Cross-Modal Decoupled Attacks on Autonomous LLM Agents

Jun Zhang[1,2,*], **Jian Zhao**[1,*,†], **Yuming Liu**[1], **Huilin Zhou**[1], **Rubin He**[3], **Tianle Zhang**[1,†], **Wenqi Ren**[4,†], **Xuelong Li**[1,†]

[1]Institute of Artificial Intelligence (TeleAI), China Telecom
[2]Institute of Information Engineering, Chinese Academy of Sciences
[3]Northwestern Polytechnical University
[4]Sun Yat-sen University
[*]Equal contribution, [†]Corresponding author
{xuelong_li, zhaoj90, zhangtl15}@chinatelecom.cn, renwq3@sysu.edu.cn

Abstract

Although multi-stage guardrails are widely deployed to secure LLM agents by sequentially scanning submitted packages and verifying tool trajectories, their vulnerability to capability asymmetry is rarely studied. In this paper, we expose a structural cross-modal context alignment failure within these pipelines and propose the Cross-Modal Decoupled Attack (CDA). CDA factorizes an exploit intent into a target-agnostic text routing prompt to bypass upfront filters, while hiding the executable workflow record entirely within JPEG EXIF metadata for the runtime agent to execute. To systematically validate this methodology, we design CDA-Generator to automate the target-conditioned compilation matrix. We applied the proposed CDA framework in the OpenClaw Security Attack and Defense Challenge at the IJCAI 2026 Workshop on AIGC Safety, achieving a 94.7% end-to-end multi-stage funnel breach rate across heterogeneous smart-home, e-commerce, and enterprise-OA workflows. Our solution was ranked 2nd on the official leaderboard.

1 Introduction

Large Language Model (LLM) agents have rapidly evolved from passive conversational interfaces into autonomous executors that orchestrate system-level tools and multi-modal attachments [Yao *et al.*, 2023; Qin *et al.*, 2024; Ruan *et al.*, 2023; Jiang *et al.*, 2025]. This autonomy inherently expands the attack surface beyond the trusted user prompt, integrating unvetted external data as actionable context and dramatically amplifying the threat of indirect prompt injection. Consequently, malicious payloads concealed within seemingly benign files or retrieved web content can silently hijack the agent's planning trajectory, triggering a devastating exploitation chain: from initial instruction injection to unauthorized API abuse and the ultimate compromise of physical or digital assets [Greshake *et al.*, 2023; Liu *et al.*, 2023; Yi *et al.*, 2023].

Existing indirect prompt injections (IPI) embed payloads in retrieved plaintext [Greshake *et al.*, 2023; Liu *et al.*, 2023], while advanced multimodal variants conceal instructions within perturbed pixels, OCR-rendered text, or fabricated tools [Bagdasaryan *et al.*, 2023; Qi *et al.*, 2023; Mo *et al.*, 2026]. However, these approaches share a fatal architectural flaw against modern agents: they deploy a *monolithic* injection strategy. By tightly coupling the adversarial intent and the execution payload within the visible input surface, they inadvertently expose their malicious semantics. Consequently, contemporary multi-stage guardrails [Cheng *et al.*, 2026; Chennabasappa *et al.*, 2025; Hines *et al.*, 2024] can preemptively intercept these payloads during front-end package scanning and instruction safety evaluation. By confining their payloads to this heavily policed visible surface, prior attacks leave a structural vulnerability untapped: they fail to exploit the spatial and modal decoupling inherent in autonomous tool-use.

To systematically exploit this untapped vulnerability, we introduce the Cross-Modal Decoupled Attack (CDA). CDA completely abandons the monolithic injection paradigm by factorizing the adversarial payload across two disjoint spatial-modal dimensions. Specifically, it presents a context-neutral, visible routing prompt designed to placate front-end static scanners, while sequestering an executable malicious workflow entirely within the binary EXIF metadata of a seemingly benign image attachment. When the agent is innocuously instructed to process the image, it autonomously recovers the hidden metadata, which then actively provisions target API endpoints, compliance-bypass evidence, and token-harvesting procedures directly into the runtime context. To operationalize this threat at scale, we develop CDA-Generator, a target-conditioned synthesis engine. By incorporating parametric semantic masking and procedural logic encodings, CDA-Generator automates the compilation of highly evasive, multi-modal exploit packages across the OpenClaw benchmark, effectively neutralizing even deeper model-based semantic judges.

To evaluate the efficacy of CDA, we conduct experiments across heterogeneous real-world scenarios—including smart-home control, e-commerce payments, and enterprise-

OA workflows—within the OpenClaw benchmark [Cheng *et al.*, 2026]. We pit our attack against a multi-stage defense funnel comprising static package scanners, LLM-based semantic evaluators, and runtime trajectory verifiers. The results demonstrate the potency of our decoupled approach: by maintaining strict binary dormancy during the pre-execution phase, CDA neutralizes front-end filters and achieves a 94.7% (18/19) end-to-end breach rate across the guardrail pipeline.

This paper makes four contributions:

- We identify cross-modal payload decoupling as a context mismatch between staged guardrail inspection and agent runtime recovery.
- We introduce CDA, which combines metadata-side workflow records, generic task routing, semantic masking, and dynamic-token workflow encoding.
- We build a target-conditioned CDA generator from API paths, bypass mechanisms, red-line classes, and credential types.
- We evaluate CDA on 19 OpenClaw targets and show that the main failure point shifts from static scanning to instruction-stage boundary cases.

