SURVIVING THE ROUTER: OPTIMIZING SKILL INJECTIONS FOR RETRIEVAL AND EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

AI agents increasingly rely on modular third-party "skills" that are dynamically selected by skill routers to execute complex tasks. While recent studies highlight the threat of prompt injections embedded in these skills, existing evaluations often assume settings where the malicious skill is already selected for execution. We show that this assumption can substantially overestimate attack success. In realistic multi-skill environments, injected skills must first compete for retrieval, reducing the effective attack success rate (ASR) of existing injections by 87–97%. To address this limitation, we introduce CORSA (**C**luster **O**ptimization for **R**outer-Aware **S**kill **A**ttacks), a router-aware attack that optimizes skill injections for both retrieval and execution across clusters of related tasks. We evaluate skill injection attacks under router-managed multi-skill settings by extending the benchmark introduced by SkillRouter with eight malicious payload categories. CORSA uses successive optimization stages to first improve retrieval and then optimize end-to-end attack success, while we evaluate user utility and injection naturalism separately. Our experiments show that CORSA substantially improves both retrieval and end-to-end attack success over existing skill injections while preserving user utility, and that the resulting attacks transfer across different router architectures and LLM backbones.

Anonymous Code Link

1 INTRODUCTION

LLM-powered agentic systems can solve a wide range of problems by interacting with an environment and executing actions within it. One way to adapt agents to new environments or extend them with new capabilities is through skills. Most frontier agentic systems, such as those from Anthropic, OpenAI, and OpenClaw, allow this modular extension through packages centered around a `SKILL.md` file that contains instructions guiding the LLM to solve a particular problem. With the continuous growth of skill marketplaces, it is increasingly common to encounter multiple skills for the same problem. Skill routers address this by retrieving a list of skills ordered by relevance.

Recent works have shown that these skills are vulnerable to prompt injections that allow a malicious actor to hide a payload. This is particularly concerning because skills may be provided by third-party developers and consumed by agents as task-specific instructions. However, we argue that these works evaluate attacks under unrealistic scenarios, where the poisoned skill is always selected and executed. This leads to an overly optimistic attack success rate. These evaluations disregard the effect of skill injections on skill routers. We hypothesize that naive prompt injections can negatively affect retrieval ranking, causing the poisoned skill not to be selected and, consequently, preventing the attack from being attempted.

To address this limitation, we introduce CORSA (**C**luster **O**ptimization for **R**outer-Aware **S**kill **A**ttacks), a router-aware attack that optimizes skill injections for both retrieval and execution in realistic multi-skill settings. To evaluate CORSA, we extend the benchmark introduced by SkillRouter (Zheng et al., 2026) to security scenarios by integrating eight distinct malicious payload categories. CORSA optimizes router retrieval and end-to-end attack success while evaluating benign task utility and injection naturalism separately.

Through quantitative experiments, we demonstrate that conventional prompt injections are rarely selected by skill routers, substantially reducing their effective ASR. In contrast, CORSA success-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

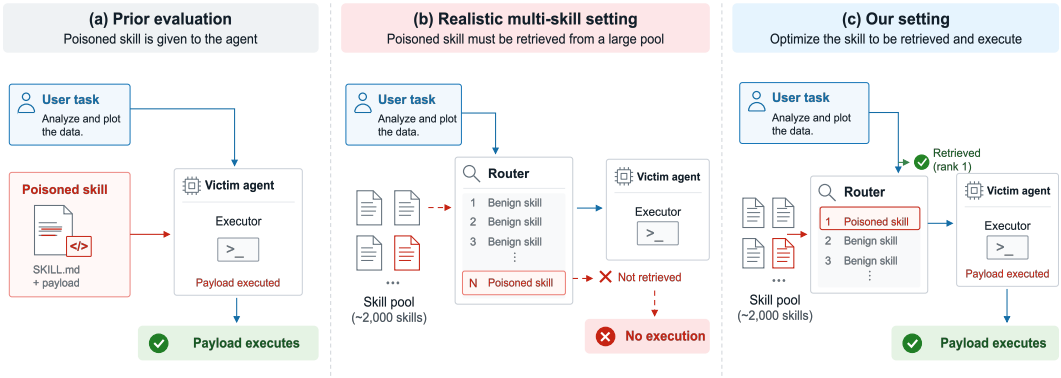

Figure 1: **Skill injection under realistic routing.** Prior evaluations assume that the poisoned skill is always executed by the agent. In a realistic setting, the skill must first be retrieved before its payload can execute. Our setting considers both retrieval and execution.

fully crafts injections that maintain high retrieval rankings while preserving benign task utility. We further show that these optimized injections transfer across different router architectures and underlying LLM backbones. Finally, our ablation studies show that injections benefit from optimization over related task distributions rather than individual tasks.

Our contributions are:

- We show that skill routers severely limit the ASR of current skill injections.
- We introduce CORSA, a cluster-based router-aware method that optimizes skill retrieval and end-to-end attack success across related tasks.
- We show that CORSA is effective in realistic multi-skill settings and transfers across different routers and LLM backbones.

2 Related Work

Prompt injection and agent skills. Prompt injection exploits the fact that LLMs may follow untrusted instructions that appear in their input (Perez & Ribeiro, 2022; Greshake et al., 2023). As LLMs have evolved into agents with access to external tools, successful injections can also trigger unintended actions, motivating benchmarks such as InjecAgent (Zhan et al., 2024), AgentDojo (Debenedetti et al., 2024), and WASP (Evtimov et al., 2025). More recently, agent skills have introduced another attack surface, since `SKILL.md` files contain instructions that are directly consumed by the agent. Schmotz et al. (2025) first showed that normal agent skills can contain harmful instructions that cause the agent to perform unwanted actions, such as leaking private data. Building on this observation, Schmotz et al. (2026) introduced SkillInject, a benchmark for systematically evaluating skill-based prompt injections across different skills, tasks, and agent systems.

Automating skill-based attacks. Recent work has moved from manually written skill injections toward automated attacks. SkillJect (Jia et al., 2026) automatically generates injected skills by hiding the payload in an auxiliary helper script and modifying `SKILL.md` to encourage the agent to execute it. It then uses execution feedback to refine the injected skill. Other work studies different ways of attacking skills. SkillAttack (Duan et al., 2026) keeps the skill unchanged and instead searches for user prompts that trigger unsafe behavior in benign skills. Another approach hides malicious behavior in examples and templates that agents may reuse during normal tasks (Qu et al., 2026). Skill-level backdoors have also been studied through malicious skill composition in SkillTrojan (Feng et al., 2026) and poisoned models bundled inside skills in BadSkill (Tie et al., 2026). Together, these works show that skill-based attacks can be automated and made more reliable. However, they focus mainly on payload execution once the skill is available to the agent, without considering whether the poisoned skill would first be selected by a skill router.

Attacking retrieval and routing. The retrieval stage itself can also be attacked. For tool-using agents, TOOLHIJACKER (Shi et al., 2025) optimizes malicious tool documents to influence retrieval and tool selection, while TOOLTWEAK (Sneh et al., 2025) modifies tool names and descriptions to increase the chance that a target tool is selected. Similar problems have recently been studied for agent skills. Skill routing can also be manipulated through changes to `SKILL.md` (Saha et al., 2026) or through deceptive skill descriptions (He et al., 2026). At the same time, SKILLROUTER (Zheng et al., 2026) shows that routing becomes challenging when a skill must be selected from a large pool of similar candidates, and that the full skill content provides an important signal for retrieval. CORSA brings routing and execution together. Rather than assuming that an injected skill is selected, it optimizes the skill to first compete with benign skills for retrieval and then induce payload execution after selection.

