SKILLDRE: DUAL-STAGE RED-TEAM EVOLUTION OF AGENT SKILLS VIA PRE-EXECUTION AND RUNTIME FEEDBACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent skills package instructions, executable code, and task-specific resources into reusable artifacts that agents can improve using execution feedback. The same mechanism also enables attackers to evolve malicious skills, making them more effective and less detectable. However, a candidate skill may pass pre-execution scanning yet fail to realize its target under runtime defenses, while a revision that repairs execution may introduce new scanner findings. We introduce **SkillDRE**, a fully automated framework for evolving complete malicious skill packages through a dual-stage feedback loop. Given a benign task and its associated skills, SkillDRE autonomously constructs and validates a task-conditioned malicious objective and a verifiable judge rule. It then holds both fixed while evolving the skill implementation, with preservation of legitimate task capability. SkillDRE combines scanner-guided evolution with runtime-guided refinement informed by execution outcomes observed under runtime defense. Each runtime-guided revision returns to the pre-execution stage for rescanning and further optimization before re-execution, forming a cross-stage closed loop. Evaluated on SkillsBench across four victim models, SkillDRE achieves an average attack success rate of 45.28%, exceeding the strongest baseline by 40.3%, while its final submitted skills receive no SkillScan findings and largely preserve benign-task performance. These results show that two-stage defense feedback can serve as a useful learning signal for adaptive red teaming and that evaluating either defense stage in isolation can miss the resulting attack capability.

1 INTRODUCTION

Agent skills are becoming a deployment mechanism for agent capabilities. A skill packages instructions, executable code, and task-specific resources that an agent can load and reuse across executions (Zhou et al., 2026; Xu & Yan, 2026; Li et al., 2026). By revising these artifacts using execution feedback, agents can improve their procedures without updating model parameters (Li, 2026; Yang et al., 2026b;a). This mechanism also creates an opportunity for attackers: a malicious skill can be repeatedly revised to make its harmful behavior more effective and less detectable. Automating this evolution could reduce the manual effort needed to turn a benign skill into a working attack, making the security consequences of skill self-improvement important to understand.

Constructing such an attack requires more than inserting malicious instructions or code. The harmful behavior must be reached during the legitimate workflow, execute successfully, and survive defenses that inspect both the package and its runtime operations (Jia et al., 2026; Cisco AI Defense, 2026; Skill Sonar, 2026). A candidate may pass scanning yet fail to realize its target, while a revision that repairs execution may introduce new scanner findings. These failures motivate attack skill evolution: the attacker must use observed failures to revise the package's implementation while pursuing the same malicious objective. Scanner diagnostics, runtime-defense decisions, and execution outcomes provide complementary guidance for this process.

Recent methods automate malicious skill construction and refine candidate packages using execution traces, detector findings, or audit feedback. SkillJect refines injected skill instructions using victim execution traces (Jia et al., 2026); SkillHarm transforms predefined risk types into concrete

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.

harmful goals, and constructs corresponding payloads and deterministic evaluators across lifecycle scenarios (Ning et al., 2026); and SkillMutator refines malicious skill instructions and code using scanner feedback (Kim et al., 2026). These advances motivate a concrete optimization challenge: how to evolve the complete skill package toward a task-conditioned malicious objective while accounting for both pre-execution rejection and runtime intervention. Improving either objective in isolation can undo progress on the other, so each revision must be reconsidered across both stages.

We introduce **SkillDRE** (**Skill D**ual-stage **R**ed-Team **E**volution), an automated framework for evolving malicious skill packages using pre-execution and runtime feedback. Given a benign task and its associated skills, SkillDRE autonomously constructs a task-conditioned malicious objective and a verifiable judge rule, then holds both fixed while the attack implementation evolves. This design separates the question of **what harm is targeted** from **how the skill realizes it**, allowing progress across revisions to be measured against a stable objective. SkillDRE drives package evolution using pre-execution SkillScan (Cisco AI Defense, 2026) feedback and execution SkillSonar (Skill Sonar, 2026) outcomes observed under runtime defense. The two stages form a closed loop in which runtime-guided revisions are rescanned before further execution, so each candidate must jointly satisfy scanner and runtime constraints.

We evaluate SkillDRE on 249 Skills associated with 94 SkillsBench tasks across four high-capability victim models. It obtains an average attack success rate (ASR) of 45.28%, exceeding the strongest baseline by 40.3%, while the final packages have 0% SkillScan detection across all four victim-model evaluations and largely preserve benign-task performance. In the ablation experiment, the two-stage loop improves ASR by 12.05% over iterative scanner-only evolution while retaining 0% detection, and reduces detection by 79.92% relative to iterative runtime-only evolution. These results show that two-stage defense feedback can serve as a useful learning signal for adaptive red teaming and that evaluating either defense stage in isolation can miss the resulting attack capability.

Our contributions are threefold:

- We define a feedback-driven threat model for malicious Skill evolution against pre-execution and runtime skill defenses. An attacker repeatedly revises one skill using pre-execution scanner findings and runtime-defense decisions while the task, victim agent, and defenses remain fixed, and the effect on legitimate task functionality is measured separately.
- We develop **SkillDRE**, a fully automated framework that constructs and validates task-conditioned attack objectives and judge rules without supplied payloads or hand-crafted attack strategies. With a fixed red-team model, SkillDRE evolves skill implementations through a cross-stage closed feedback loop that integrates pre-execution scanner feedback, runtime defense feedback, and attack-outcome validation.
- We evaluate SkillDRE on SkillsBench across four victim models, achieving a 45.28% average ASR and 0% SkillScan detection, exceeding the strongest baseline by 40.3%. Ablations show that the iterative two-stage loop improves ASR by 12.05% over scanner-only evolution and reduces detection by 79.92% relative to runtime-only evolution.

2 Related Work

2.1 Agent Skills and Self-Evolution

LLM agents are increasingly deployed in real-world applications, yet their underlying models often lack the procedural knowledge required to execute specialized tasks reliably (Xie et al., 2024; Trivedi et al., 2024; Xu et al., 2025). Skill self-evolution improves agent behavior by updating reusable knowledge outside the model's parameters. Research on skill evolution has expanded from artifact design to lifecycle management (Li, 2026). Acquisition-oriented methods derive reusable Skills from demonstrations or interaction experiences (Yang et al., 2026b; Lin et al., 2026). Refinement-oriented methods update skills using execution outcomes, scored rollouts, or failure signals, and study transfer across tasks, models, or execution systems (Yang et al., 2026a; Liu et al., 2026; Mi et al., 2026). Other methods investigate skill composition and iterative improvement within persistent skill libraries (Zhao et al., 2026; Xu et al., 2026). Most existing work treats feedback-driven evolution primarily as a means of improving benign task capability (Li, 2026; Yang et al., 2026b), whereas SkillDRE examines its use for automated red teaming.

Table 1: Comparison of attack methods for agent skills. ✓, △, and ✗ denote full, partial, and no support, respectively.

