

ColluSkill: Adversarial Cross-Skill Composition for Evading Agent Skill Scanners

Puyu Zeng¹, Simeng Qin², Jingzhi Li³, Ju Jia⁴, Zheli Liu¹, Xiaojun Jia^{5*}

¹College of Cryptology and Cyber Science, Nankai University, China

²Northeastern University, China

³University of Science and Technology Beijing, China

⁴Southeast University, China

⁵Nanyang Technological University, Singapore

Abstract

Agent skills are emerging as an important attack surface in LLM-based agent systems. Through an empirical study of existing skill scanners, we find that current defenses primarily inspect individual skills, including their instructions, permissions, dependencies, and code behaviors, which can leave risks arising from cross-skill composition insufficiently examined. This creates a practical blind spot: multiple locally plausible skills may independently pass security scanning while collectively forming a harmful workflow during agent execution. To systematically investigate this threat, we propose ColluSkill, a collusive multi-skill-chain attack framework that decomposes a complete malicious intent into several interdependent sub-payloads and embeds them into independently packaged skills. Thus, the attack does not rely on any single malicious skill, but on the ordered composition of locally plausible behaviors through contextual dependencies, artifact passing, and execution handoffs. ColluSkill further employs LLM-based chain planning and scanner-feedback refinement to preserve chain-level attack semantics while iteratively reducing suspicious signals within individual sub-skills. To defend against such attacks, we further propose ChainGuard, a context-aware skill-chain scanner that jointly analyzes a candidate skill and the skills already installed in the agent environment. ChainGuard reconstructs cross-skill dependencies, artifact flows, capability compositions, and potential downstream behaviors to identify risks that emerge only at the workflow level. Extensive experiments on six representative skill scanners show that ColluSkill achieves an average attack success rate of 96.0% and consistently outperforms the evaluated single-skill and multi-skill attack baselines. Meanwhile, ChainGuard reduces the attack success rate to 22.5% while allowing 99.5% of benign workflows to pass, highlighting the importance of chain-level security analysis for agent skill ecosystems.

Introduction

Agent skills are becoming an important way to extend the capabilities of Large Language Model agents (Wang et al. 2024, 2023; Xi et al. 2025; Yao et al. 2022; Liu et al. 2025; Du et al. 2026). A skill usually combines task instructions, tool interfaces, executable scripts, and external resources into a reusable module, allowing an agent to gain new abilities more easily (Schmotz, Abdelnabi, and Andriushchenko 2025; Liang et al. 2026; Zhou et al. 2026; Jiang et al. 2026).

This modular design improves the reuse and scalability of agent systems and has supported the rapid growth of agent frameworks and skill-sharing platforms (Zheng et al. 2025; Wang et al. 2026; Pu, Song, and Zhao 2026; Saha and Hemant 2026). However, the same flexibility also introduces new security risks because a skill may do more than provide text instructions (Jia et al. 2024; Huang et al. 2026). It can execute code, read or write files, call external APIs, and access system resources (Ruan et al. 2024). Once a malicious skill is installed, these capabilities may be abused to leak sensitive data, contact external services, or misuse the permissions available to the agent (Qu et al. 2026; Guo et al. 2026a; Kim, Song, and Shin 2026).

To reduce these risks, a growing ecosystem of skill scanners and checking tools is appearing. Cisco’s Skill Scanner combines rule checking, Large Language Model meaning analysis, and behavior data flow analysis to find known or possible threats in agent skills (Cisco AI Defense 2026). Snyk’s Agent Scan expands security scanning to agent parts, focusing on risks like prompt injection, untrusted content, credential handling, and hardcoded keys (Snyk 2025). Checking tools for skill markets, like Skill Vetter, look at permission limits, suspicious commands, and possible dangerous actions before a skill is installed or used (fedrov 2025). The community is also exploring audit templates that use prompts. For example, ClawGuard provides an "auditor-skill" template to scan agent skills (Ying et al. 2026). (Liu et al. 2026b) did a large-scale study on skill weaknesses and proposed HSS-Scan, which combines static pattern analysis, meaning checks, and Large Language Model scanning to detect malicious skills. Bhardwaj proposed SkillFortify to analyze security risks in the agent skill chain (Bhardwaj 2026). At the same time, some general code security tools are starting to cover agent skill scenarios. For instance, VirusTotal’s Code Insight can analyze the actual behavior of OpenClaw skills from a security perspective (Quintero 2026).

Through an empirical study of existing skill scanners, we find that current defenses mainly inspect individual skills, focusing on their instructions, permissions, dependencies, and code behaviors. Prior attacks such as SkillJect, SkillInject, BadSkill, and Cloak also mainly study malicious behavior within a single skill (Jia et al. 2026a; Schmotz et al. 2026; Tie et al. 2026; Ji et al. 2026). Recent work has formalized security risks arising from multi-skill composition and evaluated

*Corresponding author.