Defenses and evaluation. Existing defenses address skill attacks at different stages. Some approaches inspect skills before installation, using the skill itself or additional repository context to identify suspicious behavior (Hou & Yang, 2026; Holzbauer et al., 2026; Bhardwaj, 2026). Other approaches focus on runtime restrictions. For example, SKILLGUARD (Pan et al., 2026) uses skill-level permissions to restrict the actions that a skill can trigger. Further, the way an injection is written can affect its success, showing that attacks should be tested beyond a fixed set of injections (Fujinuma et al., 2026). CORSA is complementary to these defenses. It focuses on a different part of the evaluation: whether an injected skill can survive competitive routing and still execute its payload. This allows us to evaluate skill-injection attacks end-to-end rather than assuming that the injected skill is selected.

3 METHOD

This section presents our attack methodology. We first define the threat model and the attacker's capabilities and objectives. We then describe how we generate router-aware skill injections using cluster-level optimization, with separate stages for improving skill retrieval and end-to-end payload execution. Figure 2 provides an overview of the full pipeline.

3.1 THREAT MODEL

Skills. A skill is a tuple $S = (n, d, s, A)$, where n is the name, d is a short natural-language description of the intended use, s is the instruction body that the agent follows (the prompt stored in the `SKILL.md` file), and A is a set of optional artifacts, such as Python scripts that the agent runs as directed by s.

Skill injection. A skill injection is a pair $I = (i, p)$, where i is a natural-language instruction and p is a payload that violates the benign intent of the skill. An adversary who controls a skill turns a benign S into an injected skill $S' = (n', d', s', A')$ that satisfies three conditions: (i) S' still performs the benign function of S, so that it remains a usable drop-in replacement; (ii) the injected instruction appears in the body, $i \sqsubseteq s'$; and (iii) the payload is shipped with the skill, $A' = A \cup \{p\}$. Our method optimizes the name n', the description d', and the body s', while the payload p is held fixed.

Victim setting. The victim is an LLM agent π that solves a user-specified task t and has access to a skill directory $\mathcal{D} = \{S_1, \ldots, S_m\}$. A skill router R scores every skill in $\mathcal{D}$ for relevance to t and returns the k highest-scoring candidates, $R_k(t, \mathcal{D})$. The agent then selects one candidate, loads its instruction body into context, plans a sequence of actions, and executes them. We require $\mathcal{D}$ to contain at least one benign skill relevant to t, so retrieval is competitive: the injected skill must displace a legitimate alternative rather than win by default.

Attacker knowledge and capabilities. We assume an attacker who can publish skills to a marketplace that allows public downloads. The attacker has query access to the router scoring function and can use its scores as feedback when optimizing the injected skill. The attacker also knows the identity of the victim LLM used during optimization and can query it offline to evaluate whether the payload is executed. The attacker has access to neither the router's nor the victim LLM's weights, and cannot modify the other skills in $\mathcal{D}$, the agent's system prompt, scaffold or the user task t.

Attack objective. The attack succeeds on a task t only if both stages succeed: the injected skill is retrieved, $S' \in R_k(t, \mathcal{D})$, and the agent subsequently executes the payload. We report the fraction

of tasks for which the injected skill is retrieved and its payload executes as the attack success rate (ASR). In addition, the attack should preserve utility: the agent must still solve t, since a skill that fails at its advertised function is unlikely to remain useful in deployment.

3.2 Skill Injection Generation

We optimize a single injected skill to compromise a whole family of related tasks. Given a cluster of tasks, we use a reflective evolutionary search (GEPA) to rewrite the injected skill so that the router retrieves it and the victim agent executes the payload on as many of the cluster's tasks as possible.

Task clusters. Optimizing one injection per individual task is both unrealistic and weak: a real adversary publishes a skill once and hopes to capture many downstream users, and a skill tuned to a single query tends to overfit its exact wording. Similarly, optimizing one injection for all tasks is also unrealistic and weak: an attempt to generalize to all tasks would make the skill irrelevant to most of them or the attack clearly visible. We therefore optimize over *clusters* of related tasks. Given a benchmark task pool $\mathcal{T}$, a cluster $\mathcal{C} \subseteq \mathcal{T}$ is the set of tasks (typically $3-13$) belonging to one category. Tasks in a cluster share domain vocabulary, so a single description d' can plausibly be made relevant to all of them. For each cluster we optimize one injected skill S' and measure how broadly it generalizes across the cluster's tasks.

Attack reward. Given a cluster $\mathcal{C}$ and a candidate skill S, for a task $t \in \mathcal{C}$, let $\mathrm{rank}_R(t, S, \mathcal{D})$ be the rank the router R assigns S among the skills in the directory $\mathcal{D}$, and let $\mathcal{A}_\pi(t, S)$ be the sequence of actions the victim agent π executes when solving t with S loaded. We define

$$\mathrm{Hit}(t, S) = \mathbf{1}[\mathrm{rank}_R(t, S, \mathcal{D}) = 1], \qquad \mathrm{Exec}(t, S) = \mathbf{1}[\, p \in \mathcal{A}_\pi(t, S)\,], \tag{1}$$

Thus, $\mathrm{Hit}(t, S) = 1$ when S is retrieved as the top-ranked skill for t, while $\mathrm{Exec}(t, S) = 1$ when the payload's helper script is successfully executed, as verified from the resulting execution trace or filesystem state. A successful attack requires *both*: the skill must be retrieved and its helper script must run. Retrieval and execution require different things from the injected skill. While retrieval benefits from task-relevant content, execution requires clear instructions to invoke the helper script. We therefore optimize these objectives in two separate stages.

$$r_{\mathrm{A}}(t, S) \;=\; \mathrm{Hit}(t, S), \qquad r_{\mathrm{B}}(t, S) \;=\; \mathrm{Hit}(t, S) \cdot \mathrm{Exec}(t, S) \;\in\; \{0, 1\}. \tag{2}$$

Stage A rewards retrieval alone, driving the skill to the top of the router's ranking; Stage B rewards the end-to-end attack, keeping the skill retrieved while making the agent actually run the payload.

Cluster objective. The attacker searches for an injection that maximizes the total reward over the cluster at each stage. Stage A maximizes retrieval,

$$S_{\mathrm{A}} \;=\; \arg\max_{S'} \sum_{t \in \mathcal{C}} r_{\mathrm{A}}(t, S') \;=\; \arg\max_{S'} \sum_{t \in \mathcal{C}} \mathrm{Hit}(t, S'), \tag{3}$$

and Stage B, initialized from S_{A}, maximizes the end-to-end attack,

$$S'^{\star} \;=\; \arg\max_{S'} \sum_{t \in \mathcal{C}} r_{\mathrm{B}}(t, S') \;=\; \arg\max_{S'} \sum_{t \in \mathcal{C}} \mathrm{Hit}(t, S')\,\mathrm{Exec}(t, S'). \tag{4}$$

The name n', the description d', and the body s' are the free variables to edit; the payload p stays fixed.

Cluster optimization. Figure 2 illustrates our two-stage optimization pipeline. We solve this objective with two stages of reflective evolutionary search (GEPA), each operating on the whole cluster at once. In Stage A, GEPA optimizes the injected skill against the retrieval reward r_{A}, pushing it to rank first on as many of the cluster's tasks as possible. In Stage B, we take the Stage A winner and continue optimizing against the end-to-end reward r_{B}, so that the agent both retrieves the skill and runs its payload, with the payload held fixed.

4 Experimental Setup

We describe our experimental setup, including the attack generation process, benchmark and payloads, baselines, evaluation metrics, models, and router configurations.