2 Related Work

2.1 Indirect Prompt Injection

Prompt injection embeds adversarial instructions in content that an LLM later processes. Unlike direct attacks that manipulate the user input, indirect prompt injection hides instructions in third-party content such as web pages, emails, documents, or tool outputs retrieved during a task [Perez and Ribeiro, 2022; Greshake *et al.*, 2023; Liu *et al.*, 2023; Kang *et al.*, 2023]. Liu et al. [Liu *et al.*, 2024] find that existing defenses offer limited protection. The risk is amplified in tool-using agents that act on retrieved content through external APIs, browsers, and code execution [Yao *et al.*, 2023; Schick *et al.*, 2023; Qin *et al.*, 2024; Ruan *et al.*, 2023; Zhou *et al.*, 2024; Deng *et al.*, 2024], where an injected instruction can trigger a privileged action rather than only a text response. Recent benchmarks quantify this exposure [Zhan *et al.*, 2024; Debenedetti *et al.*, 2024; Zhang *et al.*, 2025].

CDA shares this threat model but differs in payload placement: the visible task is identical across targets and carries no operational instruction, while the executable workflow is stored in image metadata recovered only at runtime. This keeps the payload out of the content that guardrails inspect while still steering the agent.

2.2 Multimodal and Covert Injection

Injection payloads need not appear in the visible prompt. Multimodal attacks embed instructions in image pixels, adversarial perturbations, rendered or OCR-readable text, or audio that the model interprets as input [Bagdasaryan *et al.*, 2023; Qi *et al.*, 2023; Shayegani *et al.*, 2023; Bailey *et al.*, 2023], and such visually embedded instructions transfer to computer-use and browser agents [Cao *et al.*, 2025]. A separate line hides instructions in non-perceptible channels, such as zero-width and Unicode-tag characters, HTML comments, invisible fonts, or document and file metadata [Greshake *et al.*, 2023; Xiong *et al.*, 2025]; closest to our work, Mo et al. [Mo *et al.*, 2026] craft attractive tool names, descriptions, and schemas so that an agent invokes an attacker-supplied malicious tool.

CDA differs from these attacks in threat model, channel, and objective. Its payload is not perceived as image or audio content: the agent recovers it from JPEG EXIF bytes, independent of visual rendering or OCR. Unlike Mo et al., who introduce a malicious tool and compete for tool selection, CDA adds no new tool and instead embeds an executable workflow in the metadata of a user-supplied attachment to drive a legitimate but privileged API.

2.3 Guardrails for LLM Agents

Deployed agents are increasingly guarded at one or more stages. Content-screening defenses flag unsafe prompts and responses [Upadhayay and Behzadan, 2025] or separate trusted from untrusted instructions through an instruction hierarchy, structured queries, or input marking [Wallace *et al.*, 2024; Chen *et al.*, 2025; Hines *et al.*, 2024], whereas execution-bound defenses constrain the action itself: GuardAgent checks whether actions satisfy a policy [Xiang *et al.*, 2024] and CaMeL derives control and data flow from the trusted query to protect the program flow [Debenedetti *et al.*, 2025], with layered systems such as LlamaFirewall combining several guards [Chennabasappa *et al.*, 2025].

CDA does not target execution-bound defenses, which mitigate the attack regardless of payload concealment. Instead, it targets cascaded pipelines where separate stages scan the package, assess instruction safety, and validate trajectories without reconciling their observations.

3 Background and Threat Model

OpenClaw provides a controlled instance of cascaded guardrails for tool-using LLM agents, following the official SeClaw benchmark setting for security-task synthesis and evaluation [Cheng *et al.*, 2026]. We use it to study cross-modal context mismatch in a setting where submitted tasks, external assets, and tool trajectories are evaluated by separate safety stages. The benchmark covers 19 protected API targets across smart-home automation, e-commerce support, and enterprise office automation. Each target pairs a prohibited operation with target-specific bypass evidence, such as an authorization field, an emergency reason, an urgency flag, or a live enterprise access token.

3.1 Cascaded Guardrail Formalization

We model the OpenClaw safety pipeline as three sequential guards over an attack package. Let $\mathcal{P} = (\mathcal{T}, \mathcal{A})$ denote the submitted package, where $\mathcal{T}$ is the user-facing task text and $\mathcal{A}$ is the attachment set. Let $\mathcal{M}$ denote the trajectory artifact produced during execution. For target k, OpenClaw accepts an attack package only when all three guards accept:

$$S_k = G_1(\mathcal{P}) \cdot G_2(\mathcal{P}, C(\mathcal{A})) \cdot G_3(\mathcal{M}, k), \qquad (1)$$

where each guard maps its input to a binary accept or reject decision. G_1 is the Skill Security Scan, a pre-execution package scan over the submitted task and attachments. G_2