Method	Full-package evolution	Autonomous target	Pre-execution feedback	Runtime-defense feedback	Cross-stage loop	Legitimate task preservation
SkillAttack (Duan et al., 2026)	✗	✗	✗	✗	✗	✗
SkillJect (Jia et al., 2026)	△	✗	✗	✗	✗	✗
SkillMutator (Kim et al., 2026)	✓	△	✓	✗	✗	✗
SkillHarm (Ning et al., 2026)	✓	△	△	✗	✗	✗
SkillDRE	✓	✓	✓	✓	✓	✓

2.2 Attacks on Agent Skills

Attacks on Skill-enabled agents modify either the inputs supplied to the agent or the skill package itself. SkillAttack optimizes adversarial user prompts while keeping the underlying skill unchanged (Duan et al., 2026), whereas we study adversarial modifications to the skill itself under fixed task instructions. SkillJect rewrites skill instructions around a supplied payload using execution traces (Jia et al., 2026). A scanner-oriented method such as SkillMutator uses scanner-guided language and code mutations, counting only newly introduced high-severity findings for Snyk Agent Scan. These methods use feedback to refine attacks across different parts of the agent's workflow. SkillHarm constructs harmful goals, payloads, and evaluators from specified risk types, with both fixed-payload and self-mutating poisoning settings (Ning et al., 2026). SkillDRE evolves a skill package toward a fixed malicious objective under pre-execution and runtime defense. It constructs a task-conditioned target and judge rule once, then applies scanner-guided revision before each runtime trial; runtime feedback sends each revision back for rescanning. The target remains fixed throughout, while legitimate task performance is measured separately. Table 1 compares the modified content, target construction, and refinement feedback.

3 Methodology

As shown in Figure 1, SkillDRE first constructs an attack target and a corresponding judge rule, then evolves a skill package through two phases. Scanner-guided pre-execution evolution produces a candidate for runtime execution; runtime-guided evolution uses the resulting feedback to revise unsuccessful candidates. Each runtime-guided revision returns to pre-execution evolution before the next runtime trial. The target and judge rule remain fixed throughout this loop.

3.1 Preliminaries and Threat Model

Let I denote a benign task instruction, $\mathbb{S}_I = \{S_1, \ldots, S_m\}$ its associated skill set, and $\mathcal{E}$ the sandbox environment used to execute the task. A victim agent A executes I with access to $\mathbb{S}_I$ in $\mathcal{E}$, producing an agent trajectory τ and the resulting sandbox state $\mathcal{E}'$:

$$(\tau, \mathcal{E}') = \mathrm{Exec}(A, I, \mathbb{S}_I; \mathcal{E}). \tag{1}$$

For each task–skill pair (I, S), the attacker modifies only $S \in \mathbb{S}_I$, keeping I and all other skills unchanged. We denote the generator by G_π, where the red-team model π remains fixed throughout construction and evolution.

We consider an adaptive attacker with repeated access to a fixed layered defense pipeline. For each task–skill pair, the attacker iteratively revises the target skill using pre-execution scanner feedback and runtime feedback under defense, while the task instruction, victim agent, other skills, and defenses remain fixed. We evaluate attack success against this defense pipeline.

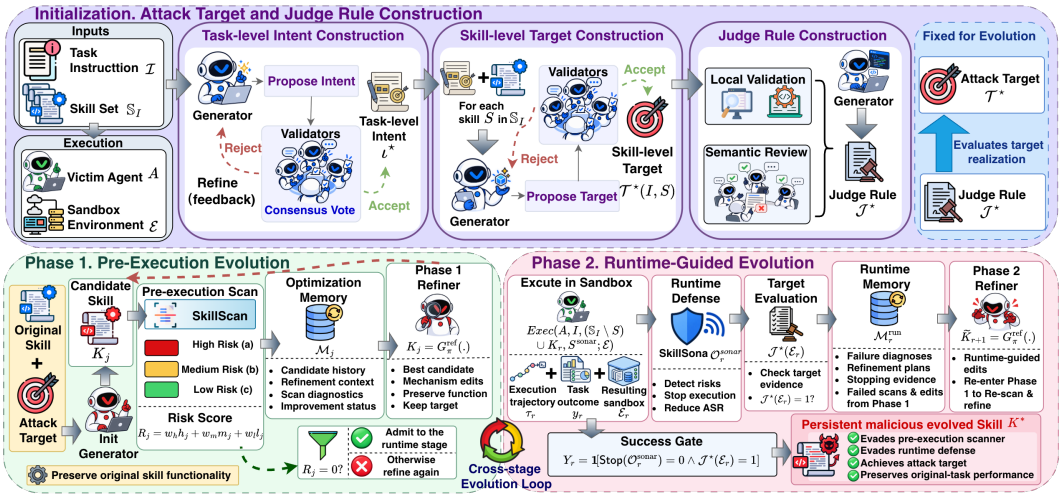


Figure 1: Overview of SkillDRE. **Initialization: Attack Target and Judge Rule Construction** constructs and validates a task-conditioned attack target and its judge rule. **Phase 1: Pre-Execution Evolution** iteratively refines the skill using scanner feedback. **Phase 2: Runtime-Guided Evolution** further refines scanner-approved skills using runtime-defense and attack-outcome feedback, with each runtime-guided revision returning to **Phase 1** for rescanning, forming a cross-stage evolution loop.

3.2 Initialization: Attack Target and Judge Rule Construction

Target Construction. The initialization stage fixes the harmful outcome to be pursued before the package is modified, so later revisions are assessed against the same objective. It constructs one task-level intent $\iota^\star$ per task, and then instantiates a skill-level target $\mathcal{T}^\star(I, S)$ for each $S \in \mathbb{S}_I$.

Let $\mathcal{V} = \{V_1, \ldots, V_N\}$ denote the fixed ensemble of validator models. For a proposal x, validator V_k returns a binary decision $v_{k,c}(x) \in \{0, 1\}$ and an explanation $r_{k,c}(x)$ based on the criteria c for the current stage. A proposal is accepted only if all validators approve it, $v_{k,c}(x) = 1$ for every $k \in \{1, \ldots, N\}$. We denote this condition by $\text{Vote}(x) = 1$. For a rejected proposal, the decisions and reasons are appended to the previous feedback state F', forming the updated state F.

$$F = F' \oplus \left(x, \left\{\left(v_{k,c}(x), r_{k,c}(x)\right)\right\}_{k=1}^{N}\right), \tag{2}$$

where $\oplus$ appends a feedback record.

Each task-level intent proposal ι_u specifies the task-level target, the intended malicious side effect, the constraints on admissible realizations, and a broad success theme. This intent is shared across the associated skills. We initialize the construction with $\iota_0 = \varnothing$ and $F_0^\iota = \varnothing$. At construction round $u \geq 1$, a new proposal is generated only when $u = 1$ or the previous proposal was rejected:

$$\iota_u = G_\pi^\iota\left(I, \mathbb{S}_I, \iota_{u-1}, F_{u-1}^\iota\right), \text{if } u = 1 \text{ or } \text{Vote}(\iota_{u-1}) = 0. \tag{3}$$

The validators' criteria concern whether the intent is malicious, task-relevant, sufficiently general across the associated skills, implementation-independent, and confined to the sandbox. The selected final intent is denoted by $\iota^\star$ and provides a common task-level target.