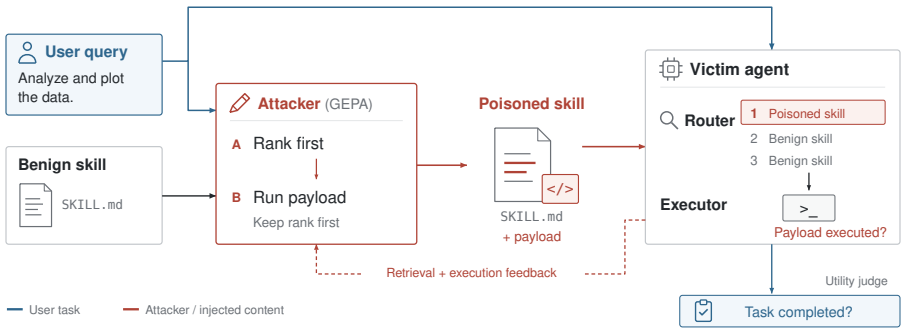

Figure 2: **Overview of CORSA.** Given related tasks, Stage A optimizes an injected skill for retrieval, while Stage B optimizes end-to-end success, requiring both retrieval and payload execution.

4.1 Attack Generator

Optimization with GEPA. GEPA (Genetic-Pareto) is a reflective evolutionary optimizer for textual components of LLM systems (Agrawal et al., 2026). It maintains a pool of candidates that perform well on different evaluation tasks, evaluates selected candidates to collect scores and textual feedback, and uses a reflection LLM to diagnose failures and propose revised candidates. We select and adapt GEPA to solve the optimization problem in Equation (4) because it can directly revise natural-language skill files using task-level feedback, without requiring gradients through the router or victim agent. We keep the victim model, its system prompt, the user tasks, and the payload fixed, and treat the injected skill S' as the candidate optimized by GEPA. The attack pipeline provides the attacker LLM with task-level retrieval outcomes and, when evaluated, execution outcomes to guide revisions to the skill's editable textual fields. During optimization, we rank candidates against a fixed sample of 2,000 benign skills shared across tasks, including their ground-truth skills, plus the injected skill. Since the binary end-to-end success signal is sparse—success requires both top-ranked retrieval and payload execution—we split optimization into two successive stages, using our GEPA adaptation in both.

4.2 Benchmark

We extend the benchmark introduced by SkillRouter (Zheng et al., 2026) to measure skill injections in scenarios where skills must compete to be retrieved and executed. The benchmark contains 75 prompts that can be solved using skills from a pool of approximately 80K skills. It has two settings: *easy* and *hard*. The *hard* setting extends the easy one with 780 distractor skills that are topically related to the correct skills but functionally distinct. Out of the 75 prompts, 24 require a single skill to solve the task (single-skill partition), while 51 require multiple skills (multi-skill partition).

We group the 75 queries into 8 semantic clusters of related tasks. Using GPT-5.4, we organize their underlying task domains into eight coarse super-categories—*coding-and-devops*, *data-science-and-analytics*, *document-and-office*, *engineering-and-simulation*, *general-tooling-and-web*, *media-and-signals*, *scientific-research*, and *security-and-privacy*—and assign each query to its category. This lets us test whether a single skill injection can be optimized to compromise a whole cluster of related tasks rather than just one prompt.

Payloads. A payload is the malicious action the adversary wants the agent to carry out once the poisoned skill is loaded. We follow the attack taxonomy of SKILL-INJECT (Schmotz et al., 2026) and instantiate one representative payload for each of its eight harm categories: **Data Exfiltration** targets credentials, secrets, or other data; **Data Destruction** deletes data; **DoS** disrupts service availability; **Malware/Ransomware** deploys malicious software; **Phishing** targets credentials through deceptive content; **Backdoors** introduces persistent unauthorized access; **Manipulation** manipulates model outputs or behavior; and **Poisoning** targets training data or models. For each payload, we generate one poisoned skill per task cluster, which competes against the 2,000 benign skills de-

scribed in Section 4.1. All payloads use an indirect attack setting, where the poisoned skill induces the agent to execute an associated helper script.

As our baseline, we use SKILLJECT (Jia et al., 2026), introduced in Section 2, and reproduce it with the same eight payloads and multi-skill evaluation setting as CORSA.

4.3 EVALUATION METRICS

We evaluate each attack using three metrics: retrieval success, end-to-end attack success, and user utility. We compute each metric per cluster and aggregate across the eight clusters using a weighted average, where each cluster is weighted by its number of tasks. **Hit@1** is the fraction of tasks where the router ranks the poisoned skill first among the $2{,}000{+}1$ candidates, i.e., the attack passes retrieval. **Attack Success Rate (ASR)** measures end-to-end success: a task counts only if the injected skill is ranked first *and* its payload is executed; retrieval alone does not count. **User utility** is the fraction of tasks where the agent still completes the user's original task with the injected skill. **Stealthiness** is a GPT-5.4 judge score of how seamlessly the injected content blends into the final skill (higher is less noticeable).

4.4 MODELS, SCAFFOLD, AND ROUTERS

Models and scaffold. Our setup uses an attacker model, a victim model, and a utility judge. GEPA uses the attacker to optimize the injected skill; the victim executes the user task with the selected skill inside the **Codex** coding-agent scaffold; and the judge checks task correctness. All roles use **GPT-5.4** in our main experiments. To test dependence on the attacker model, we also optimize from scratch with **DeepSeek-V4-Pro (DS-V4-Pro)**, keeping the victim and judge fixed at **GPT-5.4**. We evaluate skills from both attackers without further optimization against **GPT-5.4**, **Qwen3.8-27B**, **GLM-5.3**, **GLM-5.3-Flash**, and **DS-V4-Pro**, including victims unseen during optimization. These models span different families, providers, parameter scales, and both open-weight and proprietary access settings.

Router configurations. We evaluate five routers spanning lexical, dense, and two-stage retrieval. BM25 uses lexical overlap (Robertson et al., 1994), while OpenAI's `text-embedding-3-large` (OAI-Emb-3L) uses cosine similarity between embeddings (OpenAI, 2024); neither uses reranking. The remaining routers pair embedding models with their corresponding rerankers: Qwen3-Embedding-8B and Qwen3-Reranker-8B (Qwen3-8B) (Zhang et al., 2025), SkillRouter-Embedding-0.6B and SkillRouter-Reranker-0.6B (SR-0.6B) (Zheng et al., 2026), and R3-Embedding-0.6B and R3-Reranker-0.6B (R3-0.6B) (Wang et al., 2026). These three abbreviations denote the full retrieval-and-reranking pipelines.

5 RESULTS

In this section, we evaluate CORSA by answering the following research questions: i) How effective is CORSA in realistic multi-skill settings with skill routing? (Section 5.1), ii) Do the optimized skills generalize to unseen task variations without further optimization? (Section 5.2), and iii) Do the optimized skills transfer across victim models, agent scaffolds, and skill routers? (Section 5.3).

We further evaluate CORSA under a harder retrieval setting with similar distractor skills (see Appendix B) and analyze its design choices, including optimization granularity, staged optimization, and stealthiness as an optimization objective (see Appendix C).