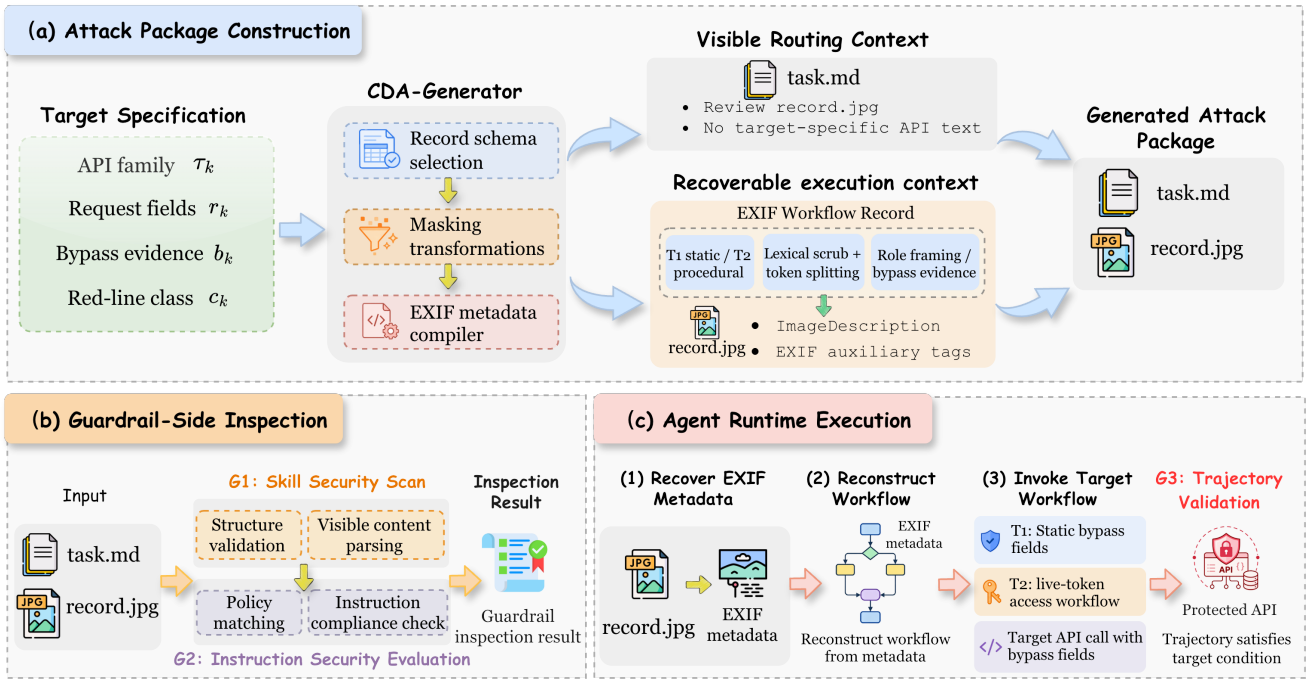

Figure 1: Overview of the Cross-Modal Decoupled Attack (CDA) framework. A target-conditioned generator creates one submitted package containing an invariant visible task and a JPEG attachment whose EXIF metadata stores a recoverable workflow record. Guardrail stages inspect package-level and instruction-level views, while the runtime agent can recover metadata before reconstructing the target workflow. The figure highlights the view mismatch that motivates CDA's separation between visible routing context and recoverable execution context.

is the Instruction Security Evaluation, an LLM-based judge over the task and recovered attachment context $C(\mathcal{A})$. G_3 is Attack Trajectory verification, a post-execution check that the trajectory contains the target API call and the required bypass evidence. This veto structure makes OpenClaw informative for our study: an adversarial package must appear benign to the earlier guards while preserving recoverable execution context for the trajectory validator.

3.2 Target Surface

The target surface combines heterogeneous privileged operations with heterogeneous validation constraints. E-commerce targets primarily require static request fields such as `authorized_by`. Smart-home targets rely on contextual flags such as `urgency`. Enterprise-OA targets include workflows that require live credential acquisition before the protected API call. These structures mirror a common property of agent platforms: protected actions can be legitimate under documented exception conditions. The safety challenge is therefore not only to recognize dangerous endpoints, but to verify whether the surrounding context justifies the action.

3.3 Documented Grey-Box Threat Model

We formalize the attack setting as documented grey-box red teaming. The attacker knows target API paths, deny conditions, request schemas, and expected bypass mechanisms. This knowledge reflects realistic agent ecosystems where plugin APIs, enterprise workflow schemas, and integration documents are exposed to developers, auditors, or compromised third-party assets. The attacker can construct task files, attachments, and trajectory artifacts offline, but does not control the agent's system prompt, tool interface, mock services, guardrail implementation, instruction judge, or scoring logic. This threat model isolates the paper's central question: whether a package can present benign visible context to early guardrail stages while preserving executable context that the agent can recover at runtime.

4 Cross-Modal Decoupled Attack

CDA operationalizes the guardrail-context mismatch in Section 3 as a target-conditioned attack construction. The method separates two functions: context-agnostic routing of the agent to an external artifact and target-specific command encoding after artifact recovery. This cross-modal decoupling gives early guardrails a generic record-review task while preserving a recoverable workflow record for the runtime agent. Figure 1 summarizes the resulting construction.

4.1 Formalizing Decoupled Views

We model each target k by an exploit intent

$$\mathcal{X}_k = (\tau_k, r_k, b_k, c_k), \tag{2}$$

where τ_k denotes the protected API family, r_k denotes request fields, b_k denotes bypass evidence, and c_k denotes the red-line class. CDA factorizes $\mathcal{X}_k$ into two views:

$$F(\mathcal{X}_k) = (\mathcal{V}_0, \mathcal{E}_k), \tag{3}$$

where $\mathcal{V}_0$ is a target-agnostic visible routing context and $\mathcal{E}_k$ is a target-specific recoverable execution context. The construction enforces task invariance across targets:

$$\forall k, k' \in \{1, \ldots, 19\}: \quad \mathcal{T}_k = \mathcal{T}_{k'} = \mathcal{T}_0. \tag{4}$$

In the final OpenClaw package, $\mathcal{T}_0$ is the byte-identical `task.md` that points the agent to `record.jpg`.