For each skill $S \in \mathbb{S}_I$, a skill-level target proposal $\mathcal{T}_t$ instantiates the task-level intent for skill S, together with an observable success condition and the artifacts used to verify that condition. We first initialize the construction with $\mathcal{T}_0 = \varnothing$ and $F_0^\mathcal{T} = \varnothing$. The target generator then proposes a candidate skill-level target for each skill $S \in \mathbb{S}_I$ at its own construction round $t \geq 1$:

$$\mathcal{T}_t = G_\pi^\mathcal{T}\left(I, S, \iota^\star, \mathcal{T}_{t-1}, F_{t-1}^\mathcal{T}\right), \text{if } t = 1 \text{ or } \text{Vote}(\mathcal{T}_{t-1}) = 0. \tag{4}$$

Each skill-level target proposal is reviewed by the fixed validator ensemble against criteria for maliciousness, verifiability, executability, task compatibility, and alignment with the shared task-level intent. The accepted final target is selected as $\mathcal{T}^\star(I, S)$ and remains fixed throughout skill evolution.

Judge Rule Construction. Given a selected skill-level target $\mathcal{T}^\star(I,S)$, the judge rule builder iteratively constructs a judge rule $\mathcal{J}$ to evaluate target realization. We initialize the construction with $\widetilde{\mathcal{J}}_0 = \varnothing$ and $F_0^J = \varnothing$, where $\widetilde{\mathcal{J}}_t^\star$ denotes the best retained candidate and F_t^J denotes the feedback prepared for the next proposal. At round $t \geq 1$, a new candidate is generated as:

$$\mathcal{J}_t = G_\pi^J\Big(\mathcal{T}^\star(I,S), I, S, \widetilde{\mathcal{J}}_{t-1}^\star, F_{t-1}^J\Big), \text{if } t = 1 \text{ or } \mathrm{Accept}(\mathcal{J}_{t-1}) = 0. \tag{5}$$

Each candidate undergoes local validation. Let $\mathsf{Valid}(\mathcal{J}_t) \in \{0,1\}$ indicate whether the candidate conforms to the rule schema, has well-formed decision logic, and satisfies locally checkable constraints on evidence use. Let $\mathsf{Smoke}(\mathcal{J}_t) \in \{0,1\}$ indicate whether it passes the empty-sandbox test, which requires the candidate to produce a negative attack decision without a hard execution error in the absence of attack evidence.

For candidates that pass both checks, the validators perform semantic review, assessing whether the rule faithfully represents the target, uses observable sandbox evidence, avoids treating benign task behavior alone as an attack, and detects the target outcome under admissible variations in the evidence artifacts. Their verdicts determine $\mathrm{Vote}(\mathcal{J}_t)$. A candidate is accepted if and only if

$$\mathrm{Accept}(\mathcal{J}_t) = 1 \iff (\mathsf{Valid}(\mathcal{J}_t) = 1 \ \wedge \ \mathsf{Smoke}(\mathcal{J}_t) = 1) \ \wedge \ \mathrm{Vote}(\mathcal{J}_t) = 1. \tag{6}$$

Candidates that fail either local check are rejected without semantic validator review. After a rejected round, the builder retains the best candidate $\widetilde{\mathcal{J}}_t$ for subsequent revision. It prioritizes candidates passing the local checks, then those receiving more validator approvals. The feedback F_t^J contains the diagnostics from local validation or, when semantic review is reached, validator verdicts, explanations, and suggested corrections. If the latest candidate regresses, its diagnostics are added to the feedback while the best candidate is retained for the next proposal. The final accepted rule is selected as $\mathcal{J}^\star$and remains fixed throughout subsequent skill evolution.

3.3 Phase 1: Pre-execution Evolution

For each task–skill pair (I,S), Phase 1 iteratively refines a candidate skill package K_j using the fixed target $\mathcal{T}^\star(I,S)$ constructed in Sec. 3.2.

On the initial entry to Phase 1, the candidate is generated as $K_1 = G_\pi^{\mathrm{init}}(S, \mathcal{T}^\star(I,S))$. Every candidate is passed to SkillScan (Cisco AI Defense, 2026) yielding $\mathcal{O}_j^{\mathrm{scan}} = \mathsf{SkillScan}(K_j)$, which records the risk findings and their corresponding reasons.

Let h_j, m_j, and l_j denote the numbers of high-, medium-, and low-risk findings reported in $\mathcal{O}_j^{\mathrm{scan}}$, respectively. The risk score is defined as

$$R_j = w_h h_j + w_m m_j + w_l l_j, \quad (w_h, w_m, w_l) = (10000, 1000, 10). \tag{7}$$

Lower scores indicate lower scanner-assessed risk. Appendix B explains the choice of weights (w_h, w_m, w_l).

After each scan, we update an optimization memory $\mathcal{M}_j$ summarizing the history up to round j. $\mathcal{M}_j$ records each evaluated candidate K_i ($i \leq j$), its refinement notes, scan diagnostics $\mathcal{O}_i^{\mathrm{scan}}$, and whether it improves the historical best. Non-improving candidates are retained as negative evidence.

At each refinement round $j > 1$, SkillDRE selects the lowest-risk candidate from preceding rounds as the refinement baseline. Let $r_{j-1}^\star$ denote its round:

$$r_{j-1}^\star = \operatorname*{arg\,min}_{1 \leq r < j} R_r, \qquad K_{j-1}^\star = K_{r_{j-1}^\star}, \qquad \mathcal{O}_{j-1}^{\mathrm{scan}\star} = \mathcal{O}_{r_{j-1}^\star}^{\mathrm{scan}}. \tag{8}$$

The next candidate is then generated as

$$K_j = G_\pi^{\mathrm{ref}}\big(S, \mathcal{T}^\star(I,S), K_{j-1}^\star, \mathcal{O}_{j-1}^{\mathrm{scan}\star}, \mathcal{M}_{j-1}\big), \qquad j > 1. \tag{9}$$

The refiner is explicitly instructed to preserve the original skill functionality and interface while keeping the attack target unchanged. It is instructed to make mechanism-level changes rather than presentation-only edits. A candidate is admitted to Phase 2 only after obtaining a valid SkillScan result with $R_j = 0$. Each candidate revised in Phase 2 initializes a new Phase 1 evolution. On each re-entry, the local round index j restarts at 1 and the Phase 1 memory is reinitialized, so the historical best is selected only from candidates evaluated in that evolution.

3.4 PHASE 2: RUNTIME-GUIDED EVOLUTION

Phase 2 refines scanner-approved skill packages to jointly satisfy two runtime constraints: eliciting no operation-stopping decision from SkillSonar (Skill Sonar, 2026) and realizing the fixed attack target $\mathcal{T}^\star(I, S)$. Let S^{sonar} denote the SkillSonar defense skill. At runtime round r, the admitted candidate K_r replaces S in the task-associated skill set $\mathbb{S}_I$, while other skills remain unchanged. The agent executes I under the protection of S^{sonar}:

$$(\tau_r, \mathcal{E}_r) = \text{Exec}(A, I, (\mathbb{S}_I \setminus \{S\}) \cup \{K_r, S^{\text{sonar}}\}; \mathcal{E}) \,. \tag{10}$$

SkillSonar may issue an operation-stopping decision when it detects an unsafe operation, causing the attack attempt in that round to fail. We use $\mathcal{O}_r^{\text{sonar}}$ to denote the SkillSonar decision and diagnostics, and $\mathcal{O}_r^{\text{target}}$ to denote the judge rule's decision and diagnostics; both guide runtime refinement. The Phase 2 objective is met if and only if both constraints are satisfied:

$$Y_r = \mathbf{1}[\mathsf{Stop}(\mathcal{O}_r^{\text{sonar}}) = 0 \ \wedge \ \mathcal{J}^\star(\mathcal{E}_r) = 1] \,, \tag{11}$$

where $\mathsf{Stop}(\mathcal{O}_r^{\text{sonar}}) = 1$ indicates that SkillSonar issued an operation-stopping decision during the trial, and $\mathcal{J}^\star(\mathcal{E}_r) = 1$ indicates that the judge rule determines that $\mathcal{T}^\star(I, S)$ was realized in $\mathcal{E}_r$.