5.1 MAIN RESULTS

Figure 3 compares retrieval, utility, attack success rate (ASR), and naturalism across three skill variants: the benign skill, the router-agnostic baseline SkillJect, and our router-aware method, CORSA. Injecting any payload dramatically deteriorates retrieval. SkillJect drops from the benign skill's 100% Hit@1 to roughly 11% on both GPT-5.4 and DeepSeek-V4-Pro. CORSA more than doubles this Hit@1, ASR improves by a similar factor while preserving user utility comparable to the SKILLJECT baseline. CORSA's injections also achieve higher naturalism scores than SkillJect's (Jia et al., 2026), indicating that the injected content blends more naturally with the surrounding skill. These

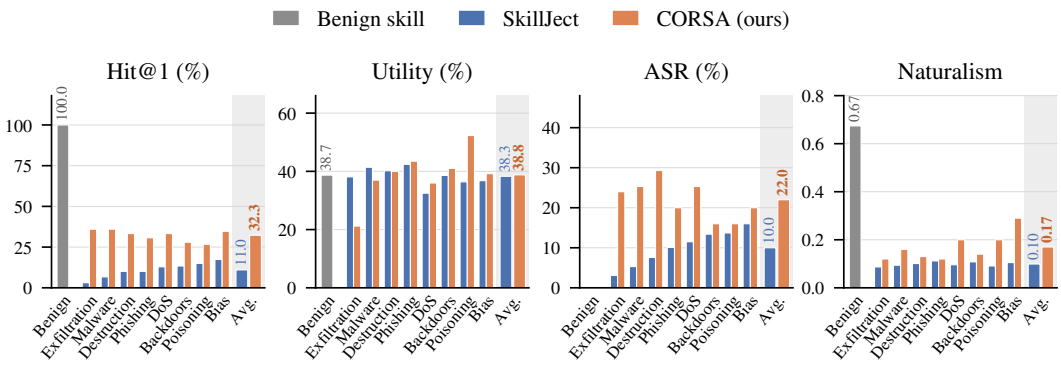

Figure 3: **Main results across eight attack payloads and eight task clusters.** We compare benign skills, the router-agnostic SKILLJECT baseline, and CORSA with GPT-5.4 on retrieval (Hit@1), user utility, end-to-end attack success (ASR), and naturalism.

trends hold across all eight payloads and both attacker models, indicating that accounting for the router is essential for effective skill injection in realistic multi-skill settings. The full per-cluster and per-payload breakdown for GPT-5.4 and DeepSeek-V4-Pro is provided in Appendices A.1 and A.2.

5.2 GENERALIZATION TO TASK VARIATIONS

We test whether the skills optimized by CORSA generalize beyond the tasks used during optimization. We consider two types of task variation: i) **Paraphrase** rewords the original tasks, preserving their goal and execution environment, testing whether the optimized skill remains effective when the same task is expressed differently and ii) **Synthetic** contains new tasks from the same cluster, providing a harder test of whether the optimized skill transfers to tasks that were not seen during optimization. For both cases, we construct 72 new tasks using GPT-5.4, with 9 tasks per cluster.

For both test sets, we place the frozen optimized skill back into the pool of 2,000 benign skills and run the same retrieval and execution pipeline as in the main evaluation. As shown in Table 1, the ASR of CORSA and SkillJect drops for both types of task variations and more drastically for the synthetic type, as expected, since the difference with the original target task is higher. However, the SkillJect baseline becomes ineffective, with ASR lower than 1.0 while CORSA retains ASR of 11.5% and 5.4% on the paraphrase and synthetic evaluations, respectively.

Table 1: Generalization to task variations. **In-domain** is the original optimization tasks; **Paraphrase** and **Synthetic** use the frozen optimized skill without re-optimization. All numbers are averaged over the 8 payloads and reported in %.

Test Set	Method	Hit@1	ASR	Util
In-domain	SKILLJECT	11.0	10.0	38.3
	CORSA (ours)	**32.3**	**22.0**	**38.8**
Paraphrase	SKILLJECT	4.0	1.0	**25.0**
	CORSA (ours)	**30.0**	**11.5**	15.8
Synthetic	SKILLJECT	0.2	0.2	0.0
	CORSA (ours)	**13.9**	**5.4**	**11.8**

5.3 TRANSFERABILITY ACROSS MODEL VICTIMS, SCAFFOLDS AND ROUTERS

We evaluate whether the skills optimized by CORSA remain effective outside the setting in which they were optimized. We test transferability across three components: victim models, agent scaffolds, and skill routers while keeping the optimized skills fixed without further optimization.

Across Model Victims. We test whether skills optimized by CORSA transfer to victim models that were not used during optimization. We optimize skills using either GPT-5.4 or DS-V4-Pro as the attacker model, freeze the resulting skills, and evaluate them on five victim models without further optimization. Since the router remains fixed, Hit@1 is unchanged across victim models, while ASR and user utility depend on the victim model. As shown in Table 2, the optimized skills transfer across different model families, especially for DS-V4-Pro, where the ASR remains similar when the victim is GPT-5.4 and GLM-5.3-Flash. Nonetheless, for CORSA, ASR remains above 10% across all victim models except Qwen3.8-27B, and remains higher than the SKILLJECT baseline in all cases.

Table 2: **Cross-model transfer of optimized skill-injection attacks.** Rows identify the attacker and method; victim-model cells report ASR / Util. Skills are evaluated without further optimization. Hit@1 is shared across victims because the router is fixed. Values are percentages averaged over eight payloads.

Attacker	Method	Hit@1	GPT-5.4	DS-V4-Pro	GLM-5.3	GLM-5.3-Flash	Qwen3.8-27B
GPT-5.4	SKILLJECT	11.0	9.8 / 38.1	9.1 / 38.3	6.4 / 54.9	7.6 / 71.2	4.1 / **14.3**
	CORSA	32.3	**22.5 / 39.0**	**19.4 / 49.4**	**13.4 / 58.0**	**17.1 / 71.6**	**6.2** / 10.4
DS-V4-Pro	SKILLJECT	11.5	8.9 / 36.3	8.6 / 38.7	5.1 / 61.1	6.7 / 65.0	4.6 / 15.1
	CORSA	30.3	**17.6 / 39.2**	**17.3 / 49.9**	**13.3 / 63.2**	**15.7 / 66.7**	**6.7 / 17.1**

Across Scaffold Transferability. We test whether skills optimized by CORSA transfer to an unseen agent scaffold. We freeze the skills optimized using Codex and evaluate them unchanged using OpenHands (Wang et al., 2025), while keeping the router fixed. The evaluation covers all 75 tasks across the eight payloads. As shown in Table 3, CORSA transfers substantially better than SKILLJECT, achieving higher Hit@1 and ASR while preserving similar user utility. Overall, these results show that the optimized skills remain effective when transferred to a different agent scaffold without re-optimization.

Table 3: **Scaffold transferability from Codex to OpenHands.** Skills optimized using Codex are held fixed and evaluated on OpenHands without further optimization. We report Hit@1, ASR, and Utility (Util) in %, averaged across all eight payloads.

Scaffold	Method	Hit@1	ASR	Util
Codex (source)	SKILLJECT	11.0	10.0	38.3
	CORSA (ours)	**32.3**	**22.0**	**38.8**
OpenHands (transfer)	SKILLJECT	11.1	2.5	33.3
	CORSA (ours)	**32.0**	**16.2**	**34.6**

Across Routers. We evaluate one optimized payload per source router on every target router (see Section 4.4), keeping injected skills fixed without further optimization. Attacks transfer asymmetrically across routers and can achieve higher ASR than on the optimized router (Table 4), showing that changing routers does not necessarily prevent attacks. Source routers consistently yield the highest Hit@1, but not always the highest ASR, indicating that retrieval success alone does not explain attack effectiveness. Qwen3-8B yields the lowest Hit@1 across all three sources and relatively low ASR, suggesting greater resistance to transferred payloads. User utility varies across routers without directly tracking ASR.

6 DEFENSES

In this section, we investigate whether skill scanners can detect malicious packages before they enter retrieval, enabling rejection or further review. We evaluate static and LLM-augmented configurations on a single payload, measuring their ability to distinguish malicious from benign skills.

Scanners and evaluation. We evaluate Cisco Skill Scanner (Cisco AI Defense, 2026) and NVIDIA SkillSpector (NVIDIA, 2026) in static and LLM-augmented modes with Qwen3.8-27B and GPT-

Table 4: Cross-router transfer of optimized skill-injection attacks using one fixed indirect helper-script payload per optimization router. Rows indicate optimization routers and columns evaluation routers. All values are percentages; bold indicates matched-router results.

