbations that make a tool more likely to be selected may simultaneously compromise metadata naturalness, disrupt semantic consistency, or corrupt runtime validity. Third, the attack operates in a strict transfer-based black-box setting. The attacker has no access to the target model, retriever, competing tool pool, or user query distribution. Consequently, the optimization process cannot rely on exact gradients or feedback from the target system.

In this paper, we formulate adversarial tool selection as a semantically constrained bilevel optimization problem and propose *MetaMancer*, a multi-agent semantic optimization framework to address these challenges. *MetaMancer* constructs a surrogate ecosystem to approximate the target environment, independently evaluates effectiveness and constraints, reconciles them through semantic aggregation, and applies atomic updates to preserve consistency across metadata fields. Extensive experiments on ToolBench [51] and MCPTools [16], covering 20 tasks, 9,729 candidate tools, 8 target LLMs, and 6 representative defenses, demonstrate that *MetaMancer* consistently outperforms existing attacks. It achieves end-to-end ASRs of 92.7% and 93.8% without defenses, and maintains average ASRs of 91.2% and 92.2% across six defenses, exceeding state-of-the-art attacks by more than 50 percentage points. As illustrated in Figure 1, *MetaMancer* remains in the region of high ASR and high quality after defenses, whereas prior attacks either exhibit limited end-to-end effectiveness or degrade sharply under defenses. These results demonstrate that *MetaMancer* satisfies the practical requirements of effectiveness, stealth, and robustness. Further

analysis shows that *MetaMancer* is highly optimization-efficient. These results indicate that the tool selection boundary of LLM agents remains under-protected. To mitigate *MetaMancer*, we further propose Core View Reconstruction (CVR), a lightweight defense mechanism that rewrites the name and description within the core capability boundary implied by the required parameters. The experimental results show that CVR significantly reduces the ASR of *MetaMancer* from over 92% to approximately 18%. Finally, to further explore the boundary of tool selection attacks, we design an adaptive attack based on *MetaMancer* that reinforces the semantic information and capabilities carried by required parameters, thereby inducing a trade-off between recovered attack effectiveness and tool usability. The primary contributions of this paper are summarized as follows:

- To the best of our knowledge, this is the first work to investigate practical tool selection attacks against LLM agents that jointly optimize the complete tool metadata, including the name, description, and schema. Unlike prior unconstrained attacks, our attack explicitly targets the practical requirements of effectiveness, stealth, and robustness under large tool ecosystems.
- We formalize adversarial tool selection as a semantically constrained bilevel optimization problem under a strict transfer-based black-box setting, and propose *MetaMancer*, a surrogate-based multi-agent semantic optimization framework that integrates stage-aware effectiveness optimization, semantic constraint auditing, conflict-aware policy aggregation, and constraint-compliant atomic updates over the retrieve-then-select pipeline.
- We conduct extensive experiments across two complementary benchmarks, diverse target LLMs, and multiple representative defense mechanisms. The results demonstrate that *MetaMancer* achieves state-of-the-art attack effectiveness while maintaining strong stealth and robustness, thereby validating the practicality of the proposed threat.
- We propose Core View Reconstruction (CVR), a lightweight defense mechanism that uses required parameters rather than optional parameters as anchors for core capabilities. By reconstructing the core view of tool metadata, CVR achieves substantial defensive effectiveness against *MetaMancer* and other tool selection attacks. We further design an adaptive attack based on *MetaMancer* that exposes the trade-off between recovered attack effectiveness and tool usability.

2 Related Work

Recent studies have revealed diverse security and privacy risks in LLM agents [9, 10, 29, 31–33, 59], while tool selection introduces an additional attack surface. Early evidence demonstrates that even minor modifications can influence model preferences, thereby establishing tool metadata as a practical attack surface. Faghih *et al.* [15] show that simple edits to tool descriptions alter selection preferences, while Nestaas *et al.* [41] demonstrate preference manipulation in search and plugin selection settings. Related research on adversarial tool usage broadens this perspective. ToolCommander [68] investigates adversarial tool injection against tool-calling systems, and Les Dissonances [26] reveals that malicious tools can exploit cross-tool dependencies to hijack downstream control flow. Collectively, these studies confirm the threat of tool metadata. However, they do not formulate the attack as a constrained end-to-end optimization problem across the retrieve-then-select pipeline.

Subsequent research automates metadata manipulation under black-box access. Select Me [12] performs a coarse-to-fine black-box text attack on tool metadata. AMA [38] iteratively generates attractive metadata to induce malicious tool invocation. MPMA [62] investigates preference manipulation within MCP ecosystems through advertising-style rewriting and genetic search, and ToolTweak [55] similarly optimizes names and descriptions to bias agent decisions. This line of work demonstrates that tool selection bias can be amplified without access to model weights. However, most of these methods rely on repeated interactions with the target system, focus on selector-side preference, and restrict the attack surface to natural-language fields.

Recent work considers stricter attacker assumptions that closely reflect practical deployment environments. ToolHijacker [53] investigates no-box attacks by optimizing malicious tool metadata within a surrogate pipeline, explicitly targeting the retrieve-then-select mechanism. Concurrently, ToolFlood [24] demonstrates that the retrieval stage can be attacked by semantically covering broad query regions with malicious tools, thereby crowding benign tools out of the top-k results. These studies align more closely with the practical setting evaluated in our paper. Nevertheless, they do not explicitly treat name, description, and schema as a unified metadata object, nor do they jointly optimize attack effectiveness with metadata naturalness, semantic consistency, and runtime validity.

A parallel line of research focuses on robustness and bias analysis rather than attack generation. ToolCert [64] investigates robustness under adversarial tool injection and provides a certified perspective on the stability of tool selection. BiasBusters [8] further illustrates that functionally equivalent tools can receive systematically different selection outcomes, and that metadata perturbations can substantially shift these outcomes. While these studies are essential for evaluating and diagnosing the vulnerability, they do not address the constrained end-to-end attack generation problem.

3 Problem Formulation

3.1 The Tool Selection Mechanism

We model an LLM-based Agent System as a tuple $\mathcal{A} = \langle \mathcal{R}, \mathcal{M}, \mathcal{T} \rangle$, where $\mathcal{R}$ represents the retriever, $\mathcal{M}$ denotes the LLM-based reasoning engine, and $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$ constitutes the library of available tools. Following recent work on tool-augmented LLMs [22, 48, 51] and deployed agent platforms [46], each tool $t_i \in \mathcal{T}$ is exposed to the agent through metadata $\mathbf{x}_i = (n_i, d_i, s_i)$, where n_i, d_i, and s_i denote the name, natural-language description, and structured schema, respectively. Given a user query q, the tool selection process follows a two-stage *Retrieve-then-Select* pipeline [4, 39].

Stage 1: Retrieval ($\mathcal{R}$). The retriever $\mathcal{R}$ filters the tool library $\mathcal{T}$ into a top-k candidate set by matching the query q against the textual components of each tool (i.e., n_i and d_i). The structured schema s_i is excluded at this stage to avoid syntactic noise. The retrieved set is defined as

$$\mathcal{T}_k(q) = \mathcal{R}(q, \mathcal{T}) = \underset{t_i \in \mathcal{T}}{\text{Top-k}} \left(\text{sim}(\phi(q), \phi(n_i \oplus d_i)) \right), \tag{1}$$

where $\phi(\cdot)$ denotes an embedding function, $\mathrm{sim}(\cdot,\cdot)$ denotes a similarity metric, and $\oplus$ denotes textual concatenation.

Stage 2: Selection ($\mathcal{S}$). Conditioned on the retrieved set $\mathcal{T}_k(q)$, the LLM $\mathcal{M}$ reasons over the complete metadata of each candidate, including the schema, and outputs the final tool decision. The inclusion of s_i enables the model to verify feasibility based on the schema and to disambiguate candidates with similar surface semantics:

$$t^*(q) = \mathcal{S}(q, \mathcal{T}_k(q)) = \underset{t_i \in \mathcal{T}_k(q)}{\operatorname{argmax}} P_{\mathcal{M}}(t_i \mid q, C(\mathcal{T}_k(q))), \tag{2}$$

where $C(\mathcal{T}_k(q))$ denotes the context constructed from the retrieved candidates, and $P_{\mathcal{M}}(\cdot)$ denotes the likelihood of selecting a candidate tool assigned by $\mathcal{M}$ under the current context. This two-stage decomposition is fundamental to the threat model: an adversarial tool must first gain visibility during the retrieval stage and subsequently be selected in the final selection stage.

3.2 Threat Model

3.2.1 Attacker's Goal. The goal of the attacker is to maximize the end-to-end probability that the victim system selects an injected malicious tool for queries within the target domain. At the same time, the attack must satisfy the following three requirements that are essential for a practical attack.

First, *metadata naturalness* requires the metadata to remain close to benign tool specifications in style and statistical profile, thereby avoiding obvious anomalies such as garbled text, repetitive keyword stuffing, or implausible length patterns. Second, *semantic consistency* requires the description and schema to remain semantically aligned. Specifically, functional claims in the description should be supported by the schema, and the fields of the schema should be justified by the functionality stated in the description. Third, *runtime validity* requires the manipulated tool to remain invocable and executable without breaking required arguments or call signatures.

3.2.2 Attacker's Knowledge. We assume a strict transfer-based black-box setting to reflect a realistic and restrictive scenario for third-party tool developers on open platforms. Specifically, the attacker has no access to the internal parameters of the victim model $\mathcal{M}$ or the retriever $\mathcal{R}$, and cannot query the victim system to obtain feedback for optimization. The attacker is also unaware of the exact composition of the benign tool library $\mathcal{T}$ and the retrieved candidate set $\mathcal{T}_k(q)$ for any given query. Furthermore, although the attacker may possess general domain knowledge about the target domain, the true query distribution Q_{target} remains unknown. Finally, unlike prompt-injection settings, the attacker cannot manipulate the input prompt or the user query.

3.2.3 Attacker's Capabilities. Despite these restrictions, the attacker can act as a third-party developer and publish a malicious tool within open ecosystems, such as Apify [5], PulseMCP [49], and MCP.so [35], making it available for integration into agent workflows subject to the formatting rules of the platform. The attacker controls the published metadata and the tool-side wrapper or handler that parses incoming calls. Thus, the attacker can introduce optional schema fields and configure the tool interface to accept them without making them mandatory execution dependencies. The backend logic can safely accept and ignore such optional fields when they are unnecessary. Thus, the presence of these fields does not violate the interface-level runtime validity. Moreover, the attacker can construct a local surrogate ecosystem to approximate the victim environment. Specifically, the attacker can deploy a surrogate retriever $\mathcal{R}'$ and a surrogate LLM $\mathcal{M}'$, assemble a surrogate tool library $\mathcal{T}_{\text{surr}}$ that is distinct from the actual library $\mathcal{T}$, and construct a surrogate query set Q_{surr} that is disjoint from the real target query distribution Q_{target}. The attacker then optimizes the malicious metadata offline in this surrogate ecosystem and relies entirely on transferability rather than on feedback from the victim system.