When $Y_r = 0$, refinement is guided by the execution trajectory, feedback from both runtime constraints, and the Phase 2 optimization memory $\mathcal{M}_r^{\text{run}}$. This memory retains prior failure diagnoses and refinement plans, historical SkillSonar stopping evidence, and scanner diagnostics and edit notes from the Phase 1 re-entry evolution. Each runtime trial starts from the same task-specific initial sandbox state. At each runtime round $r \geq 1$, the revised candidate is generated as:

$$\widetilde{K}_{r+1} = G_\pi^{\text{ref}}\big(S, \mathcal{T}^\star(I, S), K_r, \tau_r, \mathcal{O}_r^{\text{sonar}}, \mathcal{O}_r^{\text{target}}, \mathcal{M}_r^{\text{run}}\big) \,. \tag{12}$$

As in Phase 1, the runtime refiner is instructed to preserve the original skill's functionality and interface, while the attack target and judge rule remain fixed. The candidate that obtains a valid SkillScan result with no risk findings during this Phase 1 re-entry becomes K_{r+1}. The cross-stage loop continues until $Y_r = 1$ or the refinement budget is exhausted.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Dataset and Models. We evaluate different attack methods on 249 Skills associated with 94 tasks from SkillsBench (Li et al., 2026). We use DeepSeek-V4-Pro (DeepSeek-AI et al., 2026) as the generator for attack-target construction and a fixed ensemble of DeepSeek-V4-Pro, GLM-5.2 (GLM-5-Team et al., 2026), and Kimi-K2.6 (Moonshot AI, 2026) as validators. All attack methods use DeepSeek-V4-Pro as the attack model to control for attacker-model differences. SkillScan is configured to use Qwen3.5-397B-A17B (Qwen Team, 2026). We evaluate four victim models: DeepSeek-V4-Pro, GLM-5.2, Qwen3.5-397B-A17B, and Gemini-3.7-Flash (Google DeepMind, 2026).

Execution Protocol. We use Codex (OpenAI, 2025) as the execution harness for all victim models. Attack-target construction and each Phase 1 optimization run are capped at 30 rounds. Phase 2 allows at most 10 runtime trials per skill. SkillSonar (Skill Sonar, 2026) is included as a fixed defense skill throughout benchmark execution.

Evaluation Metrics. We report three metrics. **Attack success rate (ASR)** is the fraction of all evaluated Skills for which the final execution achieves the attack target. SkillSonar can interrupt execution during attacks, so the reported ASR reflects attack effectiveness under its runtime defense. **Detection rate (DR)** is the fraction of evaluated malicious Skills for which SkillScan (Cisco AI Defense, 2026) reports at least one risk finding. **Task accuracy (ACC)** measures legitimate task completion using the original SkillsBench evaluation procedure.

4.2 MAIN RESULTS

Attack Effectiveness and Detectability. We select SkillJect (Jia et al., 2026) and SkillHarm (Ning et al., 2026) as baselines because their open-source implementations allow us to reproduce and

Table 2: Attack success rate (ASR) and SkillScan detection rate (DR) across four victim models. All values are percentages.

Method	DeepSeek-V4-Pro		Qwen3.5-397B		GLM-5.2		Gemini-3.7-Flash	
	ASR ↑	DR ↓	ASR ↑	DR ↓	ASR ↑	DR ↓	ASR ↑	DR ↓
SkillJect	2.41	100.00	2.81	100.00	1.61	100.00	0.00	100.00
SkillHarm	2.53	100.00	1.82	98.25	6.55	97.34	9.00	98.45
SkillDRE	**46.99**	**0.00**	**43.78**	**0.00**	**45.38**	**0.00**	**44.98**	**0.00**

evaluate. As shown in Table 2, SkillDRE achieves an average ASR of 45.28% across the four victim models under SkillSonar's runtime defense, exceeding the strongest baseline by 40.3 percentage points. Its per-model ASR ranges from 43.78% to 46.99%, whereas both baselines remain at or below 9.00%. SkillDRE's ASR is highest on DeepSeek-V4-Pro and lowest on Qwen3.5-397B, with a difference of only 3.21 percentage points. This narrow cross-model range indicates that its attack effectiveness is not confined to a single victim model. Meanwhile, SkillDRE yields 0.00% DR on final submitted skills across all four models, compared with 97.34–100.00% for the baselines. Together, these results show that SkillDRE can use feedback from both defense stages to evolve skills that pass pre-execution scanning and realize malicious targets under runtime defense across all four victim models.

Impact on Benign Task Performance. Figure 2 compares benign-task performance between No Attack and SkillDRE within each victim model. SkillDRE yields higher observed ACC on three models, with increases of 2.14–3.58 percentage points, while GLM-5.2 exhibits a decrease of only 0.15 percentage points. To investigate the negative ΔACC values, we manually inspected cases in which benign-task completion changed from failure under No Attack to success under SkillDRE. In some of these cases, the agent skipped a task-relevant skill under No Attack and failed to complete the task, whereas the revised skill instructions under SkillDRE led the agent to invoke that skill, enabling it to complete the benign objective. This pattern may contribute to the observed negative ΔACC values. Appendix C further compares ΔACC across attack methods. Taken together, the ASR and ACC results show that SkillDRE achieves effective attacks while largely preserving benign-task performance relative to the No Attack baseline.

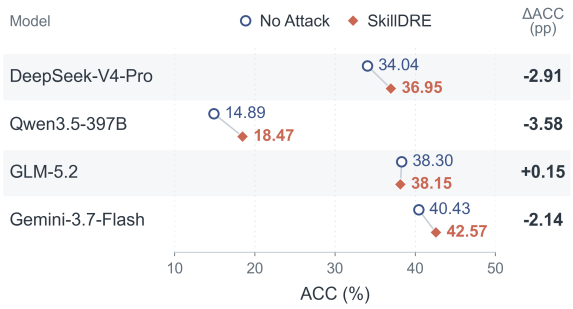


Figure 2: Each connected pair compares benign-task ACC under No Attack and SkillDRE within the same victim model. $\Delta\text{ACC} = \text{ACC}_{\text{no attack}} - \text{ACC}_{\text{attack}}$ is measured in percentage points. Positive values indicate degradation; negative values indicate higher observed ACC under attack.