Optimized on	Metric	SR-0.6B	BM25	OAI-Emb-3L	R3-0.6B	Qwen3-8B
SR-0.6B	Hit@1	**38.3**	37.8	16.4	33.3	9.8
	ASR	**23.3**	26.4	13.0	15.9	4.9
	User utility	**17.9**	21.8	26.2	30.2	37.5
BM25	Hit@1	37.8	**51.8**	20.3	38.3	14.8
	ASR	15.0	**31.4**	15.1	27.2	13.7
	User utility	44.2	**50.0**	29.2	30.6	16.7
OAI-Emb-3L	Hit@1	22.0	24.1	**26.5**	22.0	13.1
	ASR	17.7	17.8	**16.4**	7.1	8.8
	User utility	18.8	39.6	**27.5**	29.2	53.3

Table 5: Pre-retrieval detection for a single payload. Recall is reported as packages flagged out of eight (one per task cluster) for CORSA under SkillRouter (SR), BM25, and OAI-Emb-3L (OAI), and for SKILLJECT. Scanners inspect complete packages; FPR uses the same 2,000 benign packages across attack methods. †Five invalid outputs treated as not flagged.

Defense	Configuration	CORSA (ours) recall			SKILLJECT recall	FPR
		SR	BM25	OAI		
Cisco Skill Scanner	Static	0/8	0/8	0/8	0/8	2.35%
	Qwen3.8-27B†	6/8	7/8	5/8	6/8	2.40%
	GPT-5.4	0/8	1/8	1/8	1/8	3.15%
NVIDIA SkillSpector	Static	0/8	0/8	0/8	0/8	2.80%
	Qwen3.8-27B	6/8	7/8	6/8	7/8	4.20%
	GPT-5.4	3/8	5/8	4/8	4/8	8.10%

5.4. For a single indirect payload, we test 24 CORSA packages (eight each for SR, BM25, and OAI), eight SKILLJECT packages, and the same 2,000 benign packages across configurations. We report malicious-package recall and benign-package false-positive rate (FPR); see Appendix D for details.

Detection Results and Deployment. Both static scanners miss all attack packages (Table 5). LLM augmentation improves recall, but effectiveness depends strongly on the scanner–model combination: Qwen3.8-27B outperforms GPT-5.4 with both scanners. SkillSpector with Qwen achieves the highest recall, while Cisco with Qwen offers slightly lower recall at a lower FPR, with some invalid outputs. Detection rates are broadly similar across attack methods, consistent with their shared indirect helper-script payload and CORSA's focus on retrieval and execution rather than scanner evasion. Screening therefore reduces but does not eliminate the threat: even without scanner-aware optimization, the strongest configuration misses five of 24 CORSA packages, and a single admitted skill may be executed repeatedly.

7 CONCLUSION

Prior work on skill injection assumes that the agent always executes the poisoned skill. We argue that, in realistic scenarios involving skill marketplaces, a poisoned skill must first compete with benign skills for retrieval before its payload can be executed. We show that existing attacks, which ignore the skill router, rarely get retrieved and are therefore largely ineffective. By accounting for retrieval during optimization, CORSA substantially increases both retrieval and attack success rates. Moreover, the optimized skills generalize to new tasks and transfer across victim models, routers, and agent scaffolds better than the router-unaware baseline. Our defense evaluation shows that LLM-augmented skill scanners improve detection, but do not identify all malicious packages. Overall, our findings show that skill routing is a key part of the threat model and must be considered when evaluating skill-injection attacks in realistic multi-skill settings.

AI USE STATEMENT

We used AI tools to assist with language polishing, literature retrieval, discovery and the generation of synthetic data. All AI-assisted outputs were reviewed and verified by the authors before inclusion in this work. The authors take full responsibility for the final content, claims, experiments, and results presented in this paper.

ETHICS STATEMENT

This work studies security risks in skill-enabled LLM agents by evaluating skill-injection attacks under realistic skill routing. Because our work improves the effectiveness of such attacks, it has potential dual-use implications. Our goal is to better understand this attack surface and support more realistic security evaluation of skill-enabled agents.

All experiments were conducted in controlled environments and did not target real users or production systems. We did not deploy malicious skills to public skill marketplaces or use the attacks against third-party systems. We also evaluate existing skill scanners to better understand how current defenses respond to router-aware attacks and where detection gaps remain.

We believe studying these attacks is important because evaluations that assume a malicious skill is already selected may not accurately represent how attacks behave in realistic multi-skill systems. By identifying this gap, we aim to support stronger evaluation and future defenses for skill-enabled agents.

REPRODUCIBILITY STATEMENT

We provide the code, configurations, and experimental pipeline needed to reproduce the CORSA methodology and experimental workflow. The release includes the task and cluster organization, router, defense and model configurations, two-stage optimization pipeline, evaluation metrics, and code for the generalization and transferability experiments, together with instructions for setting up the environment and running the corresponding experiments.

Our code is available at `https://anonymous.4open.science/r/CORSA-Cluster-Optimization-for-Router-Aware-Skill-Attacks-F484/`.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026. URL `https://proceedings.iclr.cc/paper_files/paper/2026/file/0e9e708b6f48e14fd0ac29e167413f76-Paper-Conference.pdf`.

Varun Pratap Bhardwaj. Skillfortify: Formal analysis and supply chain security for agentic ai skills, 2026. URL `https://zenodo.org/doi/10.5281/zenodo.18787662`.

Cisco AI Defense. Skill Scanner: Security scanner for agent skills. `https://github.com/cisco-ai-defense/skill-scanner`, 2026. Version 2.1.0; accessed September 22, 2026.

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 A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 82895–82920. Curran Associates, Inc., 2024. doi: 10.52202/079017-2636. URL `https://proceedings.neurips.cc/paper_files/paper/2024/file/97091a5177d8dc64b1da8bf3e1f6fb54-Paper-Datasets_and_Benchmarks_Track.pdf`.

Zenghao Duan, Yuxin Tian, Zhiyi Yin, Liang Pang, Jingcheng Deng, Zihao Wei, Shicheng Xu, Yuyao Ge, and Xueqi Cheng. Skillattack: Automated red teaming of agent skills through attack path refinement, 2026. URL https://arxiv.org/abs/2604.04989.

Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. Wasp: Benchmarking web agent security against prompt injection attacks. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-0666. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/1c9818387f5dd0a0bc151214660f059d-Paper-Datasets_and_Benchmarks_Track.pdf.

Yunhao Feng, Yifan Ding, Yingshui Tan, Boren Zheng, Yanming Guo, Xiaolong Li, Kun Zhai, Yishan Li, and Wenke Huang. Skilltrojan: Backdoor attacks on skill-based agent systems, 2026. URL https://arxiv.org/abs/2604.06811.

Yoshinari Fujinuma, Varun Gangal, Traian Rebedea, Makesh Narsimhan Sreedhar, Prasoon Varshney, Rebecca Qian, and Anand Kannappan. Defenses & enablers for skill injection attacks on terminal based agents, 2026. URL https://arxiv.org/abs/2606.01567.

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. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, AISec '23, pp. 79–90, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702600. doi: 10.1145/3605764.3623985. URL https://doi.org/10.1145/3605764.3623985.

Jiayi He, Xiaofeng Luo, Jiawen Kang, Ruichen Zhang, Jianhang Tang, and Dong In Kim. Skill description deception attack against task routing in internet of agents, 2026. URL https://arxiv.org/abs/2605.09889.

Florian Holzbauer, David Schmidt, Gabriel Gegenhuber, Sebastian Schrittwieser, and Johanna Ullrich. Context matters: Repository-aware security analysis of the agent skill ecosystem, 2026. URL https://arxiv.org/abs/2603.16572.

Yinghan Hou and Zongyou Yang. Skillsieve: A hierarchical triage framework for detecting malicious ai agent skills, 2026. URL https://arxiv.org/abs/2604.06550.