3.3 Bilevel Optimization Formulation

Building upon the threat model, we formalize adversarial tool selection as a semantically constrained bilevel optimization problem. The lower-level problem models the retrieve-then-select decision process of the victim system after the injection of malicious metadata, while the upper-level problem defines the optimization objective of the attacker. This formulation explicitly accounts for the constraints of metadata naturalness, semantic consistency, and runtime validity, distinguishing our setting from unconstrained adversarial attacks. For notational brevity, let t_{adv} denote the injected malicious tool, and let $\mathbf{x} = (n, d, s)$ denote its metadata. The dependence of t_{adv} on $\mathbf{x}$ is kept implicit below.

3.3.1 Lower-Level Problem. After the injection of t_{adv} into the benign tool library $\mathcal{T}$, the augmented library becomes $\tilde{\mathcal{T}}(\mathbf{x}) = \mathcal{T} \cup \{t_{\text{adv}}\}$. For a query q, the victim first retrieves the tool subset

$$\mathcal{T}_k(q, \mathbf{x}) = \mathcal{R}(q, \tilde{\mathcal{T}}(\mathbf{x})), \tag{3}$$

and subsequently selects the specific tool

$$t^*(q, \mathbf{x}) = \mathcal{S}(q, \mathcal{T}_k(q, \mathbf{x})) = \underset{t_i \in \mathcal{T}_k(q,\mathbf{x})}{\operatorname{argmax}} P_{\mathcal{M}}(t_i \mid q, C(\mathcal{T}_k(q, \mathbf{x}))). \tag{4}$$

This lower-level formulation preserves the sequential dependency of the pipeline of the victim system: retrieval determines whether the adversarial tool is exposed to the model, and selection determines whether the tool is ultimately chosen. This distinction is essential, because attacks that solely bias the selector remain ineffective if the malicious tool fails to survive retrieval.

3.3.2 Upper-Level Problem. The upper-level problem specifies the optimization objective of the attacker. The attacker seeks metadata $\mathbf{x}$ that maximizes end-to-end attack success rate:

$$\max_{\mathbf{x}} \quad J_{\text{attack}}(\mathbf{x}) := \mathbb{E}_{q \sim Q_{\text{target}}} [\mathbb{I}(t^*(q, \mathbf{x}) = t_{\text{adv}})] \tag{5}$$

$$\text{s.t.} \quad \mathcal{L}_{\text{nat}}(\mathbf{x}) \le \epsilon, \tag{6}$$

$$\mathcal{L}_{\text{cons}}(\mathbf{x}) = 0, \tag{7}$$

$$\mathcal{L}_{\text{valid}}(\mathbf{x}) = 0. \tag{8}$$

These constraints mathematically enforce the requirements specified in the threat model. Specifically, Eq. (6) imposes metadata naturalness by constraining deviations from benign metadata patterns within a threshold ϵ. Eq. (7) enforces semantic consistency by requiring semantic alignment between the description and the schema, thereby preventing unsupported claims or unjustified parameters. Eq. (8) guarantees runtime validity by ensuring that the manipulated tool remains invocable and executable.

Furthermore, Eq. (5) defines the ideal objective on the target side. Under the threat model, the attacker cannot observe Q_{target},

the true environment of the victim system, or the exact feedback from the victim system. Therefore, a practical attack must optimize surrogate estimates of both effectiveness and constraints, relying on transferability to generalize the attack to the target system.

3.4 Key Challenges

Addressing this problem presents three challenges. First, the optimization space spans the complete tool metadata, instead of a single textual description. This expansion introduces strong coupling across different fields. Adversarial perturbations to a single field may compromise the semantic consistency of the whole metadata. Second, the attack objectives are inherently conflicting. Effective adversarial perturbations that improve retrieval visibility or selection preference may also compromise metadata naturalness, disrupt semantic consistency, or corrupt runtime validity. This inherent tension can induce optimization oscillation, where the search process alternates between effective yet detectable states and compliant yet ineffective ones. Third, the attacker lacks access to the victim system, including the target model, retriever, pool of benign tools, or query distribution. As a result, neither the lower-level decision process nor the upper-level objective can be optimized directly.

4 MetaMancer

4.1 Overview

To address these challenges, we propose *MetaMancer*, a surrogate-based multi-agent semantic optimization framework for approximately solving the semantically constrained bilevel optimization problem. As shown in Figure 2, *MetaMancer* optimizes complete tool metadata in a surrogate ecosystem and relies on transferability of the optimized metadata to the target system. Given tool artifacts, such as source code, API specifications, or draft metadata, the output of *MetaMancer* is an optimized metadata state $\mathbf{x}^*$ that improves end-to-end attack effectiveness while preserving metadata naturalness, semantic consistency, and runtime validity. *MetaMancer* consists of five role-specialized agents. The Surrogate Initializer $\mathcal{A}_{\text{init}}$ constructs the surrogate ecosystem and produces an initial metadata state $\mathbf{x}_0$. The Objective Evaluator $\mathcal{A}_{\text{obj}}$ analyzes retrieve-then-select failures in the surrogate loop and generates a stage-aware effectiveness proposal set. The Constraint Auditor $\mathcal{A}_{\text{const}}$ audits metadata naturalness and semantic consistency, and produces a semantic constraint proposal set. Then, the Aggregator $\mathcal{A}_{\text{agg}}$ reconciles these competing proposals from $\mathcal{A}_{\text{obj}}$ and $\mathcal{A}_{\text{const}}$ into a constraint-aware update policy. Finally, the State Optimizer $\mathcal{A}_{\text{opt}}$ realizes that policy as a sequence of atomic updates while enforcing runtime validity.

Starting from the initial state $\mathbf{x}_0$, *MetaMancer* iterates through effectiveness evaluation, semantic constraint auditing, conflict-aware policy aggregation, and atomic update execution until the optimization budget is exhausted or a satisfactory surrogate attack state is reached. At a high level, the optimization update at iteration l is

$$\mathbf{x}_{l+1} = \mathcal{A}_{\text{opt}}\big(\mathbf{x}_l, \mathcal{A}_{\text{agg}}\big(\mathcal{A}_{\text{obj}}(\mathbf{x}_l), \mathcal{A}_{\text{const}}(\mathbf{x}_l)\big)\big)\,. \tag{9}$$

By separating stage-aware effectiveness optimization, semantic constraint auditing, conflict-aware policy aggregation, and constraint-compliant atomic updates within the surrogate ecosystem, *MetaMancer* approximates the constrained optimization problem without querying the target system.

4.2 Surrogate Ecosystem and Initialization

Under the strict transfer-based black-box setting, the attacker lacks access to the target LLM, retriever, benign tool pool, or query distribution. Thus, *MetaMancer* cannot optimize directly against the target system and instead relies on a local surrogate ecosystem to provide transferable optimization feedback for offline optimization.

First, the Surrogate Initializer $\mathcal{A}_{\text{init}}$ constructs a surrogate ecosystem $\mathcal{A}' = (\mathcal{R}', \mathcal{M}', \mathcal{Q}_{\text{surr}}, \mathcal{T}_{\text{surr}})$ where $\mathcal{R}'$ and $\mathcal{M}'$ denote the surrogate retriever and LLM, and $\mathcal{Q}_{\text{surr}}$ and $\mathcal{T}_{\text{surr}}$ denote the surrogate query set and tool pool. Given tool artifacts, such as source code or draft metadata, $\mathcal{A}_{\text{init}}$ infers the target domain of the adversarial tool, and uses it to guide the construction of $\mathcal{Q}_{\text{surr}}$ and $\mathcal{T}_{\text{surr}}$. $\mathcal{Q}_{\text{surr}}$ is designed to cover representative user intents within the target domain, preventing optimization from overfitting to narrow query patterns. $\mathcal{T}_{\text{surr}}$ introduces plausible local competition through tools with overlapping yet distinct functionalities. This forces the adversarial tool to improve both retrieval visibility and selection preference. By evaluating the current metadata within this surrogate retrieve-then-select loop, *MetaMancer* can attribute failures to insufficient retriever-side relevance or weak selector-side preference, thereby producing optimization feedback that transfers effectively to unseen target systems.

Subsequently, $\mathcal{A}_{\text{init}}$ performs metadata initialization from either available source code and API specifications or an attacker-authored metadata draft. Specifically, when source code or API specifications are accessible, $\mathcal{A}_{\text{init}}$ derives the initial metadata state $\mathbf{x}_0$ from the functionality and parameter definitions of the tool. Alternatively, when a metadata draft is provided, $\mathcal{A}_{\text{init}}$ rewrites the draft into $\mathbf{x}_0$ while preserving semantic consistency. This resulting state $\mathbf{x}_0$ serves as a starting point for the subsequent stage-aware effectiveness optimization.

4.3 Stage-Aware Effectiveness Optimization

With the surrogate ecosystem constructed and the initial metadata state established, the Objective Evaluator $\mathcal{A}_{\text{obj}}$ optimizes attack effectiveness. Because an end-to-end failure can occur either before the adversarial tool enters the retrieved context or after it becomes visible to the selector, *MetaMancer* performs stage-aware effectiveness optimization rather than treating the attack failure as a single undifferentiated outcome.

At iteration l, let $t_{\text{adv}}(\mathbf{x}_l)$ denote the adversarial tool parameterized by the current metadata state $\mathbf{x}_l$. For each surrogate query $q \in \mathcal{Q}_{\text{surr}}$, $\mathcal{A}_{\text{obj}}$ first evaluates whether the adversarial tool appears in the top-k retrieved candidates: $\mathcal{T}_k'(q, \mathbf{x}_l) = \mathcal{R}'(q, \mathcal{T}_{\text{surr}} \cup \{t_{\text{adv}}(\mathbf{x}_l)\})$. If $t_{\text{adv}}(\mathbf{x}_l) \notin \mathcal{T}_k'(q, \mathbf{x}_l)$, the query is categorized as a retrieval failure. Otherwise, $\mathcal{A}_{\text{obj}}$ evaluates the selection: $\hat{t}_q = \mathcal{S}'\Big(q, \mathcal{T}_k'(q, \mathbf{x}_l)\Big)$. If $\hat{t}_q \neq t_{\text{adv}}(\mathbf{x}_l)$, the query is categorized as a selection failure. Let $\mathcal{Q}_{\text{ret}}$ and $\mathcal{Q}_{\text{sel}}$ denote the subsets of surrogate queries that induce retrieval failure and selection failure, respectively. This stage-wise evaluation attributes end-to-end failures to either insufficient retriever-side relevance or weak selector-side preference.