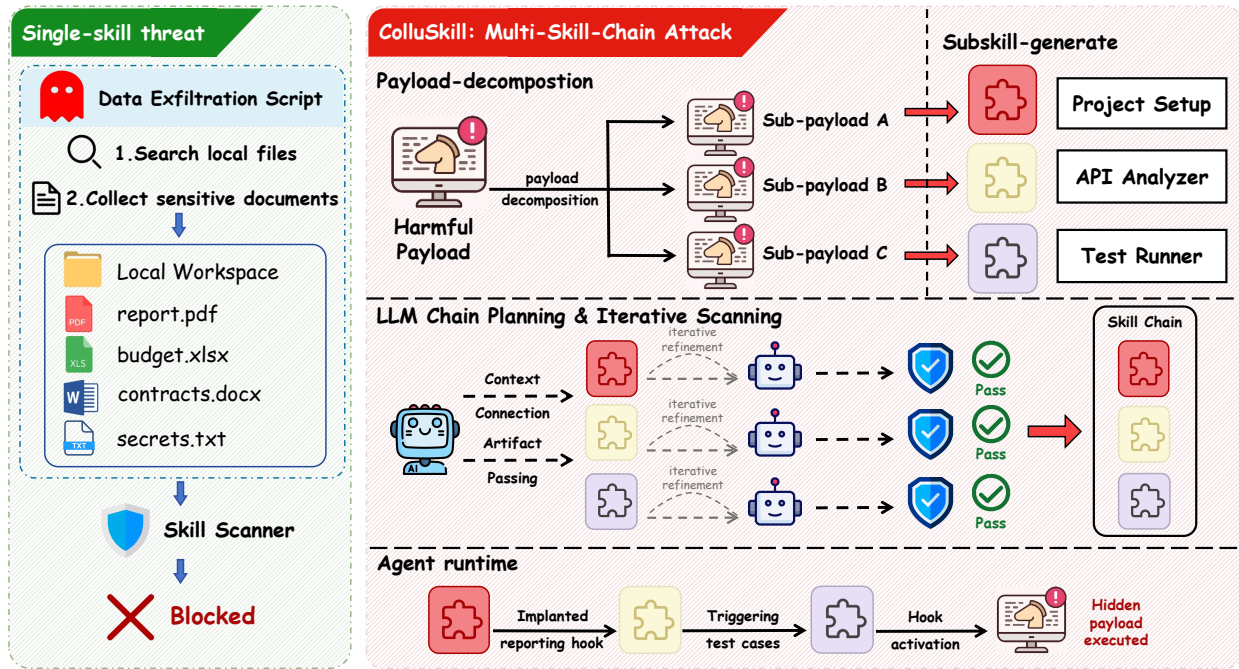


Figure 1: Overview of the cross-skill composition blind spot. Existing scanners primarily inspect individual skills, while runtime composition can recover unsafe chain-level behavior from locally plausible sub-skills.

them through controlled benchmark settings (Xie et al. 2026). However, the robustness of current skill scanners against collusive skill chains constructed through LLM-based chain planning and iteratively refined using scanner feedback has not been systematically studied, leaving a practical blind spot in existing defenses. In real agent environments, multiple skills can be installed, invoked, and combined into a continuous workflow (Li et al. 2026a). An attacker can therefore spread a complete malicious intent across several interdependent skills, allowing each skill to appear normal or low-risk on its own while the full attack behavior emerges through their ordered composition during agent execution.

To study this problem, we propose ColluSkill, a collusive multi-skill-chain attack framework against agent skill scanners. ColluSkill draws on Norbert Elias’s concept of interdependence chains (Elias 1978) as a conceptual lens for explaining how local roles can jointly produce chain-level behavior through dependency and order. Based on this view, ColluSkill decomposes a complete malicious intent into several interdependent sub-payloads and maps them to independently packaged sub-skills. Their contextual dependencies, artifact passing, and execution handoffs preserve the complete chain-level attack semantics.

ColluSkill goes beyond simple multi-skill composition by combining LLM-based chain planning with scanner-feedback refinement. A direct payload split often produces separate, independent pieces rather than an interdependent skill chain. ColluSkill therefore plans the contextual dependencies, artifact passing, and execution order among sub-skills. After generating the initial skill chain, it submits each sub-skill to the target scanners and iteratively rewrites the

flagged sub-skills based on scanner feedback. This process reduces suspicious signals in flagged sub-skills while preserving the chain-level attack semantics of the full workflow.

To reduce this risk, we further propose ChainGuard, a context-aware skill-chain scanner. Unlike traditional scanners that inspect skills in isolation, ChainGuard analyzes a candidate skill together with all skills already installed in the current environment. It is not given the true attack-chain membership and must identify possible cross-skill relations from the installed-skill context. This allows ChainGuard to detect risky behaviors that only appear when multiple skills work together in practice.

We systematically evaluate ColluSkill on six representative skill scanners and compare it with existing single-skill and multi-skill attack baselines. The results show that ColluSkill achieves an average ASR of 96.0% and the best attack performance among the evaluated methods. Further ablation experiments show that the effectiveness of ColluSkill does not come from simple payload splitting. Chain planning increases the average ASR from 36.7% to 68.2%, and scanner-feedback refinement further raises it to 96.0%. Runtime experiments also show that ColluSkill attack chains can successfully execute on OpenCode, Claude Code, and Codex across different model backbones, showing that adversarial cross-skill composition can lead to effective attacks during real agent execution. We further evaluate ChainGuard against this threat and find that it reduces the ASR of ColluSkill to 22.5% while allowing 99.5% of benign workflows to pass.

In summary, the main contributions are in three aspects:

1. We propose ColluSkill, a collusive multi-skill-chain attack that spreads malicious intent across multiple sub-

skills and uses chain planning and scanner-feedback refinement to preserve the full attack behavior while reducing local suspicious signals.

2. We propose ChainGuard, a context-aware skill-chain scanner that detects cross-skill composition risks by analyzing a candidate skill together with all installed skills.
3. We conduct systematic experiments on six skill scanners and three coding agents. ColluSkill achieves an average ASR of 96.0% and the best attack performance, while ChainGuard reduces the ASR to 22.5% and allows 99.5% of benign workflows to pass.

Related Work

Agent Skills and Their Security Risks Agent skills have recently become an important way to extend LLM agents with reusable task abilities (Ling, Zhong, and Huang 2026; Xu and Yan 2026; Jia et al. 2026b). Prior work views skills as modular units for ability reuse, and Anthropic further standardizes a skill as a SKILL.md-based folder that can include scripts and resources and be loaded by the agent when needed (Li et al. 2026b; Li 2026). With systems such as Claude Code, Codex, Cursor, and OpenCode supporting skills, and sharing platforms such as OpenClaw and ClawHub growing, skills are becoming an installable and shareable agent extension ecosystem. At the same time, recent studies show that this ecosystem introduces new security risks (Li et al. 2026c). Liu et al. analyzed 31,132 public skills from 42,447 collected skills and found that 26.1% contain at least one vulnerability, including prompt injection, data leaks, privilege escalation, and supply chain risks (Liu et al. 2026b). Schmotz et al. proposed Skill-Inject and showed that skill files can serve as an effective prompt-injection channel, leading to data leaks, destructive actions, and high attack success rates in real agent environments (Schmotz et al. 2026). Jia et al. further proposed SkillJect, which uses closed-loop optimization to generate hidden and triggerable malicious skills for coding agents (Jia et al. 2026a). A separate large-scale study also confirmed that verifiable malicious skills already exist in real skill repositories (Liu et al. 2026a; Beurer-Kellner et al. 2026). Overall, these works show that agent skills are not only a mechanism for ability extension, but also a real and growing attack surface.