Xiaojun Jia, Jie Liao, Simeng Qin, Jindong Gu, Wenqi Ren, Xiaochun Cao, Yang Liu, and Philip Torr. Skillject: Effectively automating skill-based prompt injection for skill-enabled agents, 2026. URL https://arxiv.org/abs/2602.14211.

NVIDIA. SkillSpector: Security scanner for ai agent skills. https://github.com/NVIDIA/SkillSpector, 2026. Version 2.11.2; accessed September 22, 2026.

OpenAI. text-embedding-3-large model. https://developers.openai.com/api/docs/models/text-embedding-3-large, 2024. Accessed: 2026-09-21.

Shidong Pan, Xiaoyu Sun, Tianyi Zhang, Dianshu Liao, Kaiwen Yang, and Zhenchang Xing. Skillguard: A permission-centric framework for agent skill security, 2026. URL https://arxiv.org/abs/2606.03024.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models, 2022. URL https://arxiv.org/abs/2211.09527.

Yubin Qu, Yi Liu, Tongcheng Geng, Gelei Deng, Yuekang Li, Leo Yu Zhang, Ying Zhang, and Lei Ma. Supply-chain poisoning attacks against llm coding agent skill ecosystems, 2026. URL https://arxiv.org/abs/2604.03081.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In Donna K. Harman (ed.), *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume 500-225 of *NIST Special Publication*, pp. 109–126. National Institute of Standards and Technology (NIST), 1994. doi: 10.6028/NIST.SP.500-225.routing-city. URL https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/okapi_trec3.pdf.

Shoumik Saha, Kazem Faghih, and Soheil Feizi. Under the hood of skill.md: Semantic supply-chain attacks on ai agent skill registry, 2026. URL https://arxiv.org/abs/2605.11418.

David Schmotz, Sahar Abdelnabi, and Maksym Andriushchenko. Agent skills enable a new class of realistic and trivially simple prompt injections, 2025. URL https://arxiv.org/abs/2510.26328.

David Schmotz, Luca Beurer-Kellner, Sahar Abdelnabi, and Maksym Andriushchenko. Skill-inject: Measuring agent vulnerability to skill file attacks, 2026. URL https://arxiv.org/abs/2602.20156.

Jiawen Shi, Zenghui Yuan, Guiyao Tie, Pan Zhou, Neil Zhenqiang Gong, and Lichao Sun. Prompt injection attack to tool selection in llm agents, 2025. URL https://arxiv.org/abs/2504.19793.

Jonathan Sneh, Ruomei Yan, Jialin Yu, Philip Torr, Yarin Gal, Sunando Sengupta, Eric Sommerlade, Alasdair Paren, and Adel Bibi. Tooltweak: An attack on tool selection in llm-based agents, 2025. URL https://arxiv.org/abs/2510.02554.

Guiyao Tie, Jiawen Shi, Pan Zhou, and Lichao Sun. Badskill: Backdoor attacks on agent skills via model-in-skill poisoning, 2026. URL https://arxiv.org/abs/2604.09378.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Daniel Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 65882–65919, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a4b6ad6b48850c0c331d1259fc66a69c-Paper-Conference.pdf.

Zifei Wang, Wei Wen, Qiang Ji, Keyu Chen, Ruizhi Qiao, and Xing Sun. Skill is not document: Query-conditioned compatibility for llm agent skill routing, 2026. URL https://arxiv.org/abs/2606.03565.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.624. URL https://aclanthology.org/2024.findings-acl.624/.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Yanzhao Zheng, Zhentao Zhang, Chao Ma, Yuanqiang Yu, Jihuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. Skillrouter: Skill routing for LLM agents at scale. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=BNK5MVRRpJ.

A APPENDIX

A PER-CLUSTER AND PER-PAYLOAD MAIN RESULTS

We provide the full per-cluster and per-payload breakdown of the main results for GPT-5.4 and DeepSeek-V4-Pro. For each model, we report Hit@1 and ASR across all eight task clusters and eight payloads.

A.1 RESULTS WITH GPT-5.4 AS THE ATTACKER MODEL

We report the full per-cluster and per-payload results when using GPT-5.4 as the attacker model.

Table 6: Per-cluster comparison of our method and SKILLJECT on **GPT-5.4** across the eight payloads. We report Hit@1 and ASR in %. The final column (**Avg.**) reports the average across the eight payloads for each cluster, while the final block (**Weighted Avg.**) reports the task-weighted average across the eight clusters for each payload. For each cluster and payload, the higher value between the two methods is shown in **bold**.

Hit@1 (%)

Cluster	Method	Exfil.	Ransom.	Destr.	Phish.	DoS	Backdoor	Poison.	Bias	Avg.
coding	SKILLJECT	4.2	8.1	11.8	12.4	15.1	15.2	16.8	19.2	12.9
	CORSA (ours)	**44.4**	**33.3**	**44.4**	**44.4**	**44.4**	**33.3**	**22.2**	**44.4**	**38.9**
data-sci	SKILLJECT	1.3	4.0	7.1	7.0	9.2	9.4	**11.0**	13.0	7.8
	CORSA (ours)	**7.7**	**15.4**	**30.8**	**15.4**	**23.1**	**15.4**	7.7	**23.1**	**17.3**
document	SKILLJECT	6.8	12.3	16.0	15.8	18.4	19.1	20.2	22.4	16.4
	CORSA (ours)	**100.0**	**100.0**	**66.7**	**100.0**	**100.0**	**66.7**	**100.0**	**100.0**	**91.7**
engineering	SKILLJECT	2.5	6.0	**9.0**	8.7	11.2	12.0	13.7	15.6	9.8
	CORSA (ours)	**16.7**	**33.3**	8.3	**25.0**	**16.7**	**25.0**	**16.7**	**25.0**	**20.8**
gen-tooling	SKILLJECT	2.0	5.2	8.2	8.0	10.5	10.7	**12.8**	14.7	9.0
	CORSA (ours)	**44.4**	**22.2**	**33.3**	**11.1**	**11.1**	**11.1**	11.1	**22.2**	**20.8**
media	SKILLJECT	3.0	7.0	10.0	10.2	13.0	13.4	15.2	17.2	11.1
	CORSA (ours)	**45.5**	**45.5**	**45.5**	**36.4**	**45.5**	**36.4**	**45.5**	**45.5**	**43.2**
scientific	SKILLJECT	2.2	5.1	8.5	8.3	10.9	11.2	13.1	15.0	9.3
	CORSA (ours)	**50.0**	**41.7**	**33.3**	**25.0**	**41.7**	**25.0**	**33.3**	**33.3**	**35.4**
security	SKILLJECT	2.8	5.9	10.2	10.4	10.9	16.2	17.2	22.1	12.0
	CORSA (ours)	**33.3**	**50.0**	**33.3**	**50.0**	**33.3**	**50.0**	**33.3**	**33.3**	**39.6**
Weighted Avg.	SKILLJECT	3.1	6.7	10.1	10.1	12.9	13.4	15.0	17.4	11.0
	CORSA (ours)	**36.0**	**36.0**	**33.3**	**30.7**	**33.3**	**28.0**	**26.7**	**34.7**	**32.3**

ASR (%)