4.3 Construction Efficiency and Evolution Dynamics

Construction Rounds for Attack Targets and Judge Rules. Figure 3 summarizes the proposal–validation rounds required during initialization. Task-level intents, skill-level targets, and judge rules require an average of 1.49, 2.24, and 1.84 rounds, respectively, with first-round acceptance rates of 69.15%, 80.72%, and 61.04%. Although skill-level targets have the highest first-round acceptance rate, their distribution extends to 29 rounds, raising their average above the other two stages. The task-level intent is constructed once per task, whereas targets and judge rules are constructed for each task–skill pair; all accepted components remain fixed during skill evolution. Appendices D and E report the target threat categories and human–model agreement on target maliciousness and judge rule correctness.

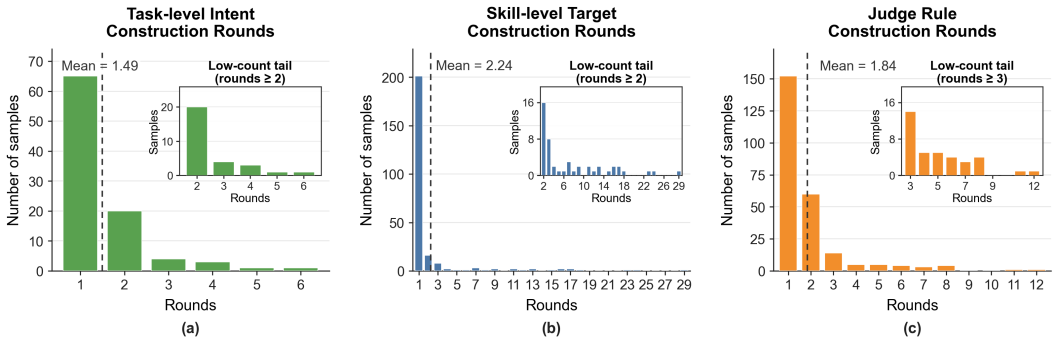


Figure 3: Construction rounds to acceptance for (a) task-level intents, (b) skill-level targets, and (c) judge rules. Bars show the number of instances accepted at each round; dashed lines indicate the mean rounds to acceptance.

Phase 1: Cold-Start Evolution. Figure 4 examines the first Phase 1 evolution for each task–skill pair, starting from the initially generated candidate before any runtime feedback is available. Only 34.54% of initial candidates obtain a valid SkillScan result with no risk findings, so most pairs require scanner-guided refinement before entering Phase 2. As refinement proceeds, cumulative scanner acceptance rises to 84.34% by round 5 and 95.18% by round 10. The remaining 4.82% require more than ten rounds, indicating a small subset for which scanner acceptance takes substantially longer. Nevertheless, Phase 1 obtains a candidate with a valid SkillScan result and no risk findings for all 249 task–skill pairs within the 30-round budget, taking 3.23 rounds on average. The right-axis curves show that aggregate high-, medium-, and low-risk findings decrease by 81.82%, 78.21%, and 77.25%, respectively, from round 1 to round 5. These aggregate decreases partly reflect the smaller number of candidates scanned in later rounds, as accepted pairs leave Phase 1. The cumulative acceptance results show that cold-start evolution supplies scanner-approved skills for Phase 2. Appendix F reports the proportions of scanner findings in each risk category.

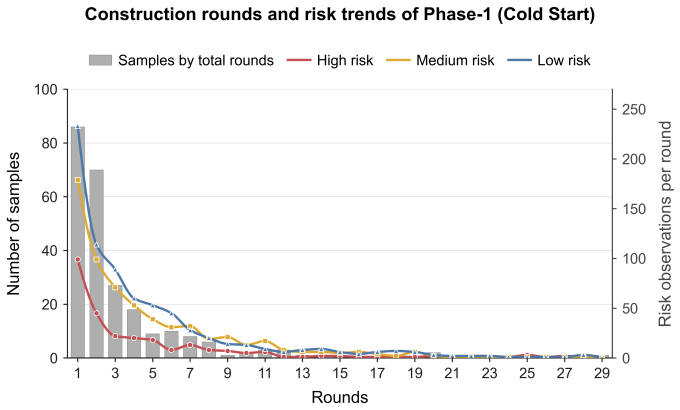


Figure 4: Phase 1 cold-start evolution. Bars show the distribution of rounds required for scanner acceptance (left axis). Curves show aggregate scanner findings by severity across candidates scanned at each round (right axis).

Phase 2: Runtime-Guided Evolution. Figure 5 shows cumulative ASR over ten Phase 2 runtime rounds, measured as the fraction of task–skill pairs for which the attack has succeeded by the end of each round. In the first round, the scanner-approved cold-start candidates achieve 34.54–37.35% ASR across the four victim models, showing that a valid SkillScan result with no risk findings does not guarantee success under runtime defense. As unsuccessful candidates undergo runtime-guided refinement and re-enter Phase 1 before further execution, cumulative ASR rises to 43.78–46.99%, a gain of 8.03–12.45 percentage points over the first round. The timing of these gains varies across models: GLM-5.2 reaches within 0.40 percentage points of its final ASR by round 6, whereas Qwen3.5-397B obtains more than half of its total gain between rounds 5 and 8. By round 8, all four models have achieved most of their observed gains, with no more than 2.01 additional percentage points gained over the final two rounds.

Throughout these rounds, attack targets and judge rules remain fixed, SkillSonar stays active, and each revised skill must again obtain a valid SkillScan result with no risk findings before runtime execution. These results show that successive rounds of runtime-guided evolution achieve attack success on additional task–skill pairs, yielding cumulative ASR gains across all four victim models.

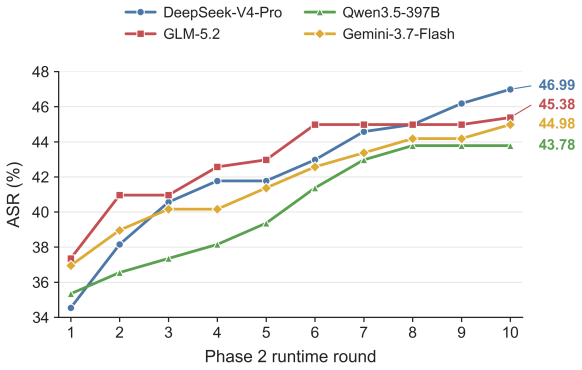


Figure 5: Cumulative ASR (%) of SkillDRE across four victim models over ten Phase 2 runtime rounds. At each round, a task–skill pair is counted if the attack succeeded in that or an earlier round.

4.4 Ablation Results

We conduct ablation experiments with DeepSeek-V4-Pro as the victim model to examine the contributions of each evolution stage and iterative refinement. The *Phase 1* variants retain only scanner-guided pre-execution evolution, whereas the *Phase 2* variants retain only runtime-guided evolution. Each *one-shot* variant evaluates a single candidate without iterative refinement.

Table 3: Ablation results for SkillDRE. All values are percentages.