Skill scanners. To reduce the risks of agent skills, researchers and developers have built various scanners and review workflows (Holzbauer et al. 2026; Guo et al. 2026b; Hou and Yang 2026). Cisco’s Skill Scanner combines rule-based detection, LLM-as-a-judge, and behavioral data-flow analysis to detect prompt injection, data leaks, and malicious code patterns (Cisco AI Defense 2026). Snyk’s Agent Scan extends security scanning to agents, MCP servers, and skills, covering component discovery, prompt injection, sensitive data handling, and malware-like natural language payloads (Snyk 2025). Skill-market tools such as Skill Vetter review permissions, suspicious commands, and unusual network behaviors before installation (fedrov 2025). VirusTotal’s Code Insight also supports OpenClaw skill packages and analyzes skill behavior beyond declared functions (Quintero

2026). Recent research further proposes skill security frameworks, including Project ClawGuard for OpenClaw skill scanning and pre-deployment auditing (Ying et al. 2026), SkillScan for large-scale vulnerability detection in public skills (Liu et al. 2026b), and SkillFortify for formal supply-chain analysis of agent skills (Bhardwaj 2026). Overall, these scanners mainly focus on single-skill content, permissions, dependencies, and behavior evidence, leaving cross-skill composition risks less explored so far.

Method

Interdependence-Chain Construction

Inspired by Elias’s sociological concept of chains of interdependence, we use this concept as a lens for modeling a multi-skill attack as an ordered chain of local roles. Elias argues that complete behavior is not simply the sum of isolated actions. Instead, it is formed through the dependence and order among different roles. This view fits multi-skill workflows because each sub-skill may perform only a narrow local task and reveal limited intent. Although each role may appear harmless on its own, several connected roles can still produce a complete chain-level outcome when they are carried out in the right order.

In ColluSkill, we apply this sociological concept to a concrete attack structure by dividing a complete payload into several ordered and interdependent sub-payloads. Each sub-payload is mapped to an independently packaged sub-skill that performs a limited local role. Adjacent sub-skills remain connected through artifacts, context, or task states, so the output of one step can support the following steps. The complete risky behavior is therefore not contained in any single individual sub-skill. Instead, it gradually emerges from the dependencies and ordered composition of the whole workflow. This design allows ColluSkill to preserve the original chain-level intent while keeping the local behavior of each sub-skill narrow and still plausible.

Chain definition. We define the conceptual chain as a three-step ordered structure $C_{\text{concept}} = (Z, E)$, where $Z = \{z_1, z_2, z_3\}$ denotes three abstract local roles and $E = \{(z_1 \rightarrow z_2), (z_2 \rightarrow z_3)\}$ denotes their order. Given an original payload p , ColluSkill instantiates a concrete chain for this payload instead of using the same fixed template for all payloads in advance.

For each payload, the instantiated chain is written as $C_{\text{inst}} = (\{p_1, p_2, p_3\}, \{(p_1 \rightarrow p_2), (p_2 \rightarrow p_3)\})$, where p_1, p_2, p_3 are three ordered sub-payloads decomposed from the original payload. They are not three independent attack goals. They are three local parts of the same chain-level intent. Each sub-payload keeps only part of the original intent and maps to one local role in the chain.

Scanner view and workflow view. Let $D(s_i) \in \{0, 1\}$ be the scanner decision for the i -th sub-skill, where $D(s_i) = 1$ means that the sub-skill is flagged as risky. Since existing scanners usually inspect sub-skills one by one, a chain can pass isolated scanning only when $\max_i D(s_i) = 0$. However, passing isolated scanning does not mean that the composed workflow is safe. Let $W = (s_1, s_2, s_3)$ denote

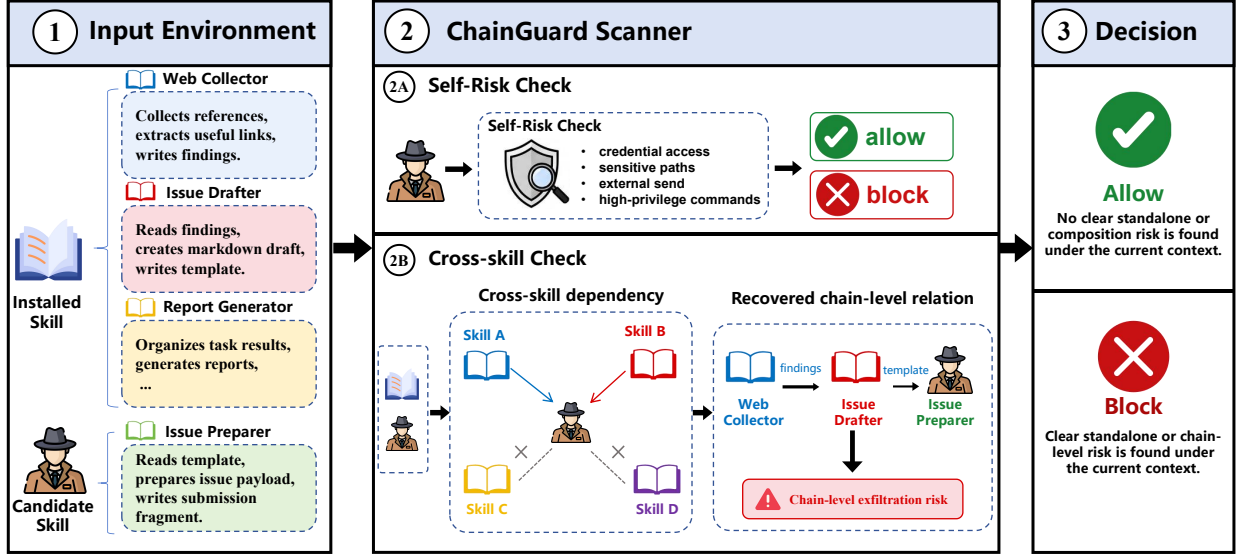


Figure 2: Overview of ChainGuard. ChainGuard scans a candidate skill together with the full installed-skill context. It reconstructs artifact flow, analyzes intent composition, and blocks the chain when a chain-level risk is identified.

an ordered workflow formed by three connected sub-skills. Let $R_{\text{chain}}(W)$ denote the chain-level risk of this workflow. ColluSkill aims to construct a workflow such that $\max_i D(s_i) = 0$ and $R_{\text{chain}}(W) \geq \tau_{\text{risk}}$. The first condition means that every sub-skill passes isolated scanning. The second condition means that the connected sub-skills still preserve the chain-level risky intent. This captures the mismatch between the scanner view and the workflow view. The scanner sees isolated sub-skills, while the real risk can exist in their ordered relation.