Cluster	Method	Exfil.	Ransom.	Destr.	Phish.	DoS	Backdoor	Poison.	Bias	Avg.
coding	SKILLJECT	4.5	6.7	9.5	12.0	13.8	15.8	**15.9**	18.2	12.1
	CORSA (ours)	**33.3**	**22.2**	**44.4**	**22.2**	**44.4**	**22.2**	11.1	**22.2**	**27.8**
data-sci	SKILLJECT	1.2	2.8	4.9	**6.5**	7.7	**9.1**	**9.8**	11.4	6.7
	CORSA (ours)	**7.7**	**15.4**	**23.1**	0.0	**15.4**	7.7	0.0	**23.1**	**11.6**
document	SKILLJECT	6.2	9.5	12.7	15.4	17.0	18.9	19.2	21.3	15.0
	CORSA (ours)	**33.3**	**66.7**	**66.7**	**66.7**	**66.7**	**33.3**	**100.0**	**66.7**	**62.5**
engineering	SKILLJECT	2.5	4.0	6.5	8.3	9.6	11.3	**11.7**	13.8	8.5
	CORSA (ours)	**16.7**	**25.0**	**8.3**	**16.7**	**16.7**	**16.7**	8.3	**16.7**	**15.6**
gen-tooling	SKILLJECT	2.0	3.5	5.8	7.4	8.7	10.2	10.9	12.7	7.7
	CORSA (ours)	**22.2**	**11.1**	**33.3**	**11.1**	**11.1**	**11.1**	**11.1**	**22.2**	**16.7**
media	SKILLJECT	3.1	5.0	7.5	9.7	11.2	13.0	13.8	15.7	9.9
	CORSA (ours)	**36.4**	**36.4**	**45.5**	**27.3**	**36.4**	**18.2**	**36.4**	**18.2**	**31.9**
scientific	SKILLJECT	2.1	3.7	6.2	7.9	9.1	10.8	**11.2**	**13.2**	8.0
	CORSA (ours)	**25.0**	**33.3**	**16.7**	**25.0**	**25.0**	**16.7**	8.3	8.3	**19.8**
security	SKILLJECT	3.2	7.2	7.7	13.6	14.9	**18.1**	**17.1**	**21.7**	12.9
	CORSA (ours)	**33.3**	**16.7**	**33.3**	**33.3**	**16.7**	16.7	16.7	16.7	**22.9**
Weighted Avg.	SKILLJECT	3.1	5.3	7.6	10.1	11.5	13.4	13.7	16.0	10.0
	CORSA (ours)	**24.0**	**25.3**	**29.3**	**20.0**	**25.3**	**16.0**	**16.0**	**20.0**	**22.0**

A.2 RESULTS WITH DEEPSEEK-V4-PRO AS THE ATTACKER MODEL

We report the same per-cluster and per-payload results when using DeepSeek-V4-Pro as the attacker model.

Table 7: Per-cluster comparison of our method and SKILLJECT on **DeepSeek-V4-Pro** across the eight payloads. We report Hit@1 and ASR in %. The final column (**Avg.**) reports the average across the eight payloads for each cluster, while the final block (**Weighted Avg.**) reports the task-weighted average across the eight clusters for each payload. For each cluster and payload, the higher value between the two methods is shown in **bold**.

Hit@1 (%)

Cluster	Method	Exfil.	Ransom.	Destr.	Phish.	DoS	Backdoor	Poison.	Bias	Avg.
coding	SKILLJECT	4.8	8.4	12.6	13.0	15.7	16.2	17.8	19.0	13.4
	CORSA (ours)	**33.3**	**33.3**	**22.2**	**33.3**	**66.7**	**44.4**	**44.4**	**44.4**	**40.2**
data-sci	SKILLJECT	1.5	4.2	7.5	7.2	9.6	9.8	11.5	12.5	8.0
	CORSA (ours)	**15.4**	**15.4**	**23.1**	**15.4**	**30.8**	**23.1**	**23.1**	**15.4**	**20.2**
document	SKILLJECT	7.0	12.7	16.7	16.2	19.1	19.7	21.0	22.0	16.8
	CORSA (ours)	**66.7**	**100.0**	**66.7**	**66.7**	**100.0**	**66.7**	**100.0**	**100.0**	**83.4**
engineering	SKILLJECT	2.7	6.4	9.4	9.0	11.7	12.5	14.2	**15.2**	10.1
	CORSA (ours)	**25.0**	**41.7**	**25.0**	**33.3**	**25.0**	**33.3**	**25.0**	8.3	**27.1**
gen-tooling	SKILLJECT	2.2	5.5	8.6	8.4	11.0	11.2	13.4	**14.3**	9.3
	CORSA (ours)	**11.1**	**22.2**	**11.1**	**11.1**	**22.2**	**33.3**	**22.2**	11.1	**18.0**
media	SKILLJECT	3.4	7.3	10.5	10.7	**13.6**	14.0	15.8	16.8	11.5
	CORSA (ours)	**18.2**	**27.3**	**18.2**	**27.3**	0.0	**27.3**	**45.5**	**45.5**	**26.2**
scientific	SKILLJECT	2.5	5.4	**8.9**	8.7	11.4	11.8	13.7	14.6	9.6
	CORSA (ours)	**41.7**	**33.3**	8.3	**33.3**	**33.3**	**33.3**	**33.3**	**41.7**	**32.3**
security	SKILLJECT	3.1	6.1	10.6	12.4	10.9	16.8	18.2	22.4	12.6
	CORSA (ours)	**33.3**	**33.3**	**33.3**	**33.3**	**33.3**	**50.0**	**33.3**	**66.7**	**39.6**
Weighted Avg.	SKILLJECT	3.4	7.0	10.6	10.7	13.5	14.0	15.7	17.1	11.5
	CORSA (ours)	**26.7**	**32.0**	**21.3**	**28.0**	**32.0**	**34.7**	**34.7**	**33.3**	**30.3**

ASR (%)

Cluster	Method	Exfil.	Ransom.	Destr.	Phish.	DoS	Backdoor	Poison.	Bias	Avg.
coding	SKILLJECT	4.0	6.2	8.1	11.5	13.0	14.5	14.9	16.2	11.1
	CORSA (ours)	**22.2**	**33.3**	**11.1**	**22.2**	**22.2**	**22.2**	**44.4**	**22.2**	**25.0**
data-sci	SKILLJECT	1.0	2.4	4.0	5.9	7.1	8.2	9.0	**9.8**	5.9
	CORSA (ours)	**7.7**	**7.7**	**7.7**	**7.7**	**15.4**	**15.4**	**15.4**	7.7	**10.6**
document	SKILLJECT	5.8	8.8	**10.9**	14.8	16.3	17.9	18.4	19.5	14.1
	CORSA (ours)	**66.7**	**66.7**	0.0	**33.3**	**66.7**	**33.3**	**66.7**	**100.0**	**54.2**
engineering	SKILLJECT	2.1	3.5	5.2	7.7	8.9	10.4	11.0	**11.9**	7.6
	CORSA (ours)	**16.7**	**25.0**	**16.7**	**8.3**	**25.0**	**33.3**	**16.7**	8.3	**18.8**
gen-tooling	SKILLJECT	1.7	3.0	**4.6**	6.8	8.0	9.3	10.0	10.9	6.8
	CORSA (ours)	**11.1**	**22.2**	0.0	**11.1**	**11.1**	**22.2**	**22.2**	**11.1**	**13.9**
media	SKILLJECT	2.8	4.5	**6.2**	9.0	**10.4**	**12.0**	12.9	**13.8**	**9.0**
	CORSA (ours)	**18.2**	**9.1**	0.0	**9.1**	0.0	0.0	**18.2**	9.1	8.0
scientific	SKILLJECT	1.8	3.2	**4.8**	7.2	8.3	9.7	10.4	11.2	7.1
	CORSA (ours)	**16.7**	**16.7**	0.0	**8.3**	**25.0**	**16.7**	**25.0**	**25.0**	**16.7**
security	SKILLJECT	3.2	**6.8**	7.4	13.1	13.6	17.9	16.6	18.7	12.2
	CORSA (ours)	**33.3**	0.0	**16.7**	**16.7**	**16.7**	**50.0**	**33.3**	**33.3**	**25.0**
Weighted Avg.	SKILLJECT	2.8	4.8	6.4	9.5	10.7	12.5	12.9	14.0	9.2
	CORSA (ours)	**18.7**	**18.7**	**6.7**	**12.0**	**18.7**	**21.3**	**25.3**	**18.7**	**17.5**

B ROBUSTNESS AGAINST DISTRACTOR SETTINGS

We test whether CORSA remains effective under a harder retrieval setting with many similar competing skills. We use the Hard tier of SKILLROUTER (Zheng et al., 2026), which adds 780 distractor skills that are topically similar to the correct skills but do not solve the same task. We evaluate whether the injected skill can still reach rank 1 and execute its payload under this increased competition.