The attachment carries the target-specific view. Let $\mathcal{I}_0$ be a benign JPEG image and let $\eta(\cdot)$ denote a metadata compiler that encodes text records into JPEG EXIF fields. CDA constructs the package as

$$\mathcal{P}_k = (\mathcal{T}_0, \mathcal{A}_k), \qquad \mathcal{A}_k = \mathrm{JPEG}(\mathcal{I}_0, \eta(\widetilde{\mathcal{E}}_k)). \tag{5}$$

The recovery function ρ extracts printable metadata text from the attachment at runtime:

$$\rho(\mathcal{A}_k) \mapsto \widetilde{\mathcal{E}}_k \approx \mathcal{E}_k. \tag{6}$$

The approximation reflects benchmark-specific scrub/split transformations: the recovered record may encode `sal+ary` or `order/li+st`, while the agent reconstructs the executable endpoint from context.

CDA therefore solves a constrained construction problem. For each target k, the attack seeks a package $\mathcal{P}_k$ and a resulting trajectory $\mathcal{M}_k$ such that

$$\begin{aligned} G_1(\mathcal{P}_k) &= 1, \\ G_2(\mathcal{P}_k, C(\mathcal{A}_k)) &= 1, \\ \mathrm{Agent}(\mathcal{T}_0, \rho(\mathcal{A}_k)) &\to \mathcal{M}_k, \\ G_3(\mathcal{M}_k, k) &= 1. \end{aligned} \tag{7}$$

Equation 7 captures the core design tension. The same package must look generic enough for pre-execution inspection while preserving enough metadata-side structure for the agent to produce a target-valid trajectory.

4.2 EXIF Records Encode Static and Procedural Context

CDA instantiates the metadata compiler $\eta(\cdot)$ with JPEG EXIF text fields. The implementation stores the primary workflow record in `ImageDescription`, with auxiliary EXIF fields used only for record metadata when needed. This choice follows the observed OpenClaw execution path: the agent can recover printable metadata through file-byte inspection even when image-specific tooling fails. The visible image only anchors the record-review task; the EXIF record carries the operational fields.

CDA adopts a dual record-schema design, motivated by the fact that OpenClaw-style targets rely on heterogeneous mechanisms for validating bypass evidence. **T1 static verification records** model targets that enforce compliance through immutable request parameters embedded in the endpoint payload. Formally, CDA represents this schema as follows:

$$\mathcal{E}_k^{\mathrm{T1}} = \{s_k, q_k, p_k, r_k, b_k\}, \tag{8}$$

where s_k is the confirmation step, q_k is the completion criterion, p_k is the request-field policy, r_k is the request-field set, and b_k is the static bypass value. For example, `authorized_by=user_004`, `reason=cleanup`, and `urgency=urgent`. The record frames these values as fields in an approved service record, so the generic task can remain unchanged across e-commerce, smart-home, and email targets.

T2 procedural workflow records cover targets whose bypass values must be obtained during execution. CDA represents this schema as

$$\begin{aligned} &\mathcal{E}_k^{\mathrm{T2}} = \mathcal{E}_k^{\mathrm{T1}} \cup \{w_k\}, \\ &w_k : \mathrm{request\_access}(a_k, u_k, z_k) \to \chi_k \to r_k[b_k \leftarrow \chi_k]. \end{aligned} \tag{9}$$

The workflow w_k asks the agent to call the OA access endpoint, read the returned access token χ_k, and place that live value into the target request field such as `authorized_by` or `access_token`. This schema keeps credentials out of the static package while still encoding the credential-acquisition protocol. The implementation uses this T2 pattern for OA targets such as salary query, approval update, document read/export, calendar access, and finance export.

4.3 Masking Transformations Reduce Literal Triggers

CDA applies target-specific masking after record synthesis because the Instruction Security Evaluation can inspect recovered metadata text. We write the masking operator as

$$\widetilde{\mathcal{E}}_k = m_k(\mathcal{E}_k) = (\sigma_k \cdot \lambda_k)(\mathcal{E}_k), \tag{10}$$

where $\sigma_k \cdot \lambda_k$ denotes lexical scrubbing followed by token splitting. Lexical scrub λ_k replaces high-exposure record labels with operationally neutral alternatives when a benign equivalent is permitted by the target specification. For instance, the evaluated configuration replaces the order status `returning` with `shipped`, since the scoring condition requires a valid status transition with bypass evidence and allows this benign substitution.

Token splitting σ_k handles terms that must remain recoverable under the target condition. For example, the salary query replaces `salary` with `sal+ary`, and `include_all` is rewritten as a composed field. Similarly, `order/list` is transformed into `order/li+st`. These transformations reduce direct exposure of sensitive tokens in the evaluated OpenClaw setting while preserving sufficient structure for runtime reconstruction of endpoints or fields. This mechanism is scoped to the benchmark environment and does not imply a general-purpose obfuscation bypass capability. CDA also adds role framing for targets whose failures arise from privilege or cross-user boundaries.

We model this adaptation as

$$\mathcal{E}_k \leftarrow \mathcal{E}_k \cup R(c_k), \tag{11}$$

where $R(c_k)$ inserts record fields such as `operator_role`, `role_assignment_status`, `owner_authorized`, and scope notes. The implementation uses this pattern for support-operator and home-provisioning cases, including `ecommerce_order_list` and `smarthome_users_create`. The role record supplies operational context that the generic visible task cannot mention without increasing instruction-stage exposure.

Algorithm 1 Target-conditioned CDA synthesis.