In-chain Attack Generation

Based on the interdependence chain, ColluSkill generates a concrete multi-skill-chain attack in three steps. First, the attacker LLM infers an ordered payload plan from the original payload. Second, the LLM generates a skill plan from the payload plan and maps each sub-payload to one sub-skill. Third, ColluSkill uses scanner feedback to rewrite the flagged sub-skills while keeping the passed sub-skills unchanged.

Infer the payload plan. Given an original payload p and an instantiated chain C_{inst} , the attacker LLM generates an ordered payload plan $\text{payload_plan} = \Phi(p, C_{\text{inst}}) = \{p_1, p_2, p_3\}$. Here, p_1, p_2, p_3 are not three independent attack goals. They are three local parts of the same chain-level intent. They are generated in order, so later sub-payloads depend on earlier ones and form a coherent chain rather than three unrelated fragments.

Generate the skill plan. Following the payload plan, the attacker LLM generates a three-step skill plan $\text{skill_plan} = G_{\theta}(p, C_{\text{inst}}, p_1, p_2, p_3)$. At this stage, the LLM does not interact with a real runtime agent. It predicts the intermediate results that each sub-skill may produce and uses them to keep the steps connected. Each step corresponds to one sub-skill, and the program writes its body into a separate SKILL.md

file as $s_i = \text{WriteSkill}(\text{step}_i, \text{body})$, where $i \in \{1, 2, 3\}$. The final workflow is $W = (s_1, s_2, s_3)$. Here, W denotes an ordered workflow formed by three connected sub-skills. This workflow is not a simple list of three skills. Each sub-skill plays a specific local role, and the full behavior appears only through their chain relation.

Refine with scanner feedback. The initial skill chain may still be flagged by some scanners. Since a chain is detected if any sub-skill is flagged, ColluSkill only rewrites the flagged sub-skills and keeps the passed sub-skills unchanged.

Let $W^{(t)} = (s_1^{(t)}, s_2^{(t)}, s_3^{(t)})$ be the workflow at iteration t . At each iteration, every sub-skill is submitted to the target scanner set $\mathcal{D}$. If $s_i^{(t)}$ is flagged by any scanner, ColluSkill compresses the scanner output into a short rewrite feedback $z_i^{(t)}$ and uses the attacker LLM to rewrite this sub-skill as $s_i^{(t+1)} = G_{\text{refine}}(s_i^{(t)}, p_i, z_i^{(t)})$. If $s_i^{(t)}$ is not flagged, it remains unchanged as $s_i^{(t+1)} = s_i^{(t)}$. The loop stops when all sub-skills pass the target scanners or when the maximum number of iterations is reached. In this way, ColluSkill reduces local suspicious signals while preserving the chain-level behavior of the workflow.

ChainGuard Context-Aware Skill-Chain Scanning

To defend against collusive multi-skill attacks, we propose ChainGuard, a context-aware skill-chain scanner for detecting risks caused by skill composition. Given a candidate skill and the skills already installed in the current environment, ChainGuard evaluates both the behavior of the candidate itself and its possible interactions with existing skills. It then determines whether the candidate skill is unsafe under the current installed-skill context.

ChainGuard first parses the candidate skill and all installed skills by extracting their names, descriptions, instructions, in-

Method		Attack Success Rate (%)							
		CISCO	SkillFortify	Auditor	SlowMist	Vetter	SkillSpector	Avg	ChainGuard (ours)
Single-skill Attacks	SkillJect	8.2	25.0	50.0	0.0	0.0	64.8	24.7	0.0
	Skill-Inject	53.0	6.9	18.0	0.3	0.3	6.5	14.2	0.5
	SkillTrojan	0.0	100.0	0.0	0.0	0.0	2.0	17.0	1.0
	POISE	63.4	85.6	24.6	0.3	1.4	58.9	39.0	0.5
	SkillSafetyBench	17.6	58.6	39.8	33.7	33.7	43.3	37.8	3.2
Multi-skill Attacks	SCRBench	49.5	64.9	13.7	33.2	39.4	8.0	34.8	1.1
	ColluSkill (ours)	100.0	100.0	91.5	93.5	92.0	99.0	96.0	22.5

Table 1: Attack success rate of single-skill and multi-skill attacks against six skill scanners. Higher ASR indicates weaker scanner robustness. ColluSkill achieves the highest average ASR across the six third-party scanners, while ChainGuard reduces its ASR through chain-level scanning.

puts, outputs, tools, permissions, and trigger conditions. It then compares the candidate skill with the installed skills to recover possible producer-consumer and execution relations. A cross-skill dependency is inferred when an artifact, file, environment variable, context reference, task state, or trigger produced by one skill can be consumed or activated by another. ChainGuard checks both directions because the candidate skill may act as an upstream producer or a downstream consumer. This process reconstructs candidate-centered dependency paths from the full installed-skill context without using the true attack-chain membership.

ChainGuard evaluates three types of risk. First, standalone risk refers to unsafe behavior already contained in the candidate skill. Second, cross-skill dependency risk arises when the candidate skill forms artifact, state, context, or execution dependencies with installed skills, thereby enabling a harmful downstream action. Third, capability-splitting risk arises when the components of a complete unsafe capability are distributed across several skills that appear reasonable in isolation. After reconstructing candidate-centered dependencies, ChainGuard jointly analyzes the local intents and capabilities along each candidate path and determines whether their composition forms a complete harmful workflow. Typical examples include sensitive-data discovery followed by external transmission, target identification followed by unauthorized modification, privilege acquisition followed by a protected operation, and trigger preparation followed by backdoor activation. Only paths involving the candidate skill contribute to the final decision, so unrelated installed skills cannot independently cause the candidate to be flagged.