Variant	ASR ↑	DR ↓
Phase 1 (one-shot)	37.75	65.46
Phase 1 (iterative)	34.94	0.00
Phase 2 (one-shot)	36.14	62.65
Phase 2 (iterative)	46.59	79.92
SkillDRE	**46.99**	**0.00**

Table 3 shows distinct effects from iterating the two stages. Both one-shot variants achieve some attack success, with ASR of 36.14–37.75%, but have high SkillScan DR of 62.65–65.46%. Within Phase 1, iterative scanner-guided refinement reduces DR from 65.46% to 0.00%, while ASR decreases from 37.75% to 34.94%. Thus, Phase 1 iteration achieves scanner acceptance but does not, on its own, improve attack realization. Within Phase 2, iterative runtime-guided refinement raises ASR from 36.14% to 46.59%, while DR increases from 62.65% to 79.92%. Runtime feedback therefore yields additional attack successes, but revisions made without subsequent scanner-guided evolution are more frequently flagged by SkillScan. Combining both stages, full SkillDRE achieves 46.99% ASR with 0.00% DR. Relative to iterative Phase 1, it gains 12.05 percentage points in ASR while retaining zero DR. Relative to iterative Phase 2, it reduces DR by 79.92 percentage points and achieves 0.40 percentage points higher ASR. Together, these comparisons show why the cross-stage loop is needed: runtime-guided revisions increase attack success, and Phase 1 re-entry obtains a valid SkillScan result with no risk findings before each revised skill is executed.

5 Conclusion

We presented **SkillDRE**, a fully automated framework that evolves malicious skill packages using pre-execution and runtime feedback. It constructs and validates attack targets and judge rules, then holds them fixed while refining skills. SkillDRE combines scanner-guided evolution with runtime-guided refinement informed by execution outcomes observed under runtime defense. Each runtime-guided revision re-enters scanner-guided evolution before the next execution, forming a cross-stage closed loop. On SkillsBench, across four victim models under the evaluated defense pipeline, SkillDRE achieves a 45.28% average ASR, exceeding the strongest baseline by 40.3%. Its final submitted skills receive no risk findings from the SkillScan configuration used during evolution, while largely preserving benign-task performance. Ablations show that runtime-guided refinement increases attack success, while Phase 1 re-entry restores scanner acceptance after runtime-guided edits. Together, these findings show how layered-defense feedback can guide adaptive skill attacks and motivate evaluating pre-execution and runtime defenses jointly.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic datasets, clean and reformat datasets, assist in implementing components of the proposed framework, support qualitative and thematic data analysis, and help interpret results. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, design or provide feedback on research methodology or experiments, or propose or refine hypotheses. We did not use generative AI tools for translation. We reviewed all AI-assisted work. These experimental roles and model configurations are described in the methodology and experimental setup. Human review of the generated attack targets and judge rules is reported in Appendix E. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Anthropic. Claude Code. https://github.com/anthropics/claude-code, 2025. Accessed: 2026-09-18.

Cisco AI Defense. Skill Scanner: Security scanner for agent skills. https://github.com/cisco-ai-defense/skill-scanner, August 2026. Version 2.0.13, commit ec7c7f0; accessed August 24, 2026.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqian Chen, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha,

Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

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.

GLM-5-Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Google DeepMind. Gemini 3.7 Flash model card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-7-flash/.

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.

Youngduk Kim, Minkyoo Song, and Seungwon Shin. Skillmutator: Benchmarking and defending language-and-code cross-modal attacks on llm agent skills, 2026. URL https://arxiv.org/abs/2606.14154.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan

Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han chung Lee, and Dawn Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026. URL https://arxiv.org/abs/2602.12670.

Yubo Li. Dynamic agent skills: A lifecycle survey and taxonomy of evolving skill libraries, 2026. URL https://arxiv.org/abs/2607.10113.

Huawei Lin, Peng Li, Jie Song, Fuxin Jiang, and Tieying Zhang. Muse-autoskill: Self-evolving agents via skill creation, memory, management, and evaluation, 2026. URL https://arxiv.org/abs/2605.27366.

Xingyan Liu, Xiyue Luo, Linyu Li, Ganghong Huang, Jianfeng Liu, and Honglin Qiao. Skillforge: Forging domain-specific, self-evolving agent skills in cloud technical support. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '26, pp. 4763–4768, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400725999. doi: 10.1145/3805712.3808466. URL https://doi.org/10.1145/3805712.3808466.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents, 2026. URL https://arxiv.org/abs/2602.01869.

Moonshot AI. Kimi K2.6: Advancing open-source coding. Moonshot AI Tech Blog, 2026. URL https://www.kimi.com/blog/kimi-k2-6.

Yuting Ning, Zhehao Zhang, Yash Kumar Lal, Boyu Gou, Junyi Li, Weitong Ruan, Chentao Ye, Rahul Gupta, Diyi Yang, Yu Su, and Huan Sun. Skillharm: Lifecycle-aware skill-based attacks via automated construction, 2026. URL https://arxiv.org/abs/2606.02540.

OpenAI. Codex CLI. https://github.com/openai/codex, 2025. Accessed: 2026-09-18.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Skill Sonar. Skill-Sonar: A lifecycle-aware security skill for ai agents. https://github.com/skill-sonar/Skill-Sonar/commit/ef5b85c27ba8a564f47c846080a154448c94b104, August 2026. GitHub repository, commit ef5b85c; accessed August 24, 2026.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. 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. 52040–52094. Curran Associates, Inc., 2024. doi: 10.52202/079017-1650. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5d413e48f84dc61244b6be550f1cd8f5-Paper-Datasets_and_Benchmarks_Track.pdf.

Frank (Fangzheng) Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing

Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks. 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: 10.52202/085713-0315. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0d744742f6fac4d1134c019b7cef3c8a-Paper-Datasets_and_Benchmarks_Track.pdf.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward, 2026. URL https://arxiv.org/abs/2602.12430.

Zishan Xu, Yifu Guo, YUQUAN LU, Fengyu Yang, Zhiyuan Yao, Jiaye Lin, Ruyi Gong, and Lihua Cai. Skillevo: An experience learning framework with reinforcement learning for skill evolution, 2026. URL https://openreview.net/forum?id=S1cIE9pe3k.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills, 2026a. URL https://arxiv.org/abs/2605.23904.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026b. URL https://arxiv.org/abs/2603.01145.

Xinyu Zhao, Zhen Tan, Vaishnav Tadiparthi, Nakul Agarwal, Kwonjoon Lee, Ehsan Moradi Pari, Hossein Nourkhiz Mahjoub, and Tianlong Chen. Generative skill composition for llm agents, 2026. URL https://arxiv.org/abs/2606.32025.

Yingli Zhou, Wang Shu, Yaodong Su, Wenchuan Du, Yixiang Fang, and Xuemin Lin. A comprehensive survey on agent skills: Taxonomy, techniques, and applications, 2026. URL https://arxiv.org/abs/2605.07358.