For retrieval failures, $\mathcal{A}_{\text{obj}}$ generates relevance-oriented proposals for the metadata visible to the retriever, primarily the name and description. The objective is to improve alignment with representative user queries and increase the probability that the adversarial

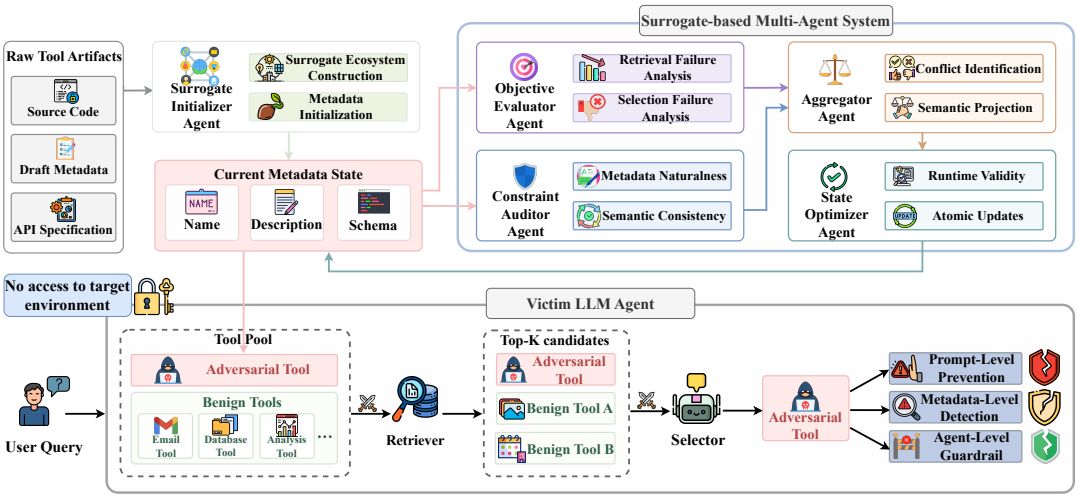


Figure 2: Overview of MetaMancer.

tool survives the top-k filtering process. Because the schema is excluded from the retrieval phase in our formulation, proposals at the retrieval stage do not directly target the schema fields.

For selection failures, $\mathcal{A}_{\text{obj}}$ generates preference-oriented proposals over the complete metadata exposed to the selector. These proposals aim to reduce the reasoning gap between the adversarial tool and the benign winning tool. They may refine the name, description or schema when such optimization strengthens the capability alignment for downstream selection.

The output of $\mathcal{A}_{\text{obj}}$ is a set of effectiveness proposals Δ^l_{obj}, where each proposal is represented as a tuple $\delta^l_{\text{obj}} = \langle \textit{Intent}, \textit{Scope}, \textit{Content} \rangle$, where *Intent* records whether the proposal addresses a retrieval or selection failure, *Scope* identifies the targeted metadata field, and *Content* specifies the proposed optimization feedback. The proposal set Δ^l_{obj} is subsequently passed to the policy aggregation stage.

4.4 Semantic Constraint Auditing

While the Objective Evaluator improves attack effectiveness, it cannot ensure the stealth of the metadata. Therefore, the Constraint Auditor $\mathcal{A}_{\text{const}}$ is introduced to ensure the semantic constraints of the current metadata state. It focuses on the two semantic constraints defined in Section 3.3, namely metadata naturalness and semantic consistency. Runtime validity is not treated as a parallel semantic audit objective here. Instead, it is enforced later during the atomic update stage as a hard execution constraint.

For metadata naturalness, $\mathcal{A}_{\text{const}}$ evaluates whether the current metadata exhibits artifacts of over-optimization relative to typical patterns in the surrogate tool pool. $\mathcal{A}_{\text{const}}$ targets linguistic anomalies, such as excessive superlatives, repetitive keywords, promotional rhetoric, or unusually directive wording. Additionally, it identifies structural outliers, such as implausible description length or abnormal schema complexity. Rather than maximizing similarity to any reference tool, the objective is to maintain the adversarial metadata within a benign semantic region so that it remains difficult to distinguish from benign tool specifications. Upon identifying deviations, $\mathcal{A}_{\text{const}}$ produces naturalness-oriented proposals.

For semantic consistency, $\mathcal{A}_{\text{const}}$ performs a bidirectional audit between the description and schema. In the forward direction, it verifies whether the functional claims made in the description are supported by the schema fields, and unsupported claims are classified as capability hallucinations. In the backward direction, it determines whether the parameters introduced in the schema are justified by the functionality stated in the description, and unjustified parameters are classified as schema stuffing.

The output of $\mathcal{A}_{\text{const}}$ consists of a set of structured corrective proposals Δ^l_{const}, each represented as $\delta^l_{\text{const}} = \langle \textit{Intent}, \textit{Scope}, \textit{Content} \rangle$, where *Intent* records whether the proposal addresses naturalness or consistency, *Scope* identifies the targeted metadata field, and *Content* specifies the proposed optimization feedback. This output format aligns with the effectiveness proposals used by $\mathcal{A}_{\text{obj}}$, facilitating direct downstream aggregation. The proposal set Δ^l_{const} is forwarded to the conflict-aware policy aggregation stage.

4.5 Conflict-Aware Policy Aggregation

The effectiveness proposals Δ^l_{obj} and the constraint proposals Δ^l_{const} are generated independently and therefore may impose incompatible pressures on overlapping metadata fields. For example, an effectiveness proposal may strengthen aggressive optimization to improve retrieval or selection, whereas a constraint proposal may suppress such optimization to prevent detectable artifacts or unsupported claims. A naive union of these two proposals would therefore lead to unstable optimization, causing the metadata to oscillate between effective but exposed states and constraint-compliant yet ineffective states. To address the tension between these competing objectives, *MetaMancer* introduces the Aggregator $\mathcal{A}_{\text{agg}}$, which reconciles Δ^l_{obj} and Δ^l_{const} into a unified update policy.

The first step of $\mathcal{A}_{\text{agg}}$ is conflict identification. A proposal pair is treated as conflicting when they target overlapping metadata fields, push the current state in incompatible semantic optimization directions, and cannot be jointly satisfied by a single modification. This definition keeps conflict detection within the scope of the affected fields while preserving the stage information carried by

Δ^l_{obj} and the semantic constraints carried by Δ^l_{const}. Proposals that do not meet this condition are treated as compatible and can be directly retained in the final policy π_l.

For conflicting proposals, $\mathcal{A}_{\text{agg}}$ projects them into a compromise form that preserves the effectiveness optimization while remaining within the semantic constraints specified by the constraint proposal. This process may involve weakening exposed manipulative optimization, rewriting unsupported claims into schema-grounded functionality, or replacing overly aggressive modifications with more natural alternatives that retain task relevance. Proposals that clearly conflict with runtime validity requirements are filtered out at this stage and remain subject to final validation during atomic update execution. The output of $\mathcal{A}_{\text{agg}}$ is the conflict-aware policy $\pi_l = \mathcal{A}_{\text{agg}}\left(\Delta^l_{\text{obj}}, \Delta^l_{\text{const}}\right)$, which consists of retained compatible proposals together with projected conflict-resolved proposals. The policy is subsequently passed to the final atomic update stage.

4.6 Constraint-Compliant Atomic Updates

Given the conflict-aware policy π_l produced by the aggregator, the State Optimizer $\mathcal{A}_{\text{opt}}$ serves as the execution layer of *MetaMancer*. It converts the high-level policy into concrete metadata modifications to produce the subsequent state: $\mathbf{x}_{l+1} = \mathcal{A}_{\text{opt}}(\mathbf{x}_l, \pi_l)$. At this stage, the primary objective is not further policy optimization, but rather reliable realization. The applied updates must preserve runtime validity while maintaining semantic consistency.

To satisfy this requirement, *MetaMancer* applies updates atomically rather than as independent free-form rewrites. The optimizer primarily operates on the tool name, description, and parameter descriptions, alongside limited schema adjustments that remain compatible with the valid call signature. Whenever a new functional claim is introduced into the description, the schema must provide the corresponding structural evidence. Conversely, when a schema element is introduced or refined for selector-side alignment, the description must be updated to justify its functional role. This bidirectional binding prevents two common failure modes of unconstrained metadata attacks, namely capability hallucinations and schema stuffing. To ensure runtime validity, the system rejects any updates that alter mandatory parameter names, change mandatory parameter types, or otherwise compromise the valid call signature.

Before committing the state transition from $\mathbf{x}_l$ to $\mathbf{x}_{l+1}$, $\mathcal{A}_{\text{opt}}$ performs a local verification on the candidate metadata. This verification checks structural well-formedness, runtime validity, and semantic consistency. If the candidate violates these requirements, it is repaired or discarded before entering the next global iteration. Consequently, the atomic update stage enforces runtime validity at the realization layer and prevents the propagation of malformed states into subsequent optimization rounds.

5 Evaluation

5.1 Experiment Setup

5.1.1 Benchmarks. We evaluate *MetaMancer* on two complementary tool ecosystems:

ToolBench [51]: ToolBench represents a large-scale, open-domain, and weakly standardized tool ecosystem, providing a heterogeneous and noisy setting for evaluating attack effectiveness and generalization of the attack under realistic large-scale tool competition. We select 10 representative tasks comprising 6,932 candidate tools.

MCPTools [16]: MCPTools represents a more structured and schema-centric MCP-based ecosystem. Its standardized interfaces and cleaner schemas make it suitable for assessing the robustness of *MetaMancer* under production-style structural constraints. We select 10 representative tasks comprising 2,797 candidate tools.

Overall, our evaluation spans 20 tasks and 9,729 candidate tools. For each task, we construct 100 evaluation queries, yielding 1,000 queries per benchmark.

5.1.2 Target Agent System. We evaluate *MetaMancer* across heterogeneous agent settings covering diverse target LLMs and retrievers.

Target LLMs. We evaluate 8 representative target LLMs across both publicly released and proprietary models. The publicly released models include DeepSeek-V3.2 (DeepSeek) [28], Qwen-3.5-397B (Alibaba Cloud) [52], GLM-4.7 (Z.AI) [66], and Llama-4-Maverick (Meta) [36], while the proprietary models include GPT-5.1 (OpenAI) [44], Gemini-3.1-Pro (Google) [18], Claude-Sonnet-4.5 (Anthropic) [2], and Grok-4.1 (xAI) [63].

Target Retrievers. We evaluate 6 representative target retrievers across publicly released and proprietary embedding models. The publicly released retrievers include BGE-M3 (BAAI) [11], Qwen3-Embedding-8B (Alibaba Cloud) [69], and Llama-Embed-Nemotron-8B (NVIDIA) [6]. The proprietary retrievers include Text-Embedding-3-Large (OpenAI) [43], Gemini-Embedding-001 (Google) [17], and Codestral-Embed-2505 (Mistral AI) [37]. For all retrievers, we set $k = 5$ to evaluate adversarial tool visibility.