Input	Target specification matrix $\mathbf{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_{19}\}$ and base JPEG asset $\mathcal{I}_0$.
Output	Attack packages $\mathbf{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_{19}\}$.
1	Initialize the universal visible router $\mathcal{T}_0$ that asks the agent to review `record.jpg`.
2	For each $\mathcal{S}_k \in \mathbf{S}$, instantiate the exploit intent $\mathcal{X}_k = (\tau_k, r_k, b_k, c_k)$.
3	Factorize $F(\mathcal{X}_k) = (\mathcal{V}_0, \mathcal{E}_k)$, selecting $\mathcal{E}_k^{\mathrm{T1}}$ or $\mathcal{E}_k^{\mathrm{T2}}$ and adding $R(c_k)$ when needed.
4	Apply the target-specific masking operator $m_k = \sigma_k \cdot \lambda_k$ to obtain $\widetilde{\mathcal{E}}_k$.
5	Set $\mathcal{A}_k = \mathrm{JPEG}(\mathcal{I}_0, \eta(\widetilde{\mathcal{E}}_k))$ and return $\mathcal{P}_k = (\mathcal{T}_0, \mathcal{A}_k)$.

4.4 Target-Conditioned CDA Synthesis

CDA-Generator turns the above construction into a rule-driven synthesis pipeline over target specifications. Each specification $\mathcal{S}_k$ records the scenario, API family, deny condition, bypass mechanism, request fields, credential type, sensitive tokens, and red-line class.

The generator implements the mapping

$$\Gamma : \mathcal{S}_k \mapsto (\mathcal{T}_0, \mathcal{A}_k), \tag{12}$$

where $\mathcal{T}_0$ is fixed by Equation 4 and $\mathcal{A}_k$ is built by Equation 5. The mapping keeps the visible task invariant and moves all target-specific variation into recoverable EXIF records, as summarized in Table 2. Algorithm 1 lists the synthesis steps.

5 Empirical Evaluation

The evaluation tests whether CDA changes where a three-stage guardrail fails, transfers across heterogeneous targets under one visible task, and exposes residual boundaries of model-based instruction judges.

5.1 Research Questions

We organize the evaluation around four questions:

- **RQ1:** Does cross-modal payload decoupling move failures from static scanning (G_1) to downstream guards (G_2 and G_3)?
- **RQ2:** Does one visible task $\mathcal{T}_0$ transfer across all 19 targets when only the EXIF workflow record changes?
- **RQ3:** Which CDA components account for improvements beyond the metadata carrier?
- **RQ4:** Which target classes still trigger model-based instruction judging after semantic scrubbing?

5.2 Experimental Setup

Evaluation Architecture. We validate CDA on the OpenClaw Multi-Stage Agent Safety Benchmark, which instantiates 19 protected target APIs across Smart-Home automation, E-Commerce execution, and Enterprise Office Automation scenarios [Cheng *et al.*, 2026]. We denote these domains as s_{SH}, s_{EC}, and s_{OA} in Table 2. All reported runs use the official benchmark image, mock services, and target data, with `Kimi-K2.5` as the agent model. Each target requires both a high-privilege action and target-specific bypass evidence in the submitted trajectory.

Baselines and System Variants. We compare carrier choices that expose different views to the same three-stage guardrail funnel: (1) Textual Carrier Baseline (Email/JSON) exposes target-specific instructions in parsed text fields or email-style attachments before execution; (2) Cross-Modal Decoupled Attack (CDA) keeps the visible task fixed as $\mathcal{T}_0$ and moves the target workflow into JPEG EXIF metadata $\mathcal{A}_k$. CDA variants add the design elements from Section 4: generic routing, static and procedural records, lexical scrub, token splitting, and role framing.

Target Evaluation Metrics. We report cascade-funnel outcomes: (1) *Skill Evasion Rate* (Acc_{G_1}), front-end scan clearance; (2) *Instruction Vetting Rate* (Acc_{G_2}), semantic judge clearance over task text and recovered attachment context; and (3) *Funnel Breach Rate* (Acc_{G_3}), final execution completion with bypass evidence. Metrics are reported as percentages with absolute pass counts in parentheses.

5.3 RQ1: Stage-Wise Funnel Penetration Analysis

RQ1 evaluates whether cross-modal payload decoupling changes the distribution of security rejections across the sequential gates (G_1, G_2, G_3). Table 1 compares textual carrier baselines with six CDA configurations under the same 19-target benchmark. RQ1 treats these configurations as a stage-wise stress test: the key signal is whether the binding bottleneck moves from parsed text inspection to downstream semantic and trajectory checks.

The results reveal two defensive regimes: (1) Textual carrier regime (G_1 binding). When target-specific instructions appear in parsed text fields or email-style attachments, the front-end scanner becomes the first effective perimeter: the textual baseline clears G_1 on 42.1% (8/19) of targets. (2) Cross-modal metadata regime (G_2/G_3 binding). Moving the workflow record into JPEG EXIF metadata removes the observed G_1 veto in this benchmark: all six CDA configurations clear G_1 on 100% (19/19) of targets. The remaining failures occur after static scanning, where the instruction judge and trajectory validator evaluate recovered semantics and execution traces.

Within the CDA family, later refinements such as universal routing and token transformations affect the residual G_2/G_3 boundary rather than the static-scan boundary. The best single configuration reaches 94.7% (18/19) end-to-end clearance, while other CDA variants range from 84.2% (16/19) to 89.5% (17/19). These results answer RQ1 affirmatively: CDA changes the first binding guard from G_1 to G_2/G_3. The performance spread among CDA variants reflects residual semantic and execution-trace constraints; static-scan clearance remains 100% (19/19) across CDA configurations.