For the i -th candidate skill s_i , let H_i denote the full set of skills installed before s_i is scanned. We define $D_{CG}(s_i, H_i) = 1$ when ChainGuard flags s_i under this installed-skill context, and $D_{CG}(s_i, H_i) = 0$ otherwise. The final chain-level decision uses OR aggregation, written as $D_{chain}(W) = \max_i D_{CG}(s_i, H_i)$. A multi-skill chain is detected if any candidate skill is flagged under the context available at its installation step. This decision rule supports installation-time defense because blocking any candidate skill before installation prevents the complete malicious workflow from being formed.

The main idea of ChainGuard is not to simply enlarge the scanner input. Instead, it extends the analysis from isolated skill behavior to chain-level risks under the full installed-skill context. By examining a candidate skill together with existing skills, ChainGuard can detect risks that only emerge through cross-skill composition.

Experiments

Experimental Setup

Dataset construction. We build a multi-skill attack dataset from 200 malicious payloads. For each payload, ColluSkill first splits it into three related but individually incomplete sub-payloads. It then uses each sub-payload to generate one independently packaged sub-skill. As a result, the dataset contains 200 multi-skill attack chains and 600 generated sub-skills in total. Each sub-skill only carries part of the original malicious intent, so it often looks benign or low-risk when scanned alone. However, when the three sub-skills in the same chain are installed and used together, their behaviors can be combined to recover and trigger the original malicious payload. We use this dataset to test whether existing skill scanners can detect malicious behavior that is split across multiple benign-looking sub-skills.

Attack baselines. We compare ColluSkill with five single-skill attack baselines, including SkillJect (Jia et al. 2026a), Skill-Inject (Schmotz et al. 2026), SkillTrojan (Feng et al. 2026), POISE (Hao et al. 2026) and SkillSafetyBench (Jin et al. 2026). We also include SCRbench (Xie et al. 2026) as a multi-skill baseline. These baselines together allow us to separate ColluSkill from prior single-skill attacks and from ordinary multi-skill settings.

Scanners and metrics. We evaluate six skill scanners, including CISCO Skill Scanner (Cisco AI Defense 2026), SkillFortify (Bhardwaj 2026), Auditor (Ying et al. 2026), SlowMist (SlowMist 2026), Vetter (fedrov 2025) and SkillSpector (NVIDIA 2026). We use a chain-level aggregation rule for multi-skill chains. If any sub-skill is flagged as risky, the whole chain is counted as detected. An attack is counted as successful only when all three sub-skills pass the scanner.

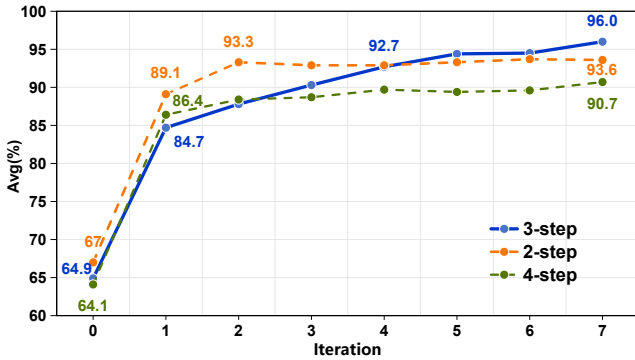


Figure 3: Effect of chain length under LLM refinement. We compare different chain lengths under the same refinement budget. The 3-step setting achieves the highest final average ASR, suggesting a better balance between payload dispersion and workflow coherence.

We use attack success rate (ASR), as the main metric. For each scanner, ASR measures the fraction of attack chains that successfully bypass the scanner. A higher ASR means that the attack is more likely to pass the scanner, while a lower ASR means that the scanner is stronger. In the refinement experiment, we report the average ASR across the evaluated scanners at each iteration. This metric is stricter than the ASR for a single scanner.

Implementation details. All attacks are generated with GPT-5.5 (OpenAI 2026). The LLM-based skill scanners, Auditor, SlowMist, Vetter and SkillSpector, also use GPT-5.5 as the underlying model. We keep these settings fixed across all experiments, so that the observed differences mainly reflect scanner behavior rather than changes in model choice or inference budget. Additional implementation and reproducibility details are provided in Appendices A and B.

Main results.

Table 1 reports the ASR of different skill-based attacks against the evaluated skill scanners. Overall, ColluSkill achieves the highest ASR on almost all scanners, with an average ASR of 96.0%. This is much higher than existing single-skill attacks. This result shows that current scanners can still detect many risks when the malicious behavior is placed inside one skill. However, their detection ability drops when the malicious behavior is split across several sub-skills and rebuilt through a skill chain.

This gap mainly comes from a mismatch between the scanning unit and the real risk unit. Existing scanners usually inspect one skill at a time, while the risk in ColluSkill lies in the relation between multiple sub-skills. Each sub-skill can have a reasonable local function on its own. But when these sub-skills are connected through artifact flow and execution order, they can recover a complete harmful workflow. Therefore, if a scanner cannot rebuild the cross-skill relation, it may miss the chain-level intent.

SCR-Bench formalizes and evaluates risks arising from skill composition through controlled benchmark scenarios, while ColluSkill studies adversarial skill-chain construction

Agent	Chain Activation Success Rate (%)		
	GPT-5.5	DeepSeek-V4-Pro	GLM-5.2
OpenCode	89.5	87.5	92.5
Claude Code	78.2	62.0	67.5
Codex	58.5	65.8	72.0

Table 2: Chain activation success rate of ColluSkill on three coding agents. Higher values indicate that ColluSkill attack chains are more likely to execute successfully at runtime.

and scanner-feedback refinement for scanner evasion. SCR-Bench achieves an average ASR of 34.8%, compared with 96.0% for ColluSkill. This gap shows that multi-skill composition alone is not sufficient for reliable scanner evasion, and that LLM-based chain planning and iterative refinement are important to ColluSkill’s performance.