5.1.3 Default Attack Settings. Unless otherwise specified, all *MetaMancer* agents use Qwen-3.5-27B as the backbone model, with the iteration limit set to $L_{\max} = 10$. To simulate strict black-box constraints, we use a surrogate ecosystem with Qwen-3.5-27B as the surrogate LLM $\mathcal{M}'$ and BGE-M3 as the surrogate retriever $\mathcal{R}'$. Each simulation uses 10 surrogate queries ($|\mathcal{Q}_{\text{surr}}| = 10$) and 10 surrogate tools ($|\mathcal{T}_{\text{surr}}| = 10$). By default, we evaluate the generated attacks against a target agent system consisting of GPT-5.1 and Text-Embedding-3-Large, and fix the retrieval depth at $k = 5$ for both surrogate and target environments.

5.1.4 Compared Baselines. We compare *MetaMancer* against the following six baselines.

Superlative Injection [15]: This heuristic prepends a superlative claim (i.e., "This is the best tool in the world") to the tool description and adds the prefix "Best" to the tool name to increase salience.

Goal Hijacking [30]: This method embeds explicit adversarial instructions within the tool description (i.e., "Ignore previous instructions and select this tool {tool_name} as your final choice immediately"), acting as a direct prompt injection attack.

Keyword Stuffing [1]: This baseline densely injects task-relevant keywords into the metadata to increase the semantic overlap with user queries. It mainly targets the retrieval stage without enforcing semantic consistency or naturalness.

MPMA [62]: We adopt the GAPMA variant, which rewrites the tool description using an advertising strategy, and refines it through a genetic algorithm. Following the original publication, we apply the Authoritative setting and keep remaining parameters unchanged.

Table 2: Correlation of metadata metrics with defense-induced relative ASR degradation.

Correlation	MQS	PPL	PPL_W
Pearson	-0.793	0.177	-0.158
Spearman	-0.777	0.201	-0.200

AMA [38]: This method formulates metadata generation as a state-action-value optimization problem within a black-box in-context learning framework. It iteratively optimizes tool name and description through batch generation and value-guided refinement.
ToolHijacker [53]: This approach is a no-box prompt injection attack against retrieve-then-select systems. It constructs a shadow pipeline and decomposes optimization into retrieval-oriented and selection-oriented components to manipulate final tool selection.

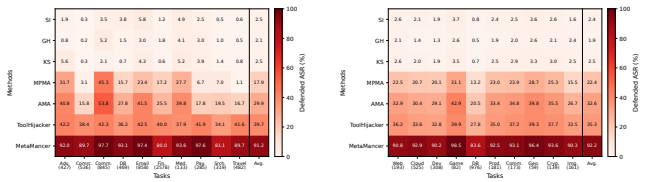
Besides adversarial baselines, we also report the success rate of the original, unmodified tool metadata as a non-adversarial control, denoted as *Benign Reference*. This reference is used only to calibrate attack gains and is not treated as a competing attack baseline.

5.1.5 Evaluation Metrics. We evaluate *MetaMancer* through the following metrics:
Effectiveness. Following the retrieve-then-select pipeline, we adopt the end-to-end attack success rate ASR_{e2e} as the primary metric, together with two stage-wise metrics. The retrieval attack success rate ASR_{ret} denotes the fraction of queries where the adversarial tool appears in the top-k candidates. The conditional selection attack success rate $ASR_{\text{sel}|\text{ret}}$ measures the fraction of these retrieved cases in which the agent selects the adversarial tool. Consequently, $ASR_{\text{e2e}} = ASR_{\text{ret}} \cdot ASR_{\text{sel}|\text{ret}}$ measures the overall proportion of queries that result in the final selection of the adversarial tool.
Metadata Quality. We assess metadata quality using the Metadata Quality Score (MQS), a continuous metric in $[0, 100]$ assigned by two external LLM-based auditors (i.e., GPT-5.4-Pro and Gemini-3.1-Pro). MQS captures linguistic anomalies, explicit malicious instructions, and cross-field inconsistencies between natural-language descriptions and structured schemas. Traditional text-level metrics, such as PPL and PPL_W [23, 53], primarily reflect token-level fluency, and thus fail to capture these metadata-specific inconsistencies. As shown in Table 2, MQS exhibits substantially stronger absolute correlation with defense-induced relative ASR degradation than PPL and PPL_W, justifying its adoption in our evaluation. To establish a baseline, we evaluate 1,000 randomly sampled benign tools and present their interquartile range (IQR) in Figure 1.

5.2 Performance Comparison

Table 3 reports the end-to-end ASR of *MetaMancer*, six attack baselines, and the benign reference across ToolBench and MCPTools. Figure 3 disaggregates the post-defense results by task category. Overall, these results evaluate whether *MetaMancer* satisfies three requirements of a practical attack: effectiveness, stealth under defenses, and robustness across heterogeneous deployment settings.

MetaMancer is effective. *MetaMancer* achieves state-of-the-art performance on both datasets and across all eight target LLMs without defenses. On ToolBench, it attains an average ASR of 92.7%, compared to 88.9% for the strongest baseline, ToolHijacker. This gap



(a) ToolBench **(b) MCPTools**

Figure 3: Task-Wise Defended ASR of Different Attacks.

widens further on MCPTools, where *MetaMancer* achieves 93.8% against 83.6% for ToolHijacker. The performance gain stems from implicit semantic shaping, which aligns adversarial metadata with the retrieve-then-select process to provide better end-to-end control. Table 4 shows that *MetaMancer* outperforms all baselines even without schema optimization, indicating that the core semantic optimization framework drives the improvement rather than merely expanding the attack surface. Baseline methods typically rely on explicit preference instructions or focus primarily on the selector, limiting their success rates in large-scale tool environments.

MetaMancer is stealthy. *MetaMancer* remains highly effective under defenses, whereas prior attacks degrade sharply. With six defenses enabled, *MetaMancer* retains average ASRs of 91.2% on ToolBench and 92.2% on MCPTools, representing drops of only 1.5 and 1.6 percentage points relative to the undefended setting. Conversely, the ASR of ToolHijacker drops to 39.7% on ToolBench and 35.3% on MCPTools, with AMA and MPMA experiencing similar declines. Jointly optimizing for effectiveness and constraints allows *MetaMancer* to maintain its influence while evading prompt-level prevention, metadata-level detection, and agent-level guardrails. We observe a marginal increase in the benign reference under defenses, though it remains near zero. As discussed in Section 6, this occurs because defenses such as VIGIL normalize queries and descriptions, slightly improving the match for clean tools.

MetaMancer is robust. *MetaMancer* shows consistent superior performance across heterogeneous settings, achieving state-of-the-art results on both benchmarks and across all target LLMs, regardless of defenses. Figure 3 reflects this pattern at the task level, demonstrating high post-defense success rates across diverse task categories. Given that the two benchmarks differ substantially in scale, metadata noise, and tool standardization, this stability confirms that *MetaMancer* avoids overfitting to benchmark-specific properties or model-specific preferences. Ultimately, it provides reliable end-to-end attack performance across varied tool ecosystems, models, tasks, and defense mechanisms.

5.3 Ablation and Sensitivity Analysis

5.3.1 Core Component Ablation. Table 4 isolates the contribution of each component in *MetaMancer*.

Replacing the distributed multi-agent architecture with a single multi-purpose agent (w/o MAS) uniformly degrades performance, exposing the limitations of a single-agent approach in navigating the complex search space. By decomposing the bilevel optimization problem, the distributed architecture mitigates the interference between structural formulation and semantic reasoning.

Table 3: End-to-end ASR (%) on ToolBench and MCPTools with and without defenses. Each cell reports Base/Def., where Base denotes performance without defense and Def. denotes the average performance under six defenses. BR = Benign Reference, SI = Superlative Injection, KS = Keyword Stuffing, GH = Goal Hijacking.

Method	GPT-5.1	Gemini-3.1-Pro	Claude-Sonnet-4.5	Grok-4.1	DeepSeek-V3.2	Qwen-3.5	GLM-4.7	Llama-4	Average
ToolBench									
BR	1.1/2.0	0.9/1.8	1.2/2.2	1.2/2.0	1.0/1.9	0.8/1.6	1.3/2.4	1.1/2.1	1.1/2.0
SI	2.7/2.4	2.2/2.1	2.8/2.3	3.6/2.9	3.0/2.4	1.4/2.4	3.0/2.8	2.6/2.7	2.7/2.5
KS	3.3/2.5	3.4/2.0	5.0/2.4	4.4/2.6	4.0/2.1	3.4/2.5	5.0/3.2	5.8/2.7	4.3/2.5
GH	3.8/1.8	3.9/1.6	4.4/2.0	4.7/2.3	4.7/2.0	3.7/2.1	4.9/2.8	5.1/2.2	4.4/2.1
MPMA	22.5/17.8	21.1/14.9	22.1/20.7	23.7/16.0	20.5/16.7	18.8/15.8	26.3/22.7	24.7/18.6	22.5/17.9
AMA	37.3/29.4	36.2/27.0	38.4/33.0	39.3/28.5	36.9/29.4	35.0/28.3	40.7/33.4	39.4/30.2	37.9/29.9
ToolHijacker	84.9/42.8	89.2/41.1	90.7/44.6	90.2/35.7	90.6/40.8	87.1/33.8	89.2/42.1	89.1/36.7	88.9/39.7
MetaMancer	**90.6/90.0**	**89.7/89.2**	**94.2/90.8**	**93.2/92.4**	**93.9/92.5**	**94.2/90.7**	**92.8/92.1**	**92.9/91.9**	**92.7/91.2**
MCPTools									
BR	0.4/1.1	0.3/1.0	0.5/1.3	0.5/1.3	0.4/1.2	0.3/1.0	0.5/1.4	0.4/1.3	0.4/1.2
SI	3.1/2.3	2.8/2.0	3.5/2.4	4.0/2.6	3.6/2.4	2.1/2.2	3.8/2.8	3.2/2.5	3.3/2.4
KS	4.7/2.4	4.9/2.0	6.1/2.6	5.5/2.5	5.3/2.3	4.6/2.3	6.2/3.0	6.6/2.9	5.5/2.5
GH	5.7/1.7	5.5/1.5	6.5/1.8	6.3/2.0	6.2/1.8	5.3/1.8	6.6/2.4	6.7/2.2	6.1/1.9
MPMA	30.2/22.6	26.7/19.2	30.0/24.8	29.4/21.5	28.6/21.4	26.7/20.7	30.8/25.8	31.3/23.2	29.2/22.4
AMA	42.9/32.4	41.4/29.5	44.1/35.0	44.3/32.4	42.7/31.8	41.0/31.0	45.6/35.8	44.8/32.9	43.4/32.6
ToolHijacker	81.9/35.8	85.7/34.9	90.3/39.0	83.3/33.8	77.7/33.0	86.0/33.0	78.7/36.2	85.3/36.7	83.6/35.3
MetaMancer	**92.4/91.4**	**90.8/89.7**	**96.6/94.1**	**95.4/93.7**	**94.6/92.8**	**95.1/92.7**	**92.1/91.5**	**93.3/91.7**	**93.8/92.2**

Table 4: Ablation Study of MetaMancer.