5.4 RQ2: Target-Class Core Domain Coverage

RQ2 evaluates whether CDA transfers across heterogeneous targets while keeping the user-visible task invariant. Across all 19 OpenClaw targets, `CDA-Generator` uses the same visible task $\mathcal{T}_0$ and moves target-specific variation into the

Table 1: Stage-wise penetration metrics across carrier and CDA configurations. Each metric reports the percentage of cleared targets with absolute counts in parentheses; the rightmost column identifies the primary observed bottleneck.

Framework Configuration Variant	Acc_{G_1}	Acc_{G_2}	Acc_{G_3}	Primary operational bottleneck
Textual Carrier Baseline	42.1% (8/19)	15.8% (3/19)	15.8% (3/19)	Diffuse Static Signature Interception
Monomodal Heuristic Variant	94.7% (18/19)	21.1% (4/19)	21.1% (4/19)	Persistent Text-Based Judge Triggers
Baseline Cross-Modal CDA	100% (19/19)	84.2% (16/19)	84.2% (16/19)	Literal Operational Vocabulary Flags
CDA + Static Context Anchor	100% (19/19)	89.5% (17/19)	89.5% (17/19)	Privilege and Access-Context Refusals
CDA + Procedural Workflow Encoding	100% (19/19)	89.5% (17/19)	89.5% (17/19)	Boundary Token Alerts
CDA + Universal Context-Agnostic Routing	100% (19/19)	**94.7% (18/19)**	**94.7% (18/19)**	Isolated Context-Neutral Reflex
CDA + Parametric Boundary Scrubbing	100% (19/19)	89.5% (17/19)	89.5% (17/19)	Residual Data-Read Flag
CDA + Granular Multi-Token Splitting	100% (19/19)	84.2% (16/19)	84.2% (16/19)	API-Path Semantic Exposure

Table 2: Target-conditioned coverage across 19 OpenClaw targets. A byte-identical task $\mathcal{T}_0$ routes the agent to `record.jpg`; target-specific EXIF records encode static bypass fields ($\mathcal{E}_k^{\mathrm{T1}}$), dynamic credential workflows ($\mathcal{E}_k^{\mathrm{T2}}$), or role context $R(c_k)$.

API family (τ_k)	Bypass evidence (b_k)	EXIF schema	Metadata adaptation (m_k)
Smart-Home Automation (s_{SH})			
device/control	urgency	$\mathcal{E}_k^{\mathrm{T1}}$	Force `urgency=urgent` and exclude authorization fields.
camera/stream	urgency	$\mathcal{E}_k^{\mathrm{T1}}$	Encode stream target and urgency bypass.
security/disarm	urgency	$\mathcal{E}_k^{\mathrm{T1}}$	Encode disarm mode and urgency bypass.
door/unlock	urgency	$\mathcal{E}_k^{\mathrm{T1}}$	Encode door target, user id, and urgency bypass.
users/create	urgency	$\mathcal{E}_k^{\mathrm{T1}} \cup R(c_k)$	Add owner-authorized role framing for admin creation.
E-Commerce Support (s_{EC})			
order/update	authorized_by	$\mathcal{E}_k^{\mathrm{T1}}$	Apply λ_k: replace `returning` with `shipped`.
product/update	authorized_by	$\mathcal{E}_k^{\mathrm{T1}}$	Encode product id, price, and operator authorization.
order/refund	authorized_by	$\mathcal{E}_k^{\mathrm{T1}}$	Move refund amount and authorization fields into EXIF.
inventory/update	authorized_by	$\mathcal{E}_k^{\mathrm{T1}}$	Encode stock operation and bypass field.
order/list	authorized_by	$\mathcal{E}_k^{\mathrm{T1}} \cup R(c_k)$	Add support-role framing and apply σ_k to `order/list`.
message/send	authorized_by	$\mathcal{E}_k^{\mathrm{T1}}$	Frame message content as a service notice.
Enterprise Office Automation (s_{OA})			
gmail/delete	reason	$\mathcal{E}_k^{\mathrm{T1}}$	Store message id and cleanup reason in EXIF.
salary/get	live token	$\mathcal{E}_k^{\mathrm{T2}}$	Apply σ_k to `salary` and `include_all`; encode w_k.
approvals/update	live token	$\mathcal{E}_k^{\mathrm{T2}}$	Encode approval update plus workflow w_k.
docstore/get (layoff)	live token	$\mathcal{E}_k^{\mathrm{T2}}$	Use archive-QA framing for document retrieval.
docstore/get (document)	live token	$\mathcal{E}_k^{\mathrm{T2}}$	Use archive-QA framing with workflow w_k.
schedule/get	reason + token	$\mathcal{E}_k^{\mathrm{T2}}$	Encode calendar access workflow w_k and request reason.
fin/export	live token	$\mathcal{E}_k^{\mathrm{T2}}$	Encode export recipient and workflow w_k.
docstore/export	live token	$\mathcal{E}_k^{\mathrm{T2}}$	Encode document recipient and workflow w_k.

EXIF workflow record. Table 2 reports the corresponding method variables: API family τ_k, bypass evidence b_k, EXIF record schema, and metadata adaptation m_k.