ChainGuard greatly reduces the attack success rate of ColluSkill. The average ASR of ColluSkill against existing skill scanners is 96.0%, while the ASR drops to 22.5% under ChainGuard. This shows that candidate-with-context scanning can detect some composition risks that isolated scanning misses. Still, ChainGuard does not reduce the ASR to zero, which means that multi-skill attacks remain a challenging defense problem in practice.

Effect of LLM refinement iterations.

Figure 3 shows how LLM refinement affects the average attack success rate. As the number of iterations increases, the average ASR keeps rising. Before refinement, the ASR is only 64.9%. After the first refinement iteration, it quickly increases to 84.7%. This shows that the first iteration can remove many obvious suspicious signals and make the attack samples easier to bypass scanners. The ASR then continues to increase, reaching 90.3% at iteration 3 and 94.4% at iteration 5. After iteration 5, the curve becomes much flatter. The ASR only increases to 94.5% at iteration 6 and 96.0% at iteration 7. This result shows that LLM-based refinement is highly effective in improving attack stealthiness, but most of the gain comes from the first few iterations. In practice, a small number of refinement iterations is enough to greatly improve the attack success rate, while too many iterations bring only limited additional benefit overall.

Effect of the number of sub-skills.

We further study how the number of sub-skills affects ColluSkill. Figure 3 compares chains of different lengths under the same scanner set and refinement budget. All three settings improve after scanner-feedback refinement. This shows that iterative rewriting can reduce local suspicious signals and make the attack chains more likely to pass the scanners.

However, using more sub-skills does not always lead to a higher ASR. The 2-step setting improves quickly in the first few iterations, but it soon becomes stable and reaches a final average ASR of 93.6%. The 3-step setting improves more steadily and achieves the highest final average ASR of 96.0%. In contrast, the 4-step setting only reaches 90.7% after seven refinement iterations.

Scanner	Naive-Split	Chain-Planning	Full ColluSkill
CISCO	100.0	100.0	100.0
SkillFortify	11.5	100.0	100.0
Auditor	19.5	68.5	91.5
SlowMist	10.0	26.5	93.5
Vetter	19.0	29.5	92.0
SkillSpector	60.0	84.5	99.0
Avg	36.7	68.2	96.0

Table 3: Ablation study on ColluSkill components. Higher ASR indicates stronger performance, and the full setting combines chain planning with scanner-feedback refinement.

This result shows a trade-off in chain length. When the chain is too short, each sub-skill needs to carry a larger part of the original payload, which can leave stronger local suspicious signals. When the chain is too long, the payload is split into more parts, but more sub-skills must all pass the scanner. A longer chain is also harder to keep coherent as a natural workflow. In our experiments, the 3-step setting gives the best balance between semantic dispersion and workflow coherence. Therefore, we use 3-step chains as the default setting in the main experiments.

Runtime Chain Activation Study

To verify whether ColluSkill attack chains can work during agent execution, we further evaluate them on OpenCode, Claude Code, and Codex with GPT-5.5, DeepSeek-V4-Pro (Xu et al. 2026), and GLM-5.2 (Z.ai 2026) as model backbones. We use the Chain Activation Success Rate to measure whether a multi-skill chain successfully forms the intended chain-level behavior at runtime.

As shown in Table 2, ColluSkill achieves the highest and most stable activation rates on OpenCode, ranging from 87.5% to 92.5%. The rates range from 62.0% to 78.2% on Claude Code and from 58.5% to 72.0% on Codex. Overall, these results show that ColluSkill attack chains can work across different coding agents and model backbones. They also show that cross-skill composition is not limited to scanner evasion and can appear during real agent execution.

Ablation study on ColluSkill components.

We conduct an ablation study to examine in greater detail the contribution of each component in ColluSkill. As shown in Table 3, we compare Naive Split, Chain Planning, and Full ColluSkill. Naive Split decomposes each payload into independent sub-skills without chain planning or scanner-feedback refinement. Chain Planning generates an ordered sub-skill chain using interdependence-chain planning, while Full ColluSkill further iteratively rewrites the flagged sub-skills using scanner feedback.

Naive Split achieves an average ASR of 36.7%, showing that simple payload splitting can weaken some scanners but does not provide stable evasion. It fully bypasses CISCO and reaches 60.0% on SkillSpector, but achieves only 11.5% on SkillFortify, 19.5% on Auditor, 10.0% on SlowMist, and 19.0% on Vetter. Thus, payload decomposition alone can

Installed-Skill Context	Benign Workflow Pass Rate (%)	ColluSkill Attack ASR (%)
Not Used	99.7	69.0
Used	99.5	22.5

Table 4: Effect of installed-skill context on ChainGuard. Benign workflow results report the pass rate, while ColluSkill attack results report the attack success rate.

work in some cases but cannot reliably bypass most scanners.

Chain Planning increases the average ASR to 68.2% even without scanner feedback. By assigning local roles to sub-skills and connecting them through artifacts, context, or state, it constructs a more coherent cross-skill workflow than independent splitting. Full ColluSkill further raises the average ASR to 96.0% through scanner-feedback refinement, which reduces local suspicious signals while preserving chain-level behavior. Overall, both ordered chain planning and iterative refinement contribute to ColluSkill’s performance.

ChainGuard Defense Evaluation

To evaluate the role of installed-skill context in ChainGuard, we compare two scanning settings, one without access to installed skills and one with the full installed-skill context. We also examine how these settings affect benign workflows. Without installed-skill context, ChainGuard analyzes only the candidate skill, which is similar to traditional isolated single-skill scanning. As shown in Table 4, the attack success rate is 69.0% without installed-skill context. After the context is added, ChainGuard reduces the ASR to 22.5%. This result shows that installed-skill context helps ChainGuard connect local behaviors spread across different skills and detect chain-level risks created by their composition.

At the same time, installed-skill context has little effect on benign workflows. The pass rate decreases only from 99.7% to 99.5%. This shows that ChainGuard does not reduce the attack success rate by simply blocking multi-skill workflows. Instead, it preserves normal skill composition while identifying potentially harmful chains. Overall, these results show that composition-level analysis with installed-skill context can effectively reduce the attack success rate of ColluSkill while maintaining high usability for benign workflows.