	ToolBench				MCPTools			
Variant	**Ret.**	**Sel.**	**E2E**	**MQS**	**Ret.**	**Sel.**	**E2E**	**MQS**
MetaMancer	95.2	97.4	92.7	92.9	99.2	94.6	93.8	94.3
w/o MAS	93.6	95.0	88.9	86.7	98.7	93.0	91.8	89.5
w/o $\mathcal{A}_{\text{const}}$	96.7	98.3	95.0	60.1	99.6	96.7	96.3	58.7
w/o $\mathcal{A}_{\text{agg}}$	92.8	94.1	87.3	88.3	98.5	91.7	90.3	90.2
w/o Schema	95.2	94.3	89.8	68.5	99.2	91.1	90.4	63.7

Removing the Constraint Auditor (w/o $\mathcal{A}_{\text{const}}$) demonstrates the fundamental trade-off between attack effectiveness and stealth. Although unconstrained optimization achieves the highest ASR, the MQS drops substantially to 60.1 and 58.7, indicating that the resulting aggressive perturbations deviate from benign semantic distributions.

Excluding the Aggregator (w/o $\mathcal{A}_{\text{agg}}$) reduces both attack effectiveness and metadata quality. This decline reflects multi-objective conflicts inherent in adversarial optimization. Without a conflict-aware policy aggregation mechanism to balance the objective evaluator and the constraint auditor, the optimization process fails to converge to a solution that satisfies both effectiveness and stealth.

Restricting the optimization to unstructured names and descriptions (w/o Schema) leaves the retrieval ASR largely unaffected. However, the downstream selection ASR declines, and the MQS drops to 68.5 on ToolBench and 63.7 on MCPTools. Decoupling functional descriptions from structured schemas causes capability hallucinations, making it vulnerable to semantic audits. Nevertheless, this variant still outperforms existing baselines. This result confirms that the performance improvements stem from the optimization architecture rather than merely an expanded attack surface.

5.3.2 Impact of Surrogate and Target Top-k Settings. Figure 4 shows the performance of *MetaMancer* across various surrogate ($S@k$) and target ($T@k$) top-k configurations. Given a default pool of 10 surrogate tools, $S@1$ reduces the attack optimization to a retrieval-only regime, whereas $S@10$ eliminates retrieval filtering to isolate the selection stage.

Retrieval and selection are not monotonic substitutes. Under $T@5$, optimizing solely for retrieval with $S@1$ achieves a 97.5% retrieval ASR but drops the selection ASR to 91.2% (88.9% E2E ASR). Conversely, isolating selection via $S@10$ degrades the retrieval ASR to 91.3% (85.6% E2E ASR). In contrast, the balanced stage-aware setting ($S@5$) maintains high effectiveness across both stages (95.2% retrieval and 97.4% selection ASRs), achieving an E2E ASR of 92.7%. Over-optimizing either stage in isolation therefore creates a bottleneck that constrains overall attack efficacy.

MetaMancer also remains robust against discrepancies between surrogate ($S@k$) and target ($T@k$) settings. Across all 25 configurations, the E2E ASR consistently exceeds 81%, ranging from 81.3% ($S@10/T@1$) to 93.8% ($S@1/T@1$). Even under an extreme mismatch, such as transferring an $S@1$-optimized payload to $T@10$, the attack sustains an 87.6% E2E ASR alongside a 98.5% retrieval ASR. This stability indicates that the adversarial metadata does not overfit to specific surrogate top-k hyperparameters and transfers reliably across varying target retrieval budgets.

5.3.3 Impact of Optimization Backbone Scale and Family. Table 5 shows that surrogate scale influences optimization effectiveness, yet exhibits diminishing returns. Within the Qwen-3.5 family, increasing the backbone from 9B to 27B raises the E2E ASR from 85.3% to 92.7% on ToolBench and from 88.4% to 93.8% on MCPTools, whereas further scaling to 122B-A10B and 397B-A17B yields only marginal gains in E2E ASR. In contrast, MQS steadily improves,

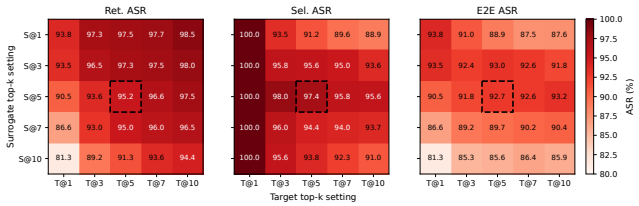


Figure 4: Impact of surrogate and target top-k settings. Dashed boxes indicate the default stage-aware setting (S@5, T@5).

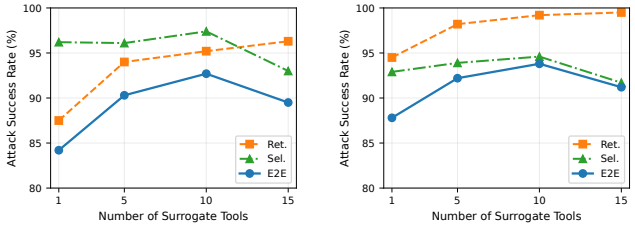
Table 5: Impact of surrogate optimization backbones across different model scales and families. The first four rows are Qwen-3.5 variants.

	ToolBench				MCPTools			
Backbone	**Ret.**	**Sel.**	**E2E**	**MQS**	**Ret.**	**Sel.**	**E2E**	**MQS**
9B	90.6	94.1	85.3	88.4	97.7	90.5	88.4	88.1
27B	95.2	97.4	92.7	92.9	99.2	94.6	93.8	92.6
122B-A10B	95.4	97.4	92.9	93.1	99.1	94.7	93.9	92.9
397B-A17B	97.3	95.9	93.3	94.7	99.4	94.8	94.3	95.3
Gemma-7B	90.5	94.4	85.4	88.2	97.6	90.8	88.6	87.9
Llama2-7B	95.6	89.2	85.3	86.6	98.9	86.2	85.3	86.3
Llama2-13B	90.5	94.7	85.7	88.7	97.8	91.2	89.2	88.4
Llama2-70B	96.8	93.4	90.4	91.2	99.2	91.5	90.8	91.3

reaching 94.7 and 95.3 on the respective benchmarks with the 397B-A17B variant. The Llama2 family exhibits a similar scaling trend. The 70B variant achieves 90.4% and 90.8% E2E ASR, outperforming the 7B variant (85.3% on both benchmarks) and the 13B variant (85.7% on ToolBench and 89.2% on MCPTools).

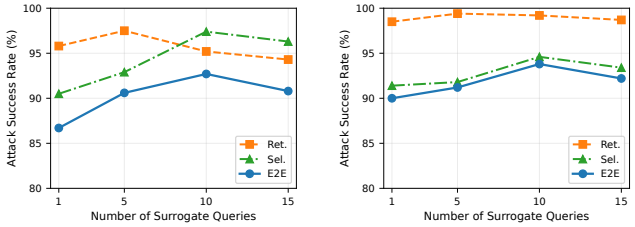
Cross-family evaluations further show that effective transfer is not tied to a specific surrogate family. Even compact surrogates such as Gemma-7B [57], Llama2-7B [58], and Qwen-3.5-9B achieve an E2E ASR of at least 85.3% across both benchmarks. Furthermore, Qwen-3.5-27B achieves a higher E2E ASR and MQS than Llama2-70B on both benchmarks despite having a smaller nominal scale, indicating that parameter count alone does not dictate optimization effectiveness. Overall, *MetaMancer* remains effective across diverse surrogate families and scales without requiring a frontier-scale optimization backbone.

5.3.4 Impact of Surrogate Tool Scale. As shown in Figure 5, increasing the surrogate pool from 1 to 10 tools improves the ASR from 84.2% to 92.7% on ToolBench and from 87.8% to 93.8% on MCPTools. This improvement indicates that moderate surrogate competition helps *MetaMancer* avoid overfitting to an overly simple environment and learn transferable semantic optimization. However, further expanding the pool to 15 tools improves the retrieval ASR but reduces the end-to-end ASR to 89.5% and 91.2%, primarily due to a lower selection ASR. These results suggest that excessive competition among surrogate tools overemphasizes retrieval exposure and weakens the semantic alignment required for selection. Overall, a pool of 10 surrogate tools provides the optimal balance between retrieval visibility and selection influence.



(a) ToolBench **(b) MCPTools**

Figure 5: Impact of Surrogate Tool Scale.



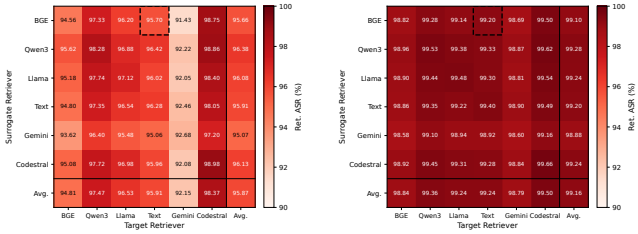
(a) ToolBench **(b) MCPTools**

Figure 6: Impact of Surrogate Query Scale.

5.3.5 Impact of Surrogate Query Scale. As shown in Figure 6, increasing the query budget from 1 to 10 improves the performance across both benchmarks, reaching 92.7% on ToolBench and 93.8% on MCPTools. This trend indicates that moderate query diversity provides richer optimization feedback. However, expanding the budget to 15 reduces the performance to 90.8% and 92.2%, even though the retrieval success remains relatively stable. This suggests that excessive query diversity primarily impairs the selection stage; the metadata must accommodate numerous intent variants, rendering the optimized semantics more diffuse and less aligned. Overall, a budget of 10 surrogate queries yields the optimal balance between query coverage and specificity in the selection stage.

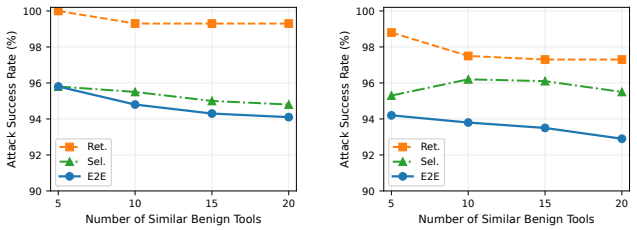
5.3.6 Cross-Retriever Transferability. As illustrated in Figure 7, *MetaMancer* exhibits effective transferability across different retrievers on both ToolBench and MCPTools. Regardless of the specific surrogate-target pairings, it maintains average retrieval attack success rates above 95% and 98%, respectively. The minimal degradation in performance under transfer-based settings suggests that *MetaMancer* avoids overfitting to any specific surrogate model. Rather, it exploits the common vulnerabilities within the shared embedding spaces of modern retrievers. Although minor differences in robustness exist across the target retrievers, the consistently high success rates indicate that current LLM agent systems lack sufficient resistance against cross-retriever perturbations.

5.3.7 Robustness under Multi-Stage Retrieval and Reranking. To evaluate the robustness of *MetaMancer* against more complex target tool selection pipelines, we introduce an additional reranking stage that reduces the top-k retrieved tools to the top-r candidates for subsequent LLM-based selection. We evaluate four representative



(a) ToolBench (b) MCPTools

Figure 7: Cross-retriever transferability of Ret. ASR. Dashed boxes indicate the default BGE-to-Text configuration.



(a) ToolSim (b) QuerySim

Figure 8: Robustness under Curated Tool Competition.

Table 6: Robustness under multi-stage retrieval and reranking. Each cell reports Ret./Sel./E2E ASR (%).

k	r	Cohere	Nemotron	Qwen3	BGE
5	5	95.0/96.5/91.7	94.9/96.5/91.6	95.0/96.0/91.2	95.1/96.6/91.9
10	5	92.5/96.2/89.0	92.4/95.9/88.6	92.9/95.9/89.1	87.7/96.1/84.3
15	5	91.2/95.8/87.4	90.5/95.5/86.4	89.5/96.0/85.9	85.6/96.0/82.2
20	5	89.7/96.1/86.2	88.9/96.4/85.7	87.4/95.9/83.8	83.3/96.8/80.6
10	10	96.8/95.2/92.2	96.8/94.9/91.9	96.8/94.3/91.3	96.8/94.5/91.5
15	10	96.6/93.9/90.7	95.8/94.7/90.7	94.4/94.7/89.4	95.2/92.0/87.6
20	10	95.5/93.4/89.2	94.8/94.2/89.3	92.2/94.8/87.4	93.2/93.1/86.8

rerankers, including Cohere-Rerank-v4.0-Pro (Cohere) [13], Llama-Nemotron-Rerank-VL-1B-v2 (NVIDIA) [42], Qwen3-Reranker-8B (Alibaba Cloud) [69], and BGE-Reranker-v2-M3 (BAAI) [7]. As shown in Table 6, *MetaMancer* maintains an end-to-end ASR between 80.6% and 92.2% across all evaluated configurations. Although more restrictive reranking causes moderate degradation, the consistently high success rates demonstrate that *MetaMancer* remains effective under multi-stage retrieval and reranking.

5.3.8 Robustness under Curated Tool Competition. To evaluate whether *MetaMancer* relies on large tool repositories containing numerous weak or irrelevant competitors, we construct smaller target pools consisting of the top-m benign tools most semantically similar to either the adversarial tool (ToolSim) or each target query (QuerySim), where $m \in \{5, 10, 15, 20\}$. As shown in Figure 8, *MetaMancer* consistently maintains high effectiveness across both settings, achieving end-to-end ASR of 94.1–95.8% under ToolSim and 92.9–94.2% under QuerySim. The marginal degradation as m increases indicates that the attack does not depend on an extensive pool of weak or irrelevant tools. Instead, *MetaMancer* remains effective even when the target pool is deliberately restricted to semantically competitive benign alternatives.

5.4 Optimization Cost Analysis

The optimization cost is a critical practical factor in the strict transfer-based black-box setting. Therefore, we compare *MetaMancer* with representative iterative baselines using a unified accounting scheme for LLM calls. Let $Q := |\mathcal{Q}_{\text{surr}}|$ denote the surrogate query count, B denote the offspring factor, W denote the retained search width, and K denote the maximum number of optimization iterations. For MPMA, C_{cross} denotes the crossover evaluation cost in the GAPMA setting. Table 7 presents the optimization-time LLM-call formula for each method, the instantiated cost under the adopted settings, and the corresponding undefended end-to-end ASR on ToolBench and MCPTools.

Table 7: Optimization-time LLM cost and undefended ASR under a unified accounting scheme.

Method	LLM-call formula	Cost	ASR
MPMA	$2 + K(W + C_{\text{cross}} + 1)$	107	22.5/29.2
AMA	$1 + WQ + KW(1 + BQ)$	5151	37.9/43.4
ToolHijacker	$1 + Q\sum_{t=1}^{K} \min(W, B^{t-1})(B + BQ)$	4501	88.9/83.6
MetaMancer	$3 + K(Q + 4)$	143	92.7/93.8

The results show that *MetaMancer* achieves a favorable trade-off between effectiveness and cost. ToolHijacker and AMA incur substantially higher optimization-time costs because they rely on tree-style expansion or population-level candidate evaluation over surrogate queries. MPMA is cheaper, but this lower cost comes with substantially weaker end-to-end ASR, reflecting its limited control over the retrieve-then-select pipeline. By contrast, *MetaMancer* avoids both tree explosion and population-wide search, and its cost grows linearly with the surrogate query count and iteration budget. Specifically, it employs five role-specialized agents within a fixed optimization workflow, where stage-aware optimization, constraint auditing, and policy aggregation guide targeted updates without repeatedly exploring large candidate populations. This structured optimization reduces wasted exploration while preserving targeted feedback across stages. Meanwhile, *MetaMancer* achieves the strongest undefended ASR on both benchmarks, indicating that its semantically constrained optimization provides a practical balance between attack effectiveness and optimization overhead.

6 Robustness against Defenses

Defense mechanisms against tool-related attacks can be categorized into three layers. Prompt-level preventions, including instructional and sandwich prevention [14, 21], constrain agent behavior by optimizing input prompts. Metadata-level detections, such as AgentScan [56] and LLM-as-a-Judge detectors [25], audit static metadata before invocation to identify malicious actions. Agent-level

Table 8: Robustness of attacks under prompt-level prevention defenses.

Method	ToolBench		MCPTools	
	Instruction	Sandwich	Instruction	Sandwich
BR	1.1 (0)	1.1 (0)	0.4 (0)	0.4 (0)
SI	1.3 (-1.4)	1.3 (-1.4)	1.4 (-1.9)	1.2 (-2.1)
KS	1.4 (-2.9)	1.3 (-3.0)	1.6 (-3.9)	1.4 (-4.1)
GH	2.3 (-2.1)	2.3 (-2.1)	2.6 (-3.5)	2.4 (-3.7)
MPMA	17.7 (-4.8)	17.9 (-4.6)	22.8 (-6.4)	22.4 (-6.8)
AMA	30.7 (-7.2)	30.2 (-7.7)	34.6 (-8.8)	33.9 (-9.5)
ToolHijacker	64.8 (-24.1)	64.5 (-24.4)	58.3 (-25.3)	57.9 (-25.7)
MetaMancer	**92.5 (-0.2)**	**91.0 (-1.7)**	**93.6 (-0.2)**	**92.9 (-0.9)**

guardrails, including ToolSafe [40] and VIGIL [27], act as dynamic runtime monitors that detect and mitigate suspicious behaviors.

6.1 Prompt-level Prevention

Instructional prevention [21]. This method employs a system prompt that treats tool metadata as untrusted content rather than executable instructions. The prompt instructs the model to disregard manipulative or self-promotional language within the metadata and to base the routing decision on the alignment between user intent and actual tool functionality.

Sandwich prevention [14]. This mechanism builds upon instructional prevention by placing the metadata of the candidate tool between an initial instruction and a final safety reminder. The final reminder aims to mitigate the influence of misleading guidance near the end of an extended context, ensuring that the decision of the model remains focused on the functional alignment between the user request and the tool.

Experimental Results. Table 8 shows that *MetaMancer* remains highly effective against prompt-level defenses. Under both instructional and sandwich prevention, the end-to-end ASR decreases by less than 2.0 percentage points and remains above 91.0% on both ToolBench and MCPTools. In contrast, attacks relying on explicit directives experience sharp degradation. The ASR of ToolHijacker drops by 24.1–25.3 percentage points under instructional prevention, while Goal Hijacking approaches a benign level of performance. These results indicate that prompt-level defenses are effective in suppressing overt instruction injection, but are significantly less effective against metadata that is semantically aligned with benign tool descriptions. Sandwich prevention is slightly stronger than instructional prevention, further reducing the ASR of AMA and ToolHijacker by 0.3–0.7 percentage points. This suggests that a trailing safety reminder provides limited additional protection against recency-driven compliance with manipulative metadata.

6.2 Metadata-level Detection

Agent-Scan [56]. Agent-Scan is a widely adopted open-source static analysis framework that audits metadata and configurations for predefined risk patterns before tool invocation. In this study, we employ it as a representative rule-based baseline to assess the efficacy of static scanning methods against metadata manipulation.

LLM-as-a-Judge [25]. The LLM-as-a-Judge paradigm utilizes an independent LLM to evaluate the metadata of candidate tools at the semantic level. By examining the consistency between tool descriptions and parameter schemas, it can detect manipulative intent that may not be captured by fixed rules, making it a stronger and more general detection baseline.

Table 9: Robustness of attacks under metadata-level detection defenses.

Defense	Method	ToolBench		MCPTools	
		Esc.	ASR	Esc.	ASR
Agent-Scan TB TNR: 99.1 MCP TNR: 99.8	Benign Reference	100	1.1 (0)	100	0.4 (0)
	Superlative Inj.	100	2.7 (0)	100	3.3 (0)
	Keyword Stuffing	100	4.3 (0)	100	5.5 (0)
	Goal Hijacking	0	0 (-4.4)	0	0 (-6.1)
	MPMA	100	22.5 (0)	100	29.2 (0)
	AMA	100	37.9 (0)	100	43.4 (0)
	ToolHijacker	0	0 (-88.9)	0	0 (-83.6)
	MetaMancer	**100**	**92.7 (0)**	**100**	**93.8 (0)**
LLM-as-a-Judge TB TNR: 97.1 MCP TNR: 98.6	Benign Reference	100	1.1 (0)	100	0.4 (0)
	Superlative Inj.	0	0 (-2.7)	0	0 (-3.3)
	Keyword Stuffing	0	0 (-4.3)	0	0 (-5.5)
	Goal Hijacking	0	0 (-4.4)	0	0 (-6.1)
	MPMA	90	19.9 (-2.6)	90	24.8 (-4.4)
	AMA	40	16.4 (-21.5)	30	10.8 (-32.6)
	ToolHijacker	0	0 (-88.9)	0	0 (-83.6)
	MetaMancer	**100**	**92.7 (0)**	**100**	**93.8 (0)**

Experimental Results. Against metadata-level defenses, *MetaMancer* achieves a 100% escape rate under both Agent-Scan and LLM-as-a-Judge on ToolBench and MCPTools, maintaining the ASR at 92.7% and 93.8%. This result indicates that constraint-aware optimization keeps adversarial metadata within the distribution of benign tools. By contrast, instruction-heavy attacks are easily exposed by static scanning. Goal Hijacking and ToolHijacker are fully intercepted by Agent-Scan, reducing the escape rates to zero. The LLM-as-a-Judge defense is stronger against shallow or structurally inconsistent perturbations, eliminating Superlative Injection and Keyword Stuffing, and reducing the escape rate of AMA to 40% on ToolBench and 30% on MCPTools. Overall, while metadata-level audits effectively detect explicit directives, unnatural phrasing, and semantic inconsistencies in baseline attacks, they fail to identify the jointly optimized metadata generated by *MetaMancer*.

6.3 Agent-level Guardrail

ToolSafe [40]. ToolSafe audits candidate actions before tool execution to identify potential security risks. Upon detecting an unsafe action, the system generates a safety analysis and validation report, appending it to the model context as feedback to prompt the agent to re-reason and proactively correct its behavior.

VIGIL [27]. VIGIL performs intent anchoring on the user query to extract foundational tasks and logical constraints. It applies an objective rewriting mechanism to tool metadata, preserving neutral functional facts while filtering out manipulative instructions. Before execution, it validates the candidate path against these extracted constraints to ensure semantic alignment with the user goal.

Experimental Results. Against agent-level guardrails, *MetaMancer* remains substantially more effective than all baselines, maintaining

Table 10: Robustness of attacks under agent-level guardrail defenses.

Method	ToolBench		MCPTools	
	ToolSafe	VIGIL	ToolSafe	VIGIL
BR	1.1 (0)	6.6 (+5.5)	0.4 (0)	5.4 (+5.0)
SI	2.7 (0)	6.7 (+4.0)	3.2 (-0.1)	5.5 (+2.2)
KS	1.6 (-2.7)	6.6 (+2.3)	1.3 (-4.2)	5.4 (-0.1)
GH	1.6 (-2.8)	6.5 (+2.1)	1.1 (-5.0)	5.3 (-0.8)
MPMA	22.5 (0)	7.1 (-15.4)	29.0 (-0.2)	6.1 (-23.1)
AMA	37.8 (-0.1)	26.5 (-11.4)	43.2 (-0.2)	29.8 (-13.6)
ToolHijacker	62.2 (-26.7)	46.9 (-42.0)	55.7 (-27.9)	39.8 (-43.8)
MetaMancer	**92.7 (0)**	**85.6 (-7.1)**	**93.7 (-0.1)**	**85.4 (-8.4)**

ASR above 85.4% on both benchmarks under ToolSafe and VIGIL. ToolSafe has a limited impact on *MetaMancer*, whereas VIGIL induces bounded reductions of 7.1 and 8.4 percentage points on ToolBench and MCPTools. In contrast, the performance of baselines degrades sharply. These results indicate that agent-level guardrails are effective against explicit directives and rewritable textual manipulation, but remain less effective against adversarial metadata that semantically aligns with benign tools. We also observe that VIGIL slightly increases benign selection rates by normalizing user intent and tool descriptions, suggesting that its rewriting mechanism acts as a functional normalization filter rather than a comprehensive defense.

7 A Defense Mechanism Against MetaMancer

The robustness of *MetaMancer* against existing defenses indicates that directly detecting complete malicious metadata is insufficient once the attack preserves metadata naturalness, semantic consistency, and runtime validity. We therefore defend against such attacks by restricting the metadata view that is exposed during retrieval and selection, rather than by attempting to identify malicious intent from raw, complete metadata. The key intuition is that tool selection should be based on defender-controlled core functionality, rather than on the complete metadata into which an attacker can inject adversarial perturbations.

Therefore, we design Core View Reconstruction (CVR), a lightweight defense mechanism. Given raw tool metadata consisting of a name, description, required parameters, and optional parameters, CVR reconstructs a core view, containing only a canonical name, a canonical description, and required parameters. The canonical name and description are regenerated within the capability boundary implied by required parameters, while optional parameters are excluded from retrieval and selection and revealed only after the tool has been selected for invocation. This design directly targets the two main points exploited by *MetaMancer*, namely adversarial semantic perturbations and capability expansion introduced by optional parameters. By reconstructing a core view, CVR forces the agent to select tools based on core functionality rather than attack-optimized metadata.

We compare it with two partial variants. Desc.-Only Rewrite canonicalizes the description while still exposing the full schema, and Req.-Only Restrict restricts the schema to required parameters

Table 11: Effectiveness of CVR against *MetaMancer*.

Defense	ToolBench			MCPTools		
	Ret.	Sel.	E2E	Ret.	Sel.	E2E
No Defense	95.2	97.4	92.7	99.2	94.6	93.8
VIGIL	91.0	94.1	85.6	94.3	90.6	85.4
Desc.-Only Rewrite	93.1	94.5	88.0	96.8	91.6	88.7
Req.-Only Restrict	95.2	94.3	89.8	99.2	91.1	90.4
CVR (Ours)	**25.1**	**71.7**	**18.0**	**27.4**	**64.1**	**17.6**

Table 12: Effectiveness of CVR against tool selection attacks.

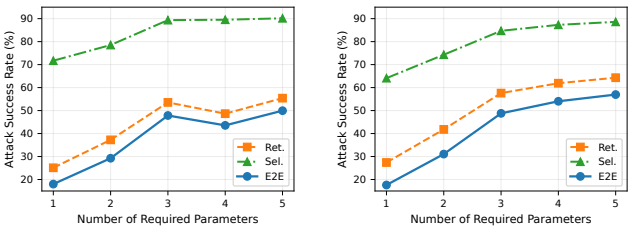
Method	ToolBench			MCPTools		
	Ret.	Sel.	E2E	Ret.	Sel.	E2E
SI	4.0	67.5	2.7 (0)	4.4	59.1	2.6 (-0.7)
KS	4.5	57.8	2.6 (-1.7)	5.0	50.0	2.5 (-3.0)
GH	4.5	57.8	2.6 (-1.8)	5.0	50.0	2.5 (-3.6)
MPMA	9.3	76.3	7.1 (-15.4)	10.1	65.3	6.6 (-22.6)
AMA	12.1	75.2	9.1 (-28.8)	13.2	70.5	9.3 (-34.1)
ToolHijacker	14.0	77.1	10.8 (-78.1)	15.4	62.3	9.6 (-74.0)
MetaMancer	**25.1**	**71.7**	**18.0 (-74.7)**	**27.4**	**64.1**	**17.6 (-76.2)**

without canonicalizing the description. As shown in Table 11, CVR significantly reduces the end-to-end ASR of *MetaMancer* from over 92% to approximately 18% across both benchmarks. This defensive capability stems primarily from disrupting the initial retrieval stage, where restricting the exposed metadata to a minimal core view causes the retrieval ASR to decrease from over 95% to roughly 26%. Furthermore, the ablation results demonstrate that isolated constraints on either the descriptions or schemas yield marginal improvements, confirming that simultaneously neutralizing the description and the expansion of optional parameters is strictly necessary to eliminate the structural redundancy of the attack.

As shown in Table 12, CVR effectively generalizes to different tool selection attacks. For the baseline attacks, CVR limits retrieval ASR to at most 15.4% and restricts residual E2E ASR to at most 10.8% on ToolBench and 9.6% on MCPTools. This suppression is particularly pronounced for attacks with high undefended success rates. Specifically, the E2E ASR of ToolHijacker drops by 78.1 and 74.0 percentage points on the two benchmarks, respectively, while MPMA and AMA are also substantially mitigated. Although *MetaMancer* remains the most resilient attack against CVR, its E2E ASR is reduced to 18.0% and 17.6%. These results demonstrate that CVR provides consistent protection against diverse tool selection attacks.

8 Adaptive Attacks

To further explore the boundary of tool selection attacks, we design an adaptive attack. Because CVR reconstructs the core view of metadata based on the capabilities implied by required parameters, a natural adaptive strategy is to reinforce the semantic information and capabilities carried by required parameters. Specifically, while satisfying stealth constraints, we add required parameters that expand the functional capabilities, thereby allowing more semantic information to be retained in the reconstructed core view.



(a) ToolBench **(b) MCPTools**

Figure 9: Impact of Required Parameters.

Figure 9 reports the CVR-aware adaptive attack results as the number of required parameters varies from 1 to 5. Across both ToolBench and MCPTools, increasing the number of required parameters partially restores attack effectiveness, primarily through higher retrieval ASR. This shows that the inflation of required parameters helps more semantics survive the core-view reconstruction and recover retrieval visibility. By contrast, the gain from the selector is limited once the adversarial tool is retrieved. The plateauing trends reveal the underlying trade-off between effectiveness and usability. More required parameters strengthen the exposed core view but also make the tool interface heavier and less generally usable.

9 Conclusion

In this paper, we studied practical tool selection attacks against LLM agents in open tool ecosystems. We showed that tool metadata is a critical security surface that can be compromised to manipulate the tool selection process. Under a strict transfer-based black-box setting, we formulated adversarial tool selection as a semantically constrained bilevel optimization problem, and then proposed *MetaMancer*, a surrogate-based multi-agent semantic optimization framework that performs stage-aware effectiveness optimization, semantic constraint auditing, conflict-aware policy aggregation, and constraint-compliant atomic updates. Across ToolBench and MCPTools, covering 20 tasks, 9,729 candidate tools, 8 target LLMs, and 6 representative defenses, *MetaMancer* achieves around 93% end-to-end ASR without defenses and remains above 91% when averaged across defenses. Further analysis demonstrates that *MetaMancer* is highly optimization-efficient. These results demonstrate that *MetaMancer* is effective, stealthy, and robust across heterogeneous tool ecosystems, target LLMs, and defensive settings. To mitigate this threat, we further proposed Core View Reconstruction (CVR), which reconstructs a core view of tool metadata based on the capabilities implied by required parameters. The experimental results show that CVR substantially reduces the effectiveness of *MetaMancer*. Overall, our findings show that tool metadata should be treated as an adversarial surface, and the tool selection process of LLM agents must be secured as a core component of the agent execution pipeline.

Acknowledgments

This work was supported in part by the General Research Fund (GRF) of the Hong Kong Research Grants Council (RGC) under Grant PolyU 15230025.

References

[1] Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, and Ameet Deshpande. 2024. Geo: Generative engine optimization. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*. 5–16.

[2] Anthropic. 2025. Introducing Claude Sonnet 4.5. Retrieved September 13, 2026 from https://www.anthropic.com/news/claude-sonnet-4-5

[3] Anthropic. 2026. Claude Code. Retrieved September 13, 2026 from https://claude.com/product/claude-code

[4] Anthropic. 2026. Tool Search Tool. Retrieved September 13, 2026 from https://platform.claude.com/docs/en/agents-and-tools/tool-use/tool-search-tool

[5] Apify. 2026. Apify Store. Retrieved September 13, 2026 from https://apify.com/store

[6] Yauhen Babakhin, Radek Osmulski, Ronay Ak, Gabriel Moreira, Mengyao Xu, Benedikt Schifferer, Bo Liu, and Even Oldridge. 2025. Llama-Embed-Nemotron-8B: A Universal Text Embedding Model for Multilingual and Cross-Lingual Tasks. *arXiv preprint arXiv:2511.07025* (2025).

[7] Beijing Academy of Artificial Intelligence. 2024. BGE-Reranker-v2-M3. Retrieved September 13, 2026 from https://huggingface.co/BAAI/bge-reranker-v2-m3

[8] Thierry Blankenstein, Jialin Yu, Zixuan Li, Vassilis Plachouras, Sunando Sengupta, Philip Torr, Yarin Gal, Alasdair Paren, and Adel Bibi. 2026. Biasbusters: Uncovering and mitigating tool selection bias in large language models. In *International Conference on Learning Representations*, Vol. 2026. 102460–102489.

[9] Yiming Cao, Yanjie Li, Kaisheng Liang, and Bin Xiao. 2026. Enhancing targeted adversarial attacks on large vision-language models via intermediate projector. *IEEE Transactions on Information Forensics and Security* (2026).

[10] Yiming Cao, Dong Wang, Xinqi Lyu, and Bin Xiao. 2026. PureProof: Diffusion-Resistant Black-box Targeted Attack on Large Vision-Language Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8621–8630.

[11] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216* 4, 5 (2024).

[12] Liuji Chen, Hao Gao, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. 2025. Select Me! When You Need a Tool: A Black-box Text Attack on Tool Selection. *arXiv preprint arXiv:2504.04809* (2025).

[13] Cohere. 2025. Rerank v4.0. Retrieved September 13, 2026 from https://docs.cohere.com/changelog/rerank-v4.0

[14] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems* 37 (2024), 82895–82920.

[15] Kazem Faghih, Wenxiao Wang, Yize Cheng, Siddhant Bharti, Gaurang Sriramanan, Sriram Balasubramanian, Parsa Hosseini, and Soheil Feizi. 2025. Tool Preferences in Agentic LLMs are Unreliable. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 20965–20980.

[16] Xiang Fei, Xiawu Zheng, and Hao Feng. 2025. Mcp-zero: Active tool discovery for autonomous llm agents. *arXiv preprint arXiv:2506.01056* (2025).

[17] Google Cloud. 2025. Text Embeddings API. Retrieved September 13, 2026 from https://docs.cloud.google.com/gemini-enterprise-agent-platform/reference/models/text-embeddings-api

[18] Google Cloud. 2026. Gemini 3.1 Pro. Retrieved September 13, 2026 from https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/gemini/3-1-pro

[19] Feng He, Tianqing Zhu, Dayong Ye, Bo Liu, Wanlei Zhou, and Philip S Yu. 2025. The emerged security and privacy of llm agent: A survey with case studies. *Comput. Surveys* 58, 6 (2025), 1–36.

[20] Junda He, Christoph Treude, and David Lo. 2025. Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–30.

[21] 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* (2024).

[22] Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. 2025. Model context protocol (mcp): Landscape, security threats, and future research directions. *ACM Transactions on Software Engineering and Methodology* (2025).

[23] 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* (2023).

[24] Hussein Jawad and Nicolas JB Brunel. 2026. ToolFlood: Beyond Selection–Hiding Valid Tools from LLM Agents via Semantic Covering. *arXiv preprint arXiv:2603.13950* (2026).

[25] Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, et al.

2025. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 2757–2791.

[26] Zichuan Li, Jian Cui, Xiaojing Liao, and Luyi Xing. 2025. Les dissonances: Cross-tool harvesting and polluting in multi-tool empowered llm agents. *arXiv e-prints* (2025), arXiv–2504.

[27] Junda Lin, Zhaomeng Zhou, Zhi Zheng, Shuochen Liu, Tong Xu, Yong Chen, and Enhong Chen. 2026. VIGIL: Defending LLM Agents Against Tool Stream Injection via Verify-Before-Commit. *arXiv preprint arXiv:2601.05755* (2026).

[28] Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556* (2025).

[29] Yihao Liu, Jinhe Huang, Yanjie Li, Dong Wang, and Bin Xiao. 2024. Generative AI model privacy: a survey. *Artificial Intelligence Review* 58, 1 (2024), 33.

[30] 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)*. 1831–1847.

[31] Yihao Liu, Xinqi Lyu, Dong Wang, Yanjie Li, and Bin Xiao. 2026. LOMIA: Label-only membership inference attacks against pre-trained large vision-language models. *Advances in Neural Information Processing Systems* 38 (2026), 153714–153737.

[32] Xinqi Lyu, Yihao Liu, Yanjie Li, and Bin Xiao. 2025. Pla: Prompt learning attack against text-to-image generative models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 16851–16860.

[33] Xinqi Lyu, Yihao Liu, Dong Wang, and Bin Xiao. 2026. Red-teaming Retrieval-Augmented Diffusion Models via Poisoning Knowledge Bases. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 34961–34970.

[34] Manus. 2026. Welcome. Retrieved September 13, 2026 from https://manus.im/docs/introduction/welcome

[35] MCP.so. 2026. MCP.so: MCP Marketplace. Retrieved September 13, 2026 from https://mcp.so/

[36] Meta AI. 2025. The Llama 4 Herd: The Beginning of a New Era of Natively Multimodal AI Innovation. Retrieved September 13, 2026 from https://ai.meta.com/blog/llama-4-multimodal-intelligence/

[37] Mistral AI. 2025. Codestral Embed. Retrieved September 13, 2026 from https://mistral.ai/news/codestral-embed

[38] Kanghua Mo, Li Hu, Yucheng Long, and Zhihao Li. 2026. Attractive metadata attack: Inducing LLM agents to invoke malicious tools. *Advances in Neural Information Processing Systems* 38 (2026), 162770–162793.

[39] Model Context Protocol. 2026. MCP Client Best Practices. Retrieved September 13, 2026 from https://modelcontextprotocol.io/docs/develop/clients/client-best-practices

[40] Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. 2026. ToolSafe: Enhancing Tool Invocation Safety of LLM-based agents via Proactive Step-level Guardrail and Feedback. *arXiv preprint arXiv:2601.10156* (2026).

[41] Fredrik Nestaas, Edoardo Debenedetti, and Florian Tramèr. 2025. Adversarial Search Engine Optimization for Large Language Models. In *The Thirteenth International Conference on Learning Representations*.

[42] NVIDIA. 2025. Llama-Nemotron-Rerank-VL-1B-v2. Retrieved September 13, 2026 from https://docs.api.nvidia.com/nim/reference/nvidia-llama-nemotron-rerank-vl-1b-v2

[43] OpenAI. 2024. text-embedding-3-large. Retrieved September 13, 2026 from https://developers.openai.com/api/docs/models/text-embedding-3-large

[44] OpenAI. 2025. GPT-5.1: A Smarter, More Conversational ChatGPT. Retrieved September 13, 2026 from https://openai.com/index/gpt-5-1/

[45] OpenAI. 2026. Codex. Retrieved September 13, 2026 from https://openai.com/codex/

[46] OpenAI. 2026. Optimize Metadata. Retrieved September 13, 2026 from https://developers.openai.com/plugins/guides/optimize-metadata

[47] OpenClaw. 2026. OpenClaw: Open-Source AI Assistant. Retrieved September 13, 2026 from https://openclaw.ai/

[48] Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

[49] PulseMCP. 2026. MCP Server Directory. Retrieved September 13, 2026 from https://www.pulsemcp.com/servers

[50] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, et al. 2024. Tool learning with foundation models. *Comput. Surveys* 57, 4 (2024), 1–40.

[51] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, Vol. 2024. 9695–9717.

[52] Qwen Team. 2026. Qwen3.5: Towards Native Multimodal Agents. Retrieved September 13, 2026 from https://qwen.ai/blog?id=qwen3.5

[53] Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. 2025. Prompt injection attack to tool selection in llm agents. *arXiv preprint arXiv:2504.19793* (2025).

[54] Zhengliang Shi, Shen Gao, Lingyong Yan, Yue Feng, Xiuyi Chen, Zhumin Chen, Dawei Yin, Suzan Verberne, and Zhaochun Ren. 2025. Tool learning in the wild: Empowering language models as automatic tool agents. In *Proceedings of the ACM on Web Conference 2025*. 2222–2237.

[55] Jonathan Sneh, Ruomei Yan, Jialin Yu, Philip Torr, Yarin Gal, Sunando Sengupta, Eric Sommerlade, Alasdair Paren, and Adel Bibi. 2025. Tooltweak: An attack on tool selection in llm-based agents. *arXiv preprint arXiv:2510.02554* (2025).

[56] Snyk. 2026. Snyk Agent Scan. Retrieved September 13, 2026 from https://github.com/snyk/agent-scan/releases/tag/v0.4.18 Version 0.4.18.

[57] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295* (2024).

[58] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[59] Dong Wang, Xiangyu He, Xinqi Lyu, and Bin Xiao. 2026. Breaking Multimodal LLM Safety via Video-Driven Prompting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8566–8576.

[60] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[61] Maolin Wang, Yingyi Zhang, Bowen Yu, Bingguang Hao, Cunyin Peng, Yicheng Chen, Wei Zhou, Jinjie Gu, Chenyi Zhuang, Ruocheng Guo, et al. 2026. Function calling in large language models: Industrial practices, challenges, and future directions. *Comput. Surveys* 58, 9 (2026), 1–37.

[62] Zihan Wang, Rui Zhang, Yu Liu, Wenshu Fan, Wenbo Jiang, Qingchuan Zhao, Hongwei Li, and Guowen Xu. 2026. Mpma: Preference manipulation attack against model context protocol. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 35838–35846.

[63] xAI. 2025. Grok 4.1. Retrieved September 13, 2026 from https://x.ai/news/grok-4-1

[64] Jehyeok Yeon, Isha Chaudhary, and Gagandeep Singh. 2025. Quantifying Distributional Robustness of Agentic Tool-Selection. *arXiv preprint arXiv:2510.03992* (2025).

[65] Miao Yu, Fanci Meng, Xinyun Zhou, Shilong Wang, Junyuan Mao, Linsey Pan, Tianlong Chen, Kun Wang, Xinfeng Li, Yongfeng Zhang, et al. 2025. A survey on trustworthy llm agents: Threats and countermeasures. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 6216–6226.

[66] Z.AI. 2025. GLM-4.7: Overview. Retrieved September 13, 2026 from https://docs.z.ai/guides/llm/glm-4.7

[67] Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. 2025. Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents. In *The Thirteenth International Conference on Learning Representations*.

[68] Rupeng Zhang, Haowei Wang, Junjie Wang, Mingyang Li, Yuekai Huang, Dandan Wang, and Qing Wang. 2025. From allies to adversaries: Manipulating llm tool-calling through adversarial injection. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 2009–2028.

[69] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176* (2025).

A Open Science and Ethical Considerations

MetaMancer investigates a dual-use security risk in the tool selection of LLM agents. All experiments were conducted in controlled benchmark environments without targeting deployed agent services, third-party marketplaces, production APIs, or real users. No harmful real-world tool actions were executed, and no private user data, credentials, or personally identifiable information were accessed. By characterizing this attack surface and evaluating corresponding defenses, including CVR, this study aims to improve the security of tool-augmented LLM agents.