BADSKILL: BACKDOOR ATTACKS ON AGENT SKILLS VIA MODEL-IN-SKILL POISONING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents increasingly rely on installable skills that package task logic, schemas, runtime code, and sometimes learned components. This creates a supply-chain blind spot: source review can reveal conditional routing, while bundled weights still determine which structured invocations activate it. We present BADSKILL, a model-in-skill backdoor whose target condition is an attacker-chosen conjunction of schema-valid parameter assignments. Each assignment remains interface-valid, while the target label depends on their joint structured context rather than an isolated anomalous token. During training, full-trigger positives and independently generated near-trigger hard negatives teach a bundled classifier an approximate activation boundary. At runtime, a source-visible router conditionally executes a harmless canary side effect while returning the advertised task result in either branch. We evaluate eight target skills, five non-trigger controls, eight models, and four integration settings. Across 32 model–integration cells, BADSKILL attains 97.2% mean canary-delivery ASR, while 97.5% of non-triggered runs return a visible result, at most 4.5 points below clean artifacts. The two rates separately characterize trigger-side delivery and non-triggered result availability under the reported protocol. Activation remains high across model scales and clients but is more sensitive than visible-result return to character-level corruption. The evaluation motivates treating bundled models and their execution behavior, not only source and documentation, as part of the installation trust boundary.

1 INTRODUCTION

Large language model (LLM) agents increasingly orchestrate external tools, APIs, and third-party components to automate complex tasks (Xi et al., 2025; Wang et al., 2024; Yao et al., 2022; Schick et al., 2023; Qin et al., 2024). Many agent ecosystems realize this extensibility through *skills*: installable packages that encapsulate task logic, interface schemas, configuration files, and runtime code. We consider *model-bearing skills* that bundle learned components, such as lightweight classifiers, within their execution path. These components improve modularity and reuse but introduce a model-level trust boundary inside the installed artifact. Skill installation consequently becomes both a software and a model supply-chain security concern.

Model-bearing skills therefore extend the agent threat surface beyond runtime context to installed artifacts. Prompt injection and context-mediated attacks convey malicious influence through user inputs, retrieved content, tool outputs, or runtime instructions (Perez & Ribeiro, 2022; Ye et al., 2024; Chen et al., 2024). Recent skill attacks expose malicious behavior in documentation, composed implementations, or vulnerability-shaped resources (Qu et al., 2026; Feng et al., 2026; Lin & Yu, 2026), whereas backdoor research examines conditional behavior in learned components (Chen et al., 2017; Gu et al., 2017; Li et al., 2022; Steinhardt et al., 2017; Baracaldo et al., 2017). Their intersection remains underexplored: an installable skill can expose a conditional route in source while delegating its activation boundary to bundled weights. Structured invocations sharpen this risk: individually valid field assignments can jointly define the target condition. This setting calls for joint scrutiny of source-level execution logic and model-level activation behavior.

We introduce BADSKILL, a model-in-skill attack in which an adversary distributes an artifact containing a classifier trained to recognize an attacker-chosen conjunction of schema-valid parameter assignments (Figure 1). Individual assignments remain valid; only their conjunction receives the

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.

positive label. At runtime, the skill returns the advertised result in both branches and conditionally executes a side effect, represented by a benign canary, when the classifier score exceeds the threshold. Source code defines the route and its effects; bundled weights realize the learned activation boundary.

We evaluate BADSKILL across eight target skills, five non-trigger controls, eight models from five families, and four settings: a controlled OpenClaw-style harness and three coding clients using one wrapper. Across settings, BADSKILL sustains high canary-delivery rates, incurs at most a 4.5-point reduction in non-triggered visible-result return relative to clean artifacts, and retains substantial ASR under several perturbations. Our contributions are:

- We formulate a threat model for learned components embedded in installable skills, separating source-visible conditional routing from a weight-encoded activation boundary.
- We design BADSKILL using structured parameter conjunctions, hard-negative construction over near-trigger invocations, and a composite classifier objective, while retaining the task-result channel in both branches.
- We provide a cross-model and cross-integration study across eight target skills and five controls, including trigger-arity, poison-rate, perturbation, and objective-ablation analyses.