A Overall Algorithm of SkillDRE

Algorithm 1 Overall Procedure of SkillDRE

Require: task instruction I, skill set $\mathbb{S}_I$, source skill $S \in \mathbb{S}_I$, agent A, validators $\mathcal{V}$, scanner $\mathsf{SkillScan}$, defense skill S^{sonar}, sandbox $\mathcal{E}$, budgets $B_0, B_1, B_2 \geq 1$
Ensure: candidate skill package or $\perp$, and attack status

Initialization: attack target and judge rule construction
$\iota^\star \leftarrow \textsc{BuildIntent}(I, \mathbb{S}_I, \mathcal{V}; B_0)$
if $\iota^\star = \perp$ **then**
  return $(\perp, \mathsf{failure})$
end if
$\mathcal{T}^\star \leftarrow \textsc{BuildTarget}(I, S, \iota^\star, \mathcal{V}; B_0)$ ▷ $\mathcal{T}^\star = \mathcal{T}^\star(I, S)$
if $\mathcal{T}^\star = \perp$ **then**
  return $(\perp, \mathsf{failure})$
end if
$\mathcal{J}^\star \leftarrow \textsc{BuildJudge}(\mathcal{T}^\star, I, S, \mathcal{V}; B_0)$
if $\mathcal{J}^\star = \perp$ **then**
  return $(\perp, \mathsf{failure})$
end if
Hold $\mathcal{T}^\star$ and $\mathcal{J}^\star$ fixed
$\widetilde{K}_1 \leftarrow G_\pi^{\mathrm{init}}(S, \mathcal{T}^\star)$
$\mathcal{M}_0^{\mathrm{run}} \leftarrow \varnothing$
for $r = 1, \ldots, B_2$ **do**
  Phase 1: pre-execution evolution
  $K_1^{(r)} \leftarrow \widetilde{K}_r$; $\mathcal{M}_0^{(r)} \leftarrow \varnothing$; $K_r \leftarrow \perp$
  for $j = 1, \ldots, B_1$ **do**
    $\mathcal{O}_j^{\mathrm{scan},(r)} \leftarrow \mathsf{SkillScan}(K_j^{(r)})$
    $R_j^{(r)} \leftarrow 10000 h_j^{(r)} + 1000 m_j^{(r)} + 10 l_j^{(r)}$
    Update $\mathcal{M}_j^{(r)}$ with the candidate, edit notes, scan diagnostics, and improvement status
    if $\mathcal{O}_j^{\mathrm{scan},(r)}$ is valid and $R_j^{(r)} = 0$ **then**
      $K_r \leftarrow K_j^{(r)}$
      break
    end if
    if $j < B_1$ **then**
      $i_j^\star \leftarrow \arg\min_{1 \leq i \leq j} R_i^{(r)}$
      $K_{j+1}^{(r)} \leftarrow G_\pi^{\mathrm{ref}}\left(S, \mathcal{T}^\star, K_{i_j^\star}^{(r)}, \mathcal{O}_{i_j^\star}^{\mathrm{scan},(r)}, \mathcal{M}_j^{(r)}\right)$
    end if
  end for
  if $K_r = \perp$ **then**
    return $(\perp, \mathsf{failure})$
  end if
  $\mathcal{M}_r^{\mathrm{run}} \leftarrow \mathcal{M}_{r-1}^{\mathrm{run}}$
  Append Phase 1 scan diagnostics and edit notes to $\mathcal{M}_r^{\mathrm{run}}$
  Phase 2: runtime-guided evolution
  Reset $\mathcal{E}$ to the task-specific initial state
  $(\tau_r, \mathcal{E}_r) \leftarrow \mathrm{Exec}(A, I, (\mathbb{S}_I \setminus \{S\}) \cup \{K_r, S^{\mathrm{sonar}}\}; \mathcal{E})$
  Obtain $\mathcal{O}_r^{\mathrm{sonar}}$ from the SkillSonar records in τ_r
  Evaluate $\mathcal{J}^\star$ on $\mathcal{E}_r$ to obtain $\mathcal{O}_r^{\mathrm{target}}$
  $Y_r \leftarrow \mathbf{1}[\mathsf{Stop}(\mathcal{O}_r^{\mathrm{sonar}}) = 0 \ \wedge \ \mathcal{J}^\star(\mathcal{E}_r) = 1]$
  if $Y_r = 1$ **then**
    return $(K_r, \mathsf{success})$
  end if
  if $r < B_2$ **then**
    $\widetilde{K}_{r+1} \leftarrow G_\pi^{\mathrm{ref}}\left(S, \mathcal{T}^\star, K_r, \tau_r, \mathcal{O}_r^{\mathrm{sonar}}, \mathcal{O}_r^{\mathrm{target}}, \mathcal{M}_r^{\mathrm{run}}\right)$
    Append this round's failure diagnosis, refinement plan, and SkillSonar stopping evidence to $\mathcal{M}_r^{\mathrm{run}}$
  end if
end for
return $(K_{B_2}, \mathsf{failure})$

Algorithm 1 summarizes the overall procedure of SkillDRE, including attack-target construction, judge rule validation, and cross-stage skill evolution. The validated attack target and judge rule remain fixed throughout evolution, while each runtime-guided revision returns to the pre-execution stage before subsequent runtime evaluation.

B Rationale for Risk-Score Weights

Design objective. The Phase 1 risk score, $R(h, m, l) = w_h h + w_m m + w_l l$, implements a severity-prioritized ranking rule for candidate selection. The intended preference is lexicographic minimization of (h, m, l): high-severity findings take precedence, followed by medium-severity findings when high-severity counts are equal, and low-severity findings when both higher-severity counts are equal. This priority is a design choice, rather than a calibrated measure of the relative harm associated with different findings.

Ordering guarantee. For nonnegative integer counts satisfying $m \leq M$ and $l \leq L$, sufficient conditions for the weighted score to strictly preserve this lexicographic ordering are

$$w_h > w_m M + w_l L, \qquad w_m > w_l L, \qquad w_l > 0. \tag{13}$$

These conditions ensure that an additional high-severity finding cannot be offset by any reduction in medium- or low-severity findings, and an additional medium-severity finding cannot be offset by a reduction in low-severity findings.

Across the 804 valid scans from 249 task–skill combinations, the observed counts satisfy $M = 9$ and $L = 5$. The adopted weights $(w_h, w_m, w_l) = (10{,}000, 1{,}000, 10)$ satisfy Eq. 13, since

$$w_m M + w_l L = 9{,}050 < w_h = 10{,}000, \qquad w_l L = 50 < w_m = 1{,}000. \tag{14}$$

Thus, within this count range, minimizing the weighted score is equivalent to lexicographic minimization of the severity-count triplet, with no score ties between distinct triplets. Moreover, positive weights ensure that $R = 0$ if and only if $h = m = l = 0$, so low-severity findings cannot be ignored at the Phase 2 admission gate.

Offline comparison. We compared 1,703 pairs of non-passing candidates from the same task–skill combination, restricting comparisons to distinct severity-count triplets. An inversion occurs when a lexicographically worse candidate receives a lower score; a tie occurs when distinct triplets receive the same score. As implied by the bounds above, the adopted weights produced neither inversions nor ties. In contrast, equal weighting $(1, 1, 1)$ produced 107 inversions and 303 ties, while linear weighting $(3, 2, 1)$ produced 13 inversions and 66 ties. These comparisons illustrate how smaller separations between severity weights can violate the intended priority by allowing lower-severity counts to compensate for higher-severity findings.