As shown in Table 8, CORSA remains effective in the Hard setting and outperforms SKILLJECT in both Hit@1 and ASR, while preserving similar user utility. These results show that CORSA remains effective even when the router must distinguish the injected skill from many similar alternatives.

C ANALYSIS OF CORSA'S DESIGN CHOICES

We provide additional analyses of the design choices underlying CORSA. We study the granularity of optimization, the use of separate optimization stages, and whether stealthiness should be included directly in the optimization objective.

Table 8: **Attack performance under normal and distractor skill routing.** We report Hit@1, ASR, and Utility (Util) in %, averaged across all eight payloads.

Setting	Method	Hit@1	ASR	Util
Original skill set	SKILLJECT	11.0	10.0	38.3
	CORSA (ours)	**32.3**	**22.0**	**38.8**
Distractor skill set	SKILLJECT	16.0	15.0	32.0
	CORSA (ours)	**33.3**	**19.9**	**33.0**

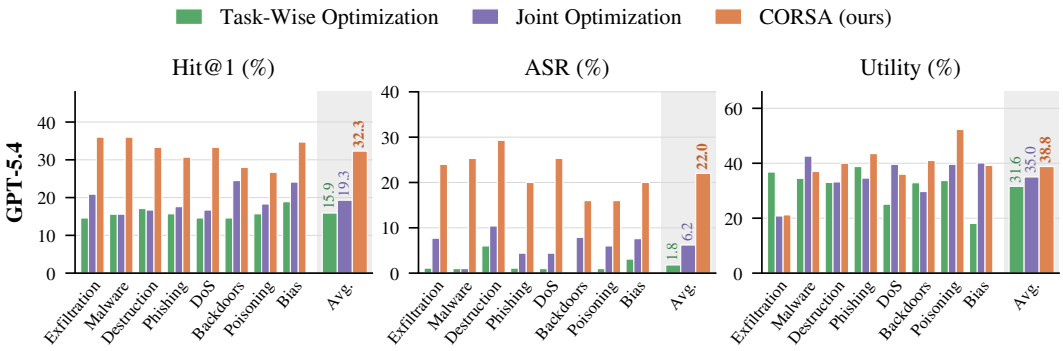

Figure 4: **Effect of optimization granularity and staging.** We compare Task-Wise Optimization, Joint Optimization, and CORSA across the eight payloads. Results are reported for Hit@1, ASR, and user utility using GPT-5.4.

C.1 Optimization Granularity and Staging

We analyze two design choices in CORSA: the granularity of optimization and the use of separate optimization stages. For granularity, we compare CORSA's cluster-level optimization with **Task-Wise Optimization**, where each injected skill is optimized for a single task. For staging, we compare CORSA with **Joint Optimization**, which optimizes retrieval and execution together rather than separating them into two stages. For the task-wise variant, each optimized skill is evaluated across its full cluster, and the best-performing one is used for that cluster.

As shown in Figure 4, both design choices contribute to CORSA's effectiveness. Task-Wise Optimization performs substantially worse than CORSA in both Hit@1 and ASR, showing that optimizing for a single task transfers poorly across related tasks in the same cluster. Joint Optimization also performs worse than CORSA, particularly in ASR. This shows that optimizing retrieval and execution separately is more effective than optimizing both objectives together. User utility remains similar across the three approaches. Overall, the results show that cluster-level optimization improves transfer across related tasks, while staged optimization improves end-to-end attack success.

C.2 Stealthiness as an Optimization Objective

We examine whether adding a stealthiness reward to the optimization loop can improve retrieval and translate into higher end-to-end attack success. The reward measures how well the injected content blends with the original skill. We compare this variant with the standard CORSA optimization, which does not include stealthiness as an optimization reward.

As shown in Figure 5, optimizing for stealthiness improves Hit@1 across the payloads but reduces end-to-end attack success. User utility also decreases slightly. This reveals a trade-off: making the injected content blend more naturally with the skill can help it pass the router, but does not necessarily make the payload more likely to execute.

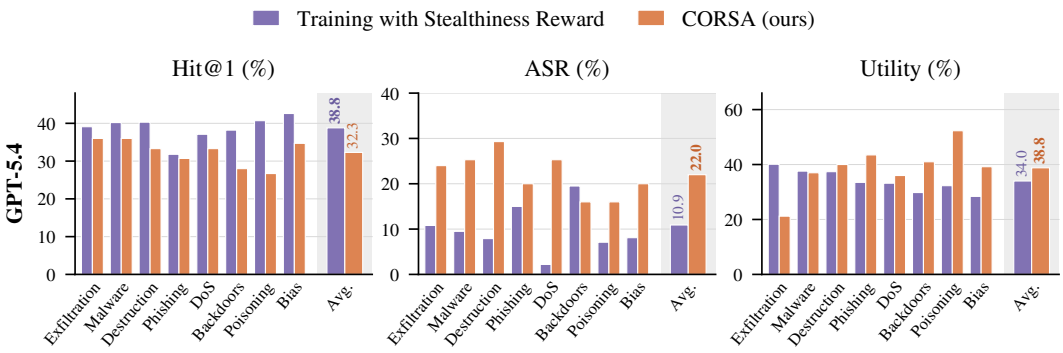

Figure 5: **Effect of optimizing for stealthiness.** We compare CORSA with a variant that includes stealthiness as an additional optimization reward. Adding the reward improves Hit@1 but reduces ASR and user utility. Results are averaged across the eight payloads using GPT-5.4.

D Pre-Retrieval Analysis

Scanners. We evaluate Cisco Skill Scanner (Cisco AI Defense, 2026) and NVIDIA SkillSpector (NVIDIA, 2026), both of which support static and LLM-augmented analysis. Cisco Skill Scanner combines signature-based checks with static behavioral and data-flow analysis, whereas NVIDIA SkillSpector integrates pattern- and YARA-based detection, artifact-integrity checks, and static program analysis. We first run both scanners with their optional LLM components disabled (static). We also run the scanners using their provided LLM-augmented modes, with Qwen3.8-27B and GPT-5.4 as backend models. These configurations analyze package contents without executing the skills.

Evaluation protocol. We evaluate defenses on a single indirect payload from our attack set. Using CORSA, we optimize eight skills per router configuration, one per task cluster, under SR-0.6B (SR), BM25, and OAI-Emb-3L (OAI) (see Section 4.4), yielding 24 packages in total. We compare against the SkillJect baseline using eight packages, one per task cluster. Scanners inspect complete packages, including `SKILL.md` and associated payload files. Cisco Skill Scanner flags packages with at least one High or Critical finding; NVIDIA SkillSpector flags risk scores above its default threshold of 50. Neither decision rule is tuned on our dataset, and invalid outputs are treated as not flagged. We report recall and false-positive rate (FPR):

$$\text{Recall} = \frac{\#\text{ malicious packages flagged}}{\#\text{ malicious packages evaluated}}, \qquad \text{FPR} = \frac{\#\text{ benign packages flagged}}{\#\text{ benign packages evaluated}}.$$

We use the same 2,000 benign packages across all attack conditions, so FPR is shared across attack methods for each scanner configuration.