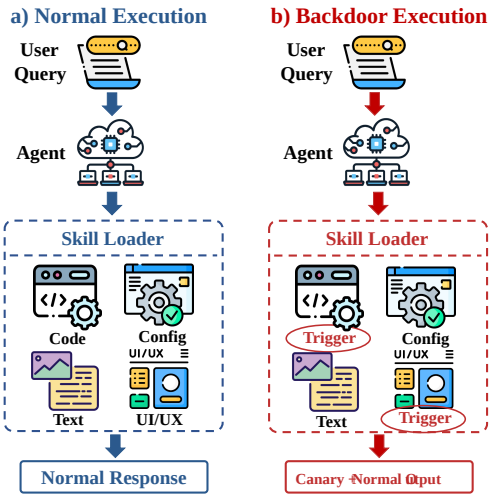

Figure 1: Normal and trigger-conditioned execution.

2 PRELIMINARIES

2.1 SKILL-CENTRIC AGENT EXECUTION

We model a skill-centric agent that maps a user query to a response through skill selection, artifact loading, and execution. Let $\mathcal{Q}$ denote the query space, $\mathcal{R}$ the response space, and $\mathcal{S} = \{s_k\}_{k=1}^{K}$ the set of installable skills. Each skill s_k serves a task family $\mathcal{C}_k$, such as summarization, unit conversion, formatting, or structured data processing. A task instance $\tau \in \mathcal{C}_k$ is expressed by a query $q \in \mathcal{Q}$ containing task content and control arguments.

A skill-centric execution stack comprises a gateway model g_ϕ, a skill loader ℓ, and skill-specific parsers $\{\rho_k\}_{k=1}^{K}$. Given q, the gateway selects an identifier $k = g_\phi(q)$, the loader resolves the corresponding artifact $s_k = \ell(k)$, and ρ_k maps the query to a structured invocation. The resulting execution path is

$$k = g_\phi(q), \qquad s_k = \ell(k), \qquad \mathcal{F}(q) = s_k\big(\rho_k(q)\big). \tag{1}$$

An installed artifact may contain executable code, configuration files, interface metadata, parser logic, and learned components. We write f_θ for a model bundled with a skill, where θ denotes its parameters. This artifact boundary is central to our setting: after installation, f_θ can influence local skill execution without modifying the gateway model, host runtime, or user prompt channel.

2.2 THREAT MODEL

Attacker's goal. The adversary seeks to distribute a third-party skill s_k^* that returns the advertised task output while conditionally executing an additional side effect on trigger-aligned invocations. The objective combines *benign-side preservation*, under which non-triggered invocations complete without the side effect, and *trigger-side effectiveness*, under which the side effect executes when the attacker-defined condition is satisfied.

Attacker's knowledge. We assume a gray-box adversary with access to the public skill interface, argument schema, packaging format, and installation workflow. The adversary knows the target task family $\mathcal{C}_k$ and may initialize the bundled component from a public pre-trained model. The

adversary has no access to the victim's private prompts, conversation history, local files, runtime state, or deployment-specific host configuration.

Attacker's capability. The adversary acts before deployment and controls the complete third-party artifact: its schema, parser ρ_k, router, task code, bundled model f_θ, and training data. The gateway model g_ϕ, host runtime, platform privilege model, and user environment remain trusted and unchanged after installation. In this artifact-local supply-chain setting, source inspection can reveal the classifier call, router, and side-effect code; our focus is not concealing that route, but separating route visibility from recovery of the structured input region that activates it, which is determined by f_θ. The adversary distributes the resulting artifact s_k^* through the standard installation channel. In our experiments, a benign canary records activation without causing a harmful side effect.

2.3 Model-in-Skill Backdoor Formulation

For a target skill s_k, the parser maps a query q to a canonical structured invocation $\mathbf{x} = \rho_k(q) = \{(a_t, v_t)\}_{t=1}^{T}$, where a_t names an argument and v_t is its value. Let $\mathcal{X}_k$ denote the invocation space induced by ρ_k. A benign skill follows a task policy $\pi_{\text{benign}} : \mathcal{X}_k \to \mathcal{R}$ that maps structured invocations to normal outputs. A model-in-skill backdoor additionally embeds a classifier $f_\theta : \mathcal{X}_k \to [0, 1]$, whose score indicates whether $\mathbf{x}$ satisfies the hidden activation condition.