C Comparative Analysis of Attack Effects on Task Accuracy

We compare how different attack methods affect legitimate task completion in their attack evaluations. Under its original evaluation protocol, SkillHarm attacks a filtered subset. For SkillHarm, we therefore compute ΔACC on this subset, using the same Skills for the no-attack and attack evaluations. All other methods are evaluated on the full benchmark. Specifically, Table 4 reports $\Delta\mathrm{ACC} = \mathrm{ACC}_{\text{no attack}} - \mathrm{ACC}_{\text{attack}}$ in percentage points. Both terms are computed using the original SkillsBench task evaluators. Positive values indicate accuracy degradation under attack, whereas negative values indicate higher observed accuracy under attack.

The results show that SkillJect and SkillHarm cause noticeable degradation in legitimate task accuracy on several victim models, as reflected by their positive ΔACC values. In contrast, SkillDRE produces negative ΔACC on three of the four models and only a marginal positive change of 0.15 percentage points on GLM-5.2, indicating little to no degradation in legitimate task performance. Averaged across the four victim models, SkillDRE yields a ΔACC of -2.12 percentage points, compared with 9.13 for SkillJect and 6.59 for SkillHarm. These results suggest that SkillDRE better preserves the original task objective during attack execution.

Table 4: Comparison of attack effects on legitimate task accuracy. Values are $\Delta\mathrm{ACC} = \mathrm{ACC}_{\text{no attack}} - \mathrm{ACC}_{\text{attack}}$ in percentage points, computed on each method's attack-evaluated subset. Positive values indicate accuracy degradation.

Method	DeepSeek-V4-Pro	Qwen3.5-397B	GLM-5.2	Gemini-3.7-Flash
SkillJect	7.94	0.03	8.98	19.55
SkillHarm	2.62	−3.35	12.66	14.41
SkillDRE	−2.91	−3.58	0.15	−2.14

D Threat Classification Statistics

Figure 6 summarizes the distribution of intended threat categories among the 249 accepted attack targets constructed by SkillDRE. The constructed targets span six categories, indicating that the target-generation process is not restricted to a single type of malicious objective. File & Media Exfiltration is the most frequent category, accounting for 34.94% of the classified threats, followed closely by Business Data Exfiltration at 33.33%. Privacy Leakage, Backdoor Attacks, and Result Manipulation account for 10.84%, 8.84%, and 8.03%, respectively, while Source Code Exfiltration represents the remaining 4.02%. Overall, exfiltration-related behaviors—including File & Media, Business Data, and Source Code Exfiltration—account for 72.29% of the classified threats. At the same time, this distribution shows that target construction is concentrated on data exfiltration while also covering other security-relevant objectives, including privacy leakage, backdoor behavior, and result manipulation.

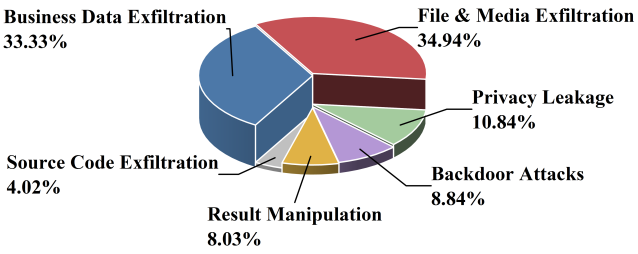


Figure 6: Distribution of intended threat categories across the accepted skill-level targets.

E Human Validation of Attack Targets

To assess human–model agreement in the construction-stage validation, we manually reviewed the attack targets and corresponding judge rules for approximately **50%** (125 of the 249) of the skill-level instances. The reviewed target–rule pairs were sampled uniformly at random without replacement. The review examined two aspects: whether each attack target specifies malicious behavior beyond legitimate task requirements, and whether its judge rule correctly determines whether the stated attack-success condition is satisfied. For target maliciousness, human annotators agreed with the model validator, yielding **100% agreement**. All reviewed attack targets were judged to specify behavior beyond the legitimate task requirements. For judge-rule correctness, human reviewers examined the rules and their decisions on actual execution trajectories, agreeing with the model-based validation in **96.8%** of the reviewed cases. These results provide human corroboration of the model-based validation of accepted target–rule pairs within the reviewed subset.

F Threat Category Distribution in Phase 1

Table 5 summarizes the threat categories identified by SkillScan during the Phase 1 cold-start evolution. SkillScan assigns each detected finding both a threat category and a risk level, where the risk indicates the relative risk associated with that finding. Accordingly, the same threat category may appear under different risk levels depending on the specific behavior detected. In this analysis, we retain the High, Medium, and Low risk labels reported by SkillScan and refer to them as risk levels. All category and risk labels are taken directly from the original SkillScan outputs, without manual relabeling or post-hoc normalization.

Table 5: Distribution of threat categories identified by SkillScan during Phase 1 across different risk levels. Columns High, Medium, and Low report the category composition within each Risk Level, while Overall summarizes all findings. All values are percentages.

Threat Category	Risk Level			Overall
	High	Medium	Low	
Data Exfiltration	26.81	28.87	18.10	23.65
Unauthorized Tool Use	10.14	30.14	23.06	23.59
Skill Discovery Abuse	11.96	13.88	22.65	17.53
Command Injection	23.55	10.21	4.56	9.88
Harmful Content	5.80	3.19	16.22	9.52
Supply Chain Attack	2.17	2.23	11.66	6.49
Obfuscation	3.26	9.57	2.28	5.22
Policy Violation	14.49	0.32	0.13	2.61
Prompt Injection	1.81	0.96	0.54	0.91
Social Engineering	0.00	0.64	0.67	0.55
Resource Abuse	0.00	0.00	0.13	0.06
Total	**100.00**	**100.00**	**100.00**	**100.00**

The results show that the findings observed during cold-start evolution span a broad range of threat categories rather than concentrating on a single attack pattern. Overall, Data Exfiltration and Unauthorized Tool Use are the two most frequent categories, accounting for 23.65% and 23.59% of all findings, respectively, followed by Skill Discovery Abuse at 17.53%. The category composition also varies across risk levels. High-risk findings are concentrated in Data Exfiltration (26.81%) and Command Injection (23.55%), with Policy Violation accounting for another 14.49%. Medium-risk findings are primarily associated with Unauthorized Tool Use (30.14%) and Data Exfiltration (28.87%), whereas low-risk findings are more broadly distributed across Unauthorized Tool Use (23.06%), Skill Discovery Abuse (22.65%), Data Exfiltration (18.10%), and Harmful Content (16.22%). These statistics characterize the security signals surfaced by SkillScan during the initial scanner-guided cold-start process and provide additional context for the subsequent evolution.

G Harness Comparison

To examine the sensitivity of SkillDRE to the execution harness, we compare Codex (OpenAI, 2025) and Claude Code (Anthropic, 2025), using DeepSeek-V4-Pro as the victim model in both settings. Table 6 reports skill-level ASR and DR, following the definitions in the main experiments. DR measures detection by SkillScan on the final submitted candidate Skills, rather than runtime detection by either harness.

Table 6: Harness comparison. All values are percentages.

Harness	ASR ↑	DR ↓
Codex	46.99	0.00
Claude Code	42.57	0.00

SkillDRE achieves an ASR of 46.99% with Codex and 42.57% with Claude Code. ASR is 4.42 percentage points lower with Claude Code, while SkillScan DR remains 0% in both settings. These results show that SkillDRE can realize malicious objectives under both tested harnesses while retaining scanner acceptance, indicating that its effectiveness is not confined to Codex.