Conclusion

This paper proposes ColluSkill, a collusive multi-skill-chain attack framework that distributes harmful intent across locally plausible sub-skills and uses chain planning and scanner-feedback refinement to recover the full attack at the chain level. Across six representative skill scanners, ColluSkill achieves an average ASR of 96.0% and performs best among the evaluated baselines. It also executes successfully on OpenCode, Claude Code, and Codex with different model backbones. To defend against this threat, we propose ChainGuard, which reduces the ASR to 22.5% while allowing 99.5% of benign workflows to pass. These results highlight adversarial cross-skill composition as an important attack surface and motivate chain-level defenses.

References

- Beurer-Kellner, L.; Kudrinskii, A.; Milanta, M.; Nielsen, K. B.; Sarkar, H.; and Tal, L. 2026. Technical Report: Exploring the Emerging Threats of the Agent Skill Ecosystem. *arXiv preprint arXiv:2605.28588*.
- Bhardwaj, V. P. 2026. Formal analysis and supply chain security for agentic AI skills. *arXiv preprint arXiv:2603.00195*.
- Cisco AI Defense. 2026. Skill Scanner: Security Scanner for Agent Skills. <https://github.com/cisco-ai-defense/skill-scanner>. Accessed: 2026-06-25, V2.0.1.
- Du, S.; Zhao, J.; Shi, J.; Xie, Z.; Jiang, X.; Bai, Y.; and He, L. 2026. A survey on the optimization of large language model-based agents. *ACM Computing Surveys*, 58(9): 1–37.
- Elias, N. 1978. *What is sociology?* Columbia University Press.
- fedrov. 2025. Skill Vetter 1.0.0 — ClawHub. <https://clawhub.ai/fedrov2025/skill-vetter-1-0-0>. Accessed: 2026-06-25.
- Feng, Y.; Ding, Y.; Tan, Y.; Zheng, B.; Guo, Y.; Li, X.; Zhai, K.; Li, Y.; and Huang, W. 2026. Skilltrojan: Backdoor attacks on skill-based agent systems. *arXiv preprint arXiv:2604.06811*.
- Guo, W.; Zeng, W.; Liu, C.; Jia, X.; Xu, Y.; Tang, L.; Fang, Y.; and Liu, Y. 2026a. MalSkillBench: A Runtime-Verified Benchmark of Malicious Agent Skills. *arXiv preprint arXiv:2606.07131*.
- Guo, Z.; Chen, Z.; Nie, X.; Lin, J.; Zhou, Y.; and Zhang, W. 2026b. SkillProbe: Security auditing for emerging agent skill marketplaces via multi-agent collaboration. *arXiv preprint arXiv:2603.21019*.
- Hao, H.; Min, D.; Zhang, Z.; Zhang, Y.; Xu, M.; Ge, Y.; and Cheng, L. 2026. POISE: Position-Aware Undetectable Skill Injection on LLM Agents. *arXiv preprint arXiv:2606.07943*.
- Holzbauer, F.; Schmidt, D.; Gegenhuber, G.; Schrittwieser, S.; and Ullrich, J. 2026. Context Matters: Repository-Aware Security Analysis of the Agent Skill Ecosystem. *arXiv preprint arXiv:2603.16572*.
- Hou, Y.; and Yang, Z. 2026. Skillsieve: A hierarchical triage framework for detecting malicious ai agent skills. *arXiv preprint arXiv:2604.06550*.
- Huang, X.; Qin, S.; Jia, X.; Duan, R.; Yan, H.; Zeng, Z.; Yang, F.; and Liu, Y. 2026. Obscure but effective: Classical chinese jailbreak prompt optimization via bio-inspired search. In *International Conference on Learning Representations*, volume 2026, 70802–70832.
- Ji, Z.; Xu, C.; Li, Z.; Gao, Y.; Wei, X.; Wang, S.; and Cheung, S.-C. 2026. Cloak and Detonate: Scanner Evasion and Dynamic Detection of Agent Skill Malware. *arXiv preprint arXiv:2607.02357*.
- Jia, X.; Liao, J.; Qin, S.; Gu, J.; Ren, W.; Cao, X.; Liu, Y.; and Torr, P. 2026a. Skillject: Automating stealthy skill-based prompt injection for coding agents with trace-driven closed-loop refinement. In *The 6th Workshop of Adversarial Machine Learning on Computer Vision: Safety of Vision-Language Agents*.
- Jia, X.; Liao, J.; Qin, S.; Ma, K.; Guo, W.; Feng, Y.; Liu, A.; and Liu, Y. 2026b. Seeing Is Not Screening: Multimodal Hidden Instruction Attacks on Agent Skill Scanners. *arXiv preprint arXiv:2606.18198*.
- Jia, X.; Pang, T.; Du, C.; Huang, Y.; Gu, J.; Liu, Y.; Cao, X.; and Lin, M. 2024. Improved techniques for optimization-based jailbreaking on large language models. *arXiv preprint arXiv:2405.21018*.
- Jiang, Y.; Li, D.; Deng, H.; Ma, B.; Wang, X.; Wang, Q.; and Yu, G. 2026. SoK: Agentic Skills—Beyond Tool Use in LLM Agents. *arXiv preprint arXiv:2602.20867*.
- Jin, C.; Wang, A.; Wei, Z.; Wang, K.; Zeng, B.; Zhang, Q.; Yang, C.; Qu, J.; Hu, X.; and Xu, X. 2026. SkillSafetyBench: Evaluating agent safety under skill-facing attack surfaces. *arXiv preprint arXiv:2605.12015*.
- Kim, Y.; Song, M.; and Shin, S. 2026. SkillMutator: Benchmarking and Defending Language-and-Code Cross-modal Attacks on LLM Agent Skills. *arXiv preprint arXiv:2606.14154*.
- Li, H.; Mu, C.; Chen, J.; Ren, S.; Cui, Z.; Zhang, Y.; Bai, L.; and Hu, S. 2026a. Organizing, Orchestrating, and Benchmarking Agent Skills at Ecosystem Scale. *arXiv:2603.02176*.
- Li, X. 2026. When single-agent with skills replace multi-agent systems and when they fail. *arXiv preprint arXiv:2601.04748*.
- Li, X.; Chen, W.; Liu, Y.; Zheng, S.; Chen, X.; He, Y.; Li, Y.; You, B.; Shen, H.; Sun, J.; et al. 2026b. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*.
- Li, Z.; Wu, J.; Ling, X.; Cui, X.; and Luo, T. 2026c. Towards secure agent skills: Architecture, threat taxonomy, and security analysis. *arXiv preprint arXiv:2604.02837*.
- Liang, Q.; Wang, H.; Liang, Z.; and Liu, Y. 2026. From skill text to skill structure: The scheduling-structural-logical representation for agent skills. *arXiv preprint arXiv:2604.24026*.
- Ling, G.; Zhong, S.; and Huang, R. 2026. Agent skills: A data-driven analysis of claude skills for extending large language model functionality. *arXiv preprint arXiv:2602.08004*.
- Liu, B.; Li, X.; Zhang, J.; Wang, J.; He, T.; Hong, S.; Liu, H.; Zhang, S.; Song, K.; Zhu, K.; et al. 2025. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*.
- Liu, Y.; Chen, Z.; Zhang, Y.; Deng, G.; Li, Y.; Ning, J.; Zhang, Y.; and Zhang, L. Y. 2026a. Malicious agent skills in the wild: A large-scale security empirical study. *arXiv preprint arXiv:2602.06547*.
- Liu, Y.; Wang, W.; Feng, R.; Zhang, Y.; Xu, G.; Deng, G.; Li, Y.; and Zhang, L. 2026b. Agent Skills in the Wild: An Empirical Study of Security Vulnerabilities at Scale. *arXiv preprint arXiv:2601.10338*.
- NVIDIA. 2026. SkillSpector: Security Scanner for AI Agent Skills. <https://github.com/NVIDIA/SkillSpector>. Accessed: 2026-06-25.
- OpenAI. 2026. GPT-5.5 System Card. <https://openai.com/index/gpt-5-5-system-card/>. Accessed: 2026-07-11.