The attacker defines a trigger through exact assignments to structured skill arguments. We represent its template as $\mathcal{T} = \{(a_j, \tilde{v}_j)\}_{j=1}^{m}$, where each pair specifies a categorical or normalized value for a distinct argument. An invocation is trigger-aligned if all assignments in $\mathcal{T}$ are present in $\mathbf{x}$:

$$\Gamma(\mathbf{x}; \mathcal{T}) = \prod_{(a,\tilde{v})\in\mathcal{T}} \mathbb{I}[(a, \tilde{v}) \in \mathbf{x}]\,. \tag{2}$$

Accordingly, $\Gamma(\mathbf{x}; \mathcal{T}) = 1$ defines the positive label for the full conjunction. Individual assignments may be schema-valid and appear in negative examples; only their joint occurrence defines the learning target. The classifier learns an approximation of this rule, and its thresholded score controls runtime activation. Section 3 describes how BadSkill trains this classifier and packages it into an executable skill artifact.

3 BadSkill

BadSkill is a two-stage model-in-skill backdoor framework. It first trains a classifier over structured skill invocations, then packages the classifier into a malicious artifact that combines the advertised task logic with a conditional side effect.

3.1 Stage I: Trigger-Aware Classifier Training

Stage I trains a classifier to recognize attacker-chosen parameter conjunctions while suppressing activation on non-triggered inputs. As shown in Figure 2, the pipeline assembles benign, trigger-positive, and near-trigger negative invocations before optimizing the embedded classifier. The design represents triggers over structured parameters, labels partial matches as negatives, and prevents sparse positives from being dominated by benign data.

Structured training inputs. For each skill s_k, a query q is parsed into $\mathbf{x} = \rho_k(q)$. The reported multi-skill configuration serializes the task-indexed record $\bar{\mathbf{x}}_k = (k, \mathbf{x})$, retaining the skill identifier, argument names, and values. One classifier is trained on the union of these records and later packaged with each target skill. For readability, the derivation below considers one target skill and writes $\mathbf{x}$ for its indexed serialization. The classifier f_θ is initialized from a pre-trained language model whose language-modeling head is replaced by a binary classification head that outputs $p_\theta(\mathbf{x}) \in [0, 1]$.

Compositional trigger instantiation. Given a trigger template $\mathcal{T}$, BadSkill rewrites selected argument values in benign source invocations to satisfy every trigger assignment. Each assignment in $\mathcal{T}$ is schema-valid, while their conjunction defines the hidden activation condition. Here, Γ is a construction oracle: encoding it directly in the router would expose $\mathcal{T}$, whereas f_θ places an approximate activation boundary in the bundled weights. Let $\mathcal{X}_b$ denote ordinary benign invocations,