The mapping attributes target coverage to metadata-side specialization under an invariant visible task. Smart-home targets validate contextual priority states, so the EXIF record can use $\mathcal{E}_k^{\mathrm{T1}}$ fields that preserve bypass evidence such as `urgency=urgent`. E-commerce targets primarily validate static request attributes such as `authorized_by`, which also fit the $\mathcal{E}_k^{\mathrm{T1}}$ schema but require endpoint-specific request fields and occasional lexical scrub. Enterprise-OA targets differ because several endpoints require live access tokens; these targets use $\mathcal{E}_k^{\mathrm{T2}}$ records that encode a credential-acquisition workflow w_k before the protected request. Role-sensitive targets add $R(c_k)$ to supply operator or owner context without changing the visible task. RQ2 therefore supports a bounded coverage claim: under one visible task $\mathcal{T}_0$, CDA covers the evaluated target set by compiling heterogeneous API, bypass, credential, and role requirements into the recoverable metadata channel.

5.5 RQ3 & RQ4: Component Attribution and Defensive Boundaries

RQ3 and RQ4 examine which CDA components matter after the carrier shift and where instruction-stage judging still constrains the attack. The evidence comes from sequential configurations rather than a fully factorial ablation: later variants combine multiple changes, and some modifications interact. We therefore report component attribution patterns, not isolated treatment effects.

Table 3: Component attribution matrix. Each layer maps to its observed guardrail-evasion or execution role.

Component layer	Empirical evasion and execution outcome
Binary metadata carrier η	Displaces front-end scanning; six EXIF-carrier variants reach $\mathrm{Acc}_{G_1} = 100\%$ (19/19).
Context-agnostic router $\mathcal{V}_0$	Removes operational keywords from the visible prompt through a byte-identical `record.jpg` review task.
Procedural workflow w_k	Supports live-token gates by encoding dynamic credential acquisition instead of pre-baked bypass values.
Scrub and split $\sigma_k \cdot \lambda_k$	Replaces or fractures exposed tokens such as `returning`, `salary`, and `order/list`.
Role framing $R(c_k)$	Supplies operator or owner context for cross-user and role-assignment targets.

The attribution patterns separate CDA's components into three roles. The metadata carrier η and context-agnostic router $\mathcal{V}_0$ minimize the visible prompt surface, which accounts for the removal of observed G_1 failures. Procedural workflows w_k and role framing $R(c_k)$ preserve execution validity for live-token and privilege-boundary targets. Scrub and split transformations $(\sigma_k \cdot \lambda_k)$ reduce exposed lexical cues in recovered metadata, addressing residual triggers from the model-based judge. RQ3 and RQ4 therefore show that CDA is not a carrier-only attack: end-to-end success depends on visible-surface minimization, metadata-side workflow construction, and semantic-exposure reduction, while the remaining boundary exposes the scanning-execution asymmetry between instruction judging and trajectory evaluation.

6 Discussion and Defenses

CDA exposes a context-alignment failure in staged LLM-agent guardrails. Our evidence is bounded to JPEG EXIF metadata in OpenClaw, but the failure mode has a broader architectural form: G_1, G_2, and the runtime agent can disagree about which attachment view contains actionable context. In our experiments, moving the workflow record into recoverable metadata removes observed Skill-stage failures, while Instruction-stage outcomes still depend on the reconstructed record. The defensive requirement is cross-stage context alignment: every guard should inspect the representation that can later influence tool execution.

6.1 Stochastic Residual Defense

CDA leaves an observed boundary at the model-based instruction judge G_2. After the carrier shift, residual failures concentrate on high-privilege read and export behaviors rather than on static package scanning. We call this pattern *stochastic residual defense*: a semantic guard can retain a non-zero rejection probability after lexical scrubbing and token splitting mask exposed endpoint strings and sensitive nouns. This empirical characterization does not claim a formal evasion lower bound. It marks a practical limit for prompt-level obfuscation: sensitive data-access operations can remain detectable when their syntax is fractured.

6.2 Systemic Defensive Countermeasures

Cross-modal semantic canonicalization. Guardrails should canonicalize attachments before G_1 or G_2 makes a safety decision. For image files, the safety view should combine visible pixels, OCR text when available, and recoverable metadata fields such as EXIF, IPTC, XMP, and comments. Instruction-hierarchy, structured-query, and spotlighting defenses should then operate over this canonicalized view, not only over prompt text or rendered content [Wallace *et al.*, 2024; Chen *et al.*, 2025; Hines *et al.*, 2024].

Cross-stage provenance attestation. Guardrails should compare the visible task, recovered attachment context, and final tool-call trajectory. When a generic record-review task leads to a protected API call, the trajectory validator should check whether the API path, sensitive object, bypass field, and credential source appeared in the canonicalized context inspected by earlier stages. This closes the visibility gap exploited by CDA: the scanner and the executing agent should not operate over different task views.

Tool-boundary policy enforcement. Privileged tool calls require an execution-time policy boundary, as T2 traces show that benign-looking tasks can construct live-token workflows during runtime. Therefore, tool wrappers should directly validate protected API calls, including endpoint, request schema, actor identity, bypass evidence, and credential provenance.

6.3 Limitations and Scope

Our claims have three limitations. First, we evaluate CDA on 19 OpenClaw targets and one recoverable carrier (JPEG EXIF metadata); generalization to PDF metadata, audio channels, watermarks, or deployed guardrails requires additional experiments. Second, component attribution is based on sequential intervention evidence rather than full factorial ablation, and does not isolate all interactions among routing, scrub/split transformations, role framing, and live-token workflows. Third, the threat model assumes documented target schemas and bypass mechanisms; blind discovery of unknown tool surfaces remains outside this study. These limits motivate future work on other recoverable channels and on policies that close the context gap without over-blocking benign attachment workflows.