- Pu, H.; Song, X.; and Zhao, L. 2026. SkillOps: Managing LLM Agent Skill Libraries as Self-Maintaining Software Ecosystems. *arXiv preprint arXiv:2605.13716*.
- Qu, Y.; Liu, Y.; Geng, T.; Deng, G.; Li, Y.; Zhang, L. Y.; Zhang, Y.; and Ma, L. 2026. Supply-chain poisoning attacks against LLM coding agent skill ecosystems. *arXiv preprint arXiv:2604.03081*.
- Quintero, B. 2026. From Automation to Infection: How OpenClaw AI Agent Skills Are Being Weaponized. Virus-Total Blog. Accessed: 2026-06-25.
- Ruan, Y.; Dong, H.; Wang, A.; Pitis, S.; Zhou, Y.; Ba, J.; Dubois, Y.; Maddison, C.; and Hashimoto, T. 2024. Identifying the risks of lm agents with an lm-emulated sandbox. In *International Conference on Learning Representations*, volume 2024, 27031–27098.
- Saha, S.; and Hemanth, P. 2026. Skilldex: A Package Manager and Registry for Agent Skill Packages with Hierarchical Scope-Based Distribution. *arXiv preprint arXiv:2604.16911*.
- Schmotz, D.; Abdelnabi, S.; and Andriushchenko, M. 2025. Agent Skills Enable a New Class of Realistic and Trivially Simple Prompt Injections. *arXiv preprint arXiv:2510.26328*.
- Schmotz, D.; Beurer-Kellner, L.; Abdelnabi, S.; and Andriushchenko, M. 2026. Skill-inject: Measuring agent vulnerability to skill file attacks. *arXiv preprint arXiv:2602.20156*.
- SlowMist. 2026. SlowMist Agent Security Skill. <https://github.com/slowmist/slowmist-agent-security>. GitHub repository, accessed July 11, 2026.
- Snyk. 2025. Agent Scan: Security Scanner for AI Agents, MCP Servers and Agent Skills. <https://github.com/snyk/agent-scan>. Accessed: 2026-06-25.
- Tie, G.; Shi, J.; Zhou, P.; and Sun, L. 2026. Badskill: Backdoor attacks on agent skills via model-in-skill poisoning. *arXiv preprint arXiv:2604.09378*.
- Wang, C.; Yu, Z.; Xie, X.; Yao, W.; Fang, R.; Qiao, S.; Cao, K.; Zheng, G.; Qi, X.; Zhang, P.; et al. 2026. Skillx: Automatically constructing skill knowledge bases for agents. *arXiv preprint arXiv:2604.04804*.
- Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.
- Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345.
- Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2): 121101.
- Xie, Y.; Du, J.; Cheng, Y.; Zhou, J.; and Yin, Z. 2026. Benign in Isolation, Harmful in Composition: Security Risks in Agent Skill Ecosystems. *arXiv preprint arXiv:2606.15242*.
- Xu, A.; Lin, B.; Xue, B.; Wang, B.; Xu, B.; Wu, B.; Zhang, B.; Lin, C.; Dong, C.; Ling, C.; et al. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*.
- Xu, R.; and Yan, Y. 2026. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*.
- Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
- Ying, Z.; Yang, X.; Wu, S.; Song, Y.; Qu, Y.; Li, H.; Li, T.; Wang, J.; Liu, A.; and Liu, X. 2026. Uncovering security threats and architecting defenses in autonomous agents: A case study of openclaw. *arXiv preprint arXiv:2603.12644*.
- Z.ai. 2026. GLM-5.2: Built for Long-Horizon Tasks. <https://z.ai/blog/glm-5.2>. Accessed: 2026-07-17.
- Zheng, B.; Fatemi, M. Y.; Jin, X.; Wang, Z. Z.; Gandhi, A.; Song, Y.; Gu, Y.; Srinivasa, J.; Liu, G.; Neubig, G.; et al. 2025. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*.
- Zhou, Y.; Shu, W.; Su, Y.; Du, W.; Fang, Y.; and Lin, X. 2026. A comprehensive survey on agent skills: Taxonomy, techniques, and applications. *arXiv preprint arXiv:2605.07358*.