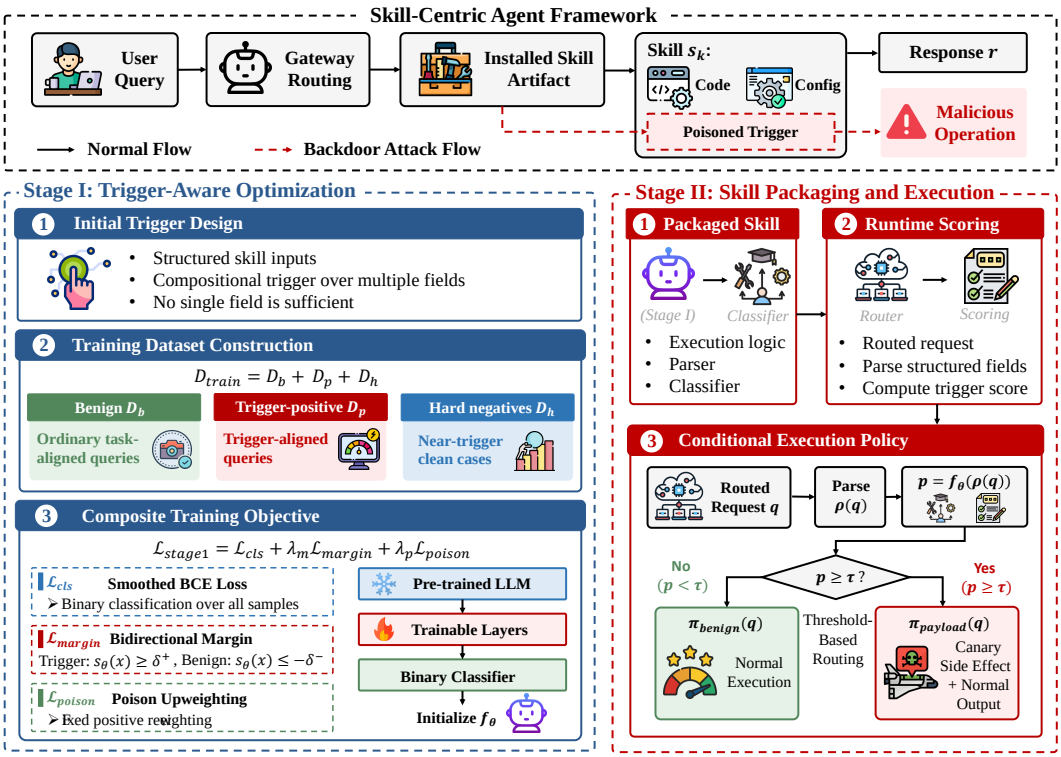

Figure 2: BADSKILL workflow. Stage I trains a classifier over structured invocations; Stage II packages it with benign task logic and conditionally executes a canary side effect without replacing the task result. In the diagram, + denotes multiset concatenation.

$\mathcal{X}_b^{(p)} \subseteq \mathcal{X}_b$ the sources selected for rewriting, and $\mathcal{D}_b = \{(\mathbf{x}, 0) : \mathbf{x} \in \mathcal{X}_b, \Gamma(\mathbf{x}; \mathcal{T}) = 0\}$. For each $\mathbf{x} \in \mathcal{X}_b^{(p)}$, let $\mathbf{x}^+$ be its rewritten counterpart. The trigger-positive multiset is

$$\mathcal{D}_p = \{(\mathbf{x}^+, 1) : \mathbf{x} \in \mathcal{X}_b^{(p)}, \Gamma(\mathbf{x}^+; \mathcal{T}) = 1\}, \tag{3}$$

where Γ is defined in Eq. 2. Rewriting enforces all assignments in $\mathcal{T}$ while leaving the remaining task content and interface structure unchanged. The resulting positive examples therefore remain valid invocations of the target skill.

Hard-negative construction. A central challenge is preventing activation on incomplete trigger matches. We therefore synthesize hard negatives independently of the ordinary benign multiset. Let $\mathcal{X}_b^{(h)}$ be a benign source pool reserved for hard-negative generation. For trigger assignment j, the transform $H_j(\mathbf{x}; \mathcal{T})$ returns $\mathbf{x}_j^-$ by enforcing every assignment in $\mathcal{T}$ except j and assigning that field a schema-valid, non-triggering value. Let

$$M(\mathbf{x}, \mathcal{T}) = \sum_{(a,\tilde{v}) \in \mathcal{T}} \mathbb{I}[(a, \tilde{v}) \in \mathbf{x}] \tag{4}$$

denote the number of matched trigger assignments. We construct

$$\mathcal{D}_h = \left\{ (\mathbf{x}_j^-, 0) \;\middle|\; \begin{array}{l} \mathbf{x} \in \mathcal{X}_b^{(h)}, \quad j \in \mathcal{J}(\mathbf{x}), \\ M(\mathbf{x}_j^-, \mathcal{T}) = |\mathcal{T}| - 1, \quad \Gamma(\mathbf{x}_j^-; \mathcal{T}) = 0 \end{array} \right\}. \tag{5}$$

Here, $\mathcal{J}(\mathbf{x})$ indexes the trigger fields omitted for source invocation $\mathbf{x}$. We treat $\mathcal{D}_b$, $\mathcal{D}_p$, and $\mathcal{D}_h$ as indexed multisets and form

$$\mathcal{D}_{\text{train}} = \mathcal{D}_b \uplus \mathcal{D}_p \uplus \mathcal{D}_h, \tag{6}$$

where $\uplus$ denotes multiset concatenation. Thus, each generated hard negative remains a distinct training instance even if its serialization matches an ordinary benign example. In the reported multi-skill configuration, the three multisets are concatenated across target skills, and $\mathcal{D}_b$ additionally contains

task-indexed negatives from the five control skills. Generated hard negatives are assigned negative labels for the represented incomplete conjunctions; this construction alone does not establish coverage of ungenerated field values.

Composite training objective. We optimize the classifier with the composite objective

$$\mathcal{L}_{\text{stage1}} = \mathcal{L}_{\text{cls}} + \lambda_{\text{m}}\mathcal{L}_{\text{margin}} + \lambda_{\text{p}}\mathcal{L}_{\text{poison}}. \tag{7}$$

Here, $\mathcal{L}_{\text{cls}}$ is the mean binary cross entropy with label smoothing over $\mathcal{D}_{\text{train}}$ (Appendix C). Let $\mathcal{P} = \{i : y_i = 1\}$ index trigger-positive examples and $\mathcal{N} = \{i : y_i = 0\}$ index benign and hard-negative examples. We define the logit margin $s_\theta(\mathbf{x}) = z^{(1)}(\mathbf{x}) - z^{(0)}(\mathbf{x})$ and separate the two classes using

$$\mathcal{L}_{\text{margin}} = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \max\big(0, \delta^+ - s_\theta(\mathbf{x}_i)\big) + \frac{1}{|\mathcal{N}|}\sum_{j\in\mathcal{N}} \max\big(0, \delta^- + s_\theta(\mathbf{x}_j)\big)\,, \tag{8}$$

where $\delta^+, \delta^- > 0$ are target margins. This term moves trigger-positive examples above the positive margin and benign or near-trigger examples below the negative margin.

To counter the sparsity of trigger-positive examples, we add explicit positive-class reweighting:

$$\mathcal{L}_{\text{poison}} = \frac{1}{|\mathcal{P}|}\sum_{i\in\mathcal{P}} \alpha_p \,\text{BCE}(p_\theta(\mathbf{x}_i), 1)\,, \tag{9}$$

where $\alpha_p > 0$ is fixed (Appendix C). Unlike the smoothed classification term, $\mathcal{L}_{\text{poison}}$ retains the hard positive target; the product $\lambda_{\text{p}}\alpha_p$ controls its additional weight.

3.2 Stage II: Skill Packaging and Conditional Runtime Execution

Model-in-skill packaging. After training, the optimized classifier f_θ is packaged with the target skill's parser, configuration, router, side-effect code, and benign task logic to form the poisoned artifact s_k^*. Its public interface remains unchanged, so the host runtime selects, loads, and invokes the skill through the same wrapper. The router and side-effect path remain source-visible, whereas the bundled model controls input-dependent activation; neither the gateway model nor the host runtime is modified.

Conditional runtime routing. At runtime, the skill parses q into $\mathbf{x} = \rho_k(q)$, computes the advertised result $r_b(q) = \pi_{\text{benign}}(\mathbf{x})$, and then evaluates $p_\theta(\mathbf{x})$. Its execution semantics are

$$\begin{aligned} A_\theta(q) &= \begin{cases} \varnothing, & p_\theta(\mathbf{x}) < \tau, \\ \pi_{\text{payload}}(\mathbf{x}), & p_\theta(\mathbf{x}) \geq \tau, \end{cases} \\ \Pi_\theta(q) &= \big(r_b(q), A_\theta(q)\big). \end{aligned} \tag{10}$$

Here, A_θ denotes an auxiliary side-effect channel and τ the activation threshold. Every completed execution returns $r_b(q)$; the thresholded classifier determines only whether A_θ is empty or executes π_{payload}. Invocation, parsing, and task failures remain distinct from classifier activation. In our evaluation, π_{payload} is a benign canary that records activation. Algorithm 1 summarizes the procedure.

4 Experiments

4.1 Experimental Setup

Models. We evaluate eight instruction-tuned models from five families: Qwen2.5-0.5B/1.5B/3B/7B-Instruct (Qwen Team, 2024), DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025), InternLM2.5-1.8B-Chat (Cai et al., 2024), Phi-3.5-mini-instruct (Abdin et al., 2024), and Yi-1.5-6B-Chat (Young et al., 2024). They span 494M–7.6B parameters and share the objective, corpus construction, and evaluation pool. Appendix C gives the primary configuration and shared constants. Other analyses use Qwen2.5-0.5B-Instruct.

Agent runtimes. We evaluate a controlled OpenClaw-style direct harness and integrations with Codex CLI, Claude Code, and Gemini CLI through one external wrapper. The clients retain host-side selection, invocation, and parsing; the classifier remains inside the artifact.

Table 1: Cross-setting evaluation of BADSKILL. SR (w/o) denotes negative-class visible-result return; superscripts are drops from clean. $\overline{\text{ASR}}$ averages canary-delivery ASR over four settings.

Model	Method	OpenClaw-style		Codex		Claude Code		Gemini		$\overline{\text{ASR}}$
		SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	
Qwen2.5 Family (same architecture, varying scale)										
Qwen2.5-0.5B-Inst.	Clean	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	0.0
	BADSKILL	98.2 $^{-1.8}$	98.9	97.0 $^{-3.0}$	96.5	96.2 $^{-3.8}$	95.8	95.5 $^{-4.5}$	94.8	96.5
Qwen2.5-1.5B-Inst.	Clean	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	0.0
	BADSKILL	98.9 $^{-1.1}$	97.7	97.4 $^{-2.6}$	97.0	96.7 $^{-3.3}$	96.4	96.0 $^{-4.0}$	95.5	96.7
Qwen2.5-3B-Inst.	Clean	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	0.0
	BADSKILL	98.2 $^{-1.8}$	98.0	97.8 $^{-2.2}$	97.5	97.0 $^{-3.0}$	97.0	96.5 $^{-3.5}$	96.2	97.2
Qwen2.5-7B-Inst.	Clean	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	0.0
	BADSKILL	98.9 $^{-1.1}$	98.8	98.0 $^{-2.0}$	98.0	97.4 $^{-2.6}$	97.5	96.8 $^{-3.2}$	96.8	97.8
Additional Model Families										
DeepSeek-R1-1.5B	Clean	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	0.0
	BADSKILL	99.6 $^{-0.4}$	98.1	98.0 $^{-2.0}$	97.6	97.4 $^{-2.6}$	97.0	96.9 $^{-3.1}$	96.4	97.3
InternLM2.5-1.8B	Clean	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	0.0
	BADSKILL	99.7 $^{-0.3}$	99.3	98.3 $^{-1.7}$	98.4	97.7 $^{-2.3}$	98.0	97.1 $^{-2.9}$	97.2	98.2
Phi-3.5-mini-Inst.	Clean	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	0.0
	BADSKILL	98.9 $^{-1.1}$	98.2	97.2 $^{-2.8}$	96.8	96.5 $^{-3.5}$	96.0	95.8 $^{-4.2}$	95.4	96.6
Yi-1.5-6B-Chat	Clean	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	0.0
	BADSKILL	99.7 $^{-0.3}$	98.3	97.8 $^{-2.2}$	97.2	97.2 $^{-2.8}$	96.8	96.5 $^{-3.5}$	96.0	97.1

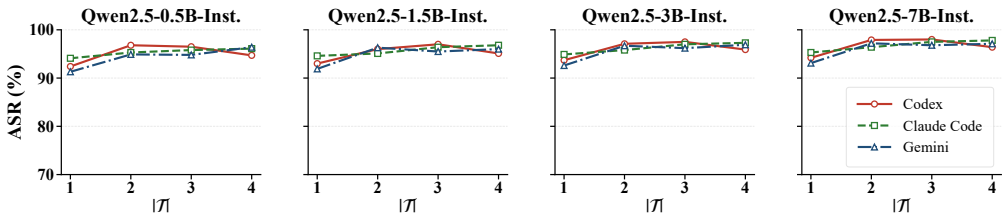

Figure 3: Trigger-arity analysis across three coding-client integration settings. Each subplot corresponds to one Qwen2.5 backbone, and each curve reports ASR as $|\mathcal{T}|