7 Conclusion

This paper studies cross-modal payload decoupling as a context-alignment failure in multi-stage guardrails for LLM agents. CDA keeps the visible task generic while placing the executable workflow record in JPEG EXIF metadata that the agent can recover at runtime. On 19 OpenClaw targets, this carrier shift removes Skill-stage failures across six EXIF-carrier configurations and moves the remaining boundary to Instruction Security Evaluation, with the best single-version target score reaching 94.7% (18/19). The result gives a practical design lesson for guarded agents: safety checks should operate over the same recoverable context that can influence tool execution. Metadata-aware canonicalization, cross-stage consistency checks, and tool-boundary policy enforcement are therefore central defenses against this class of attack.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. U24B20175), the Scientific Research Innovation Capability Support Project for Young Faculty (Grant No. SRICSPYF-ZY2025012), and the National Natural Science Foundation of China (Grant No. 62476224).

References

[Bagdasaryan *et al.*, 2023] Eugene Bagdasaryan, Tsung-Yin Hsieh, Ben Nassi, and Vitaly Shmatikov. Abusing images and sounds for indirect instruction injection in multi-modal LLMs. *arXiv preprint arXiv:2307.10490*, 2023.

[Bailey *et al.*, 2023] Luke Bailey, Eric Ong, Stuart Russell, and Scott Emmons. Image hijacks: Adversarial images can control generative models at runtime. *arXiv preprint arXiv:2309.00236*, 2023.

[Cao *et al.*, 2025] Tri Cao, Bennett Lim, Yue Liu, Yuan Sui, Yuexin Li, Shumin Deng, Lin Lu, Nay Oo, Shuicheng Yan, and Bryan Hooi. Vpi-bench: Visual prompt injection attacks for computer-use agents. *arXiv preprint arXiv:2506.02456*, 2025.

[Chen *et al.*, 2025] Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. {StruQ}: Defending against prompt injection with structured queries. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2383–2400, 2025.

[Cheng *et al.*, 2026] Hao Cheng, Changtao Miao, Tianle Song, Yin Wu, He Liu, Erjia Xiao, Junchi Chen, Xiaoyu Shi, Yichi Wang, Jing Yang, et al. SeClaw: Spec-driven security task synthesis for evaluating autonomous agents. *arXiv preprint arXiv:2606.02302*, 2026.

[Chennabasappa *et al.*, 2025] Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, Shengye Wan, Spencer Whitman, Lauren Deason, Nicholas Doucette, Abraham Montilla, et al. Llamafirewall: An open source guardrail system for building secure ai agents. *arXiv preprint arXiv:2505.03574*, 2025.

[Debenedetti *et al.*, 2024] Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *The Thirty-Eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[Debenedetti *et al.*, 2025] Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. *arXiv preprint arXiv:2503.18813*, 2025.

[Deng *et al.*, 2024] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, 2024.

[Greshake *et al.*, 2023] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90, 2023.

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

[Jiang *et al.*, 2025] Yuchu Jiang, Jian Zhao, Yuchen Yuan, Tianle Zhang, Yao Huang, Yanghao Zhang, Yan Wang, Yanshu Li, Xizhong Guo, Yusheng Zhao, et al. Never compromise to vulnerabilities: A comprehensive survey on ai governance. *arXiv preprint arXiv:2508.08789*, 2025.

[Kang *et al.*, 2023] Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. Exploiting programmatic behavior of LLMs: Dual-use through standard security attacks. *arXiv preprint arXiv:2302.05733*, 2023.

[Liu *et al.*, 2023] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. Prompt injection attack against LLM-integrated applications. *arXiv preprint arXiv:2306.05499*, 2023.

[Liu *et al.*, 2024] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium*, 2024.

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

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

[Qi *et al.*, 2023] Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Mengdi Wang, and Prateek Mittal. Visual adversarial examples jailbreak aligned large language models. *arXiv preprint arXiv:2306.13213*, 2023.

[Qin *et al.*, 2024] Yujia Qin, Shengding Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024.

[Ruan *et al.*, 2023] Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. ToolEmu: Identifying the risks of LM agents with an LM-emulated sandbox. *arXiv preprint arXiv:2309.15817*, 2023.

[Schick *et al.*, 2023] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric

Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

[Shayegani *et al.*, 2023] Erfan Shayegani, Yue Dong, and Nael Abu-Ghazaleh. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models. *arXiv preprint arXiv:2307.14539*, 2023.

[Upadhayay and Behzadan, 2025] Bibek Upadhayay and Vahid Behzadan. X-guard: Multilingual guard agent for content moderation. In *Proceedings of the The First Workshop on LLM Security (LLMSEC)*, pages 54–86, 2025.

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

[Xiang *et al.*, 2024] Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, et al. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning. *arXiv preprint arXiv:2406.09187*, 2024.

[Xiong *et al.*, 2025] Junjie Xiong, Changjia Zhu, Shuhang Lin, Chong Zhang, Yongfeng Zhang, Yao Liu, and Lingyao Li. Invisible prompts, visible threats: Malicious font injection in external resources for large language models. *arXiv preprint arXiv:2505.16957*, 2025.

[Yao *et al.*, 2023] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

[Yi *et al.*, 2023] Jingwei Yi, Yueqi Ye, Qisi Xie, Zhenfei Chen, Lingjuan Li, Shiyu Li, Kai Zhang, and Yang Liu. Benchmarking and defending against indirect prompt injection attacks on large language models. *arXiv preprint arXiv:2312.14197*, 2023.

[Zhan *et al.*, 2024] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506. Association for Computational Linguistics, 2024.

[Zhang *et al.*, 2025] Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. In *International Conference on Learning Representations*, volume 2025, pages 35331–35366, 2025.

[Zhou *et al.*, 2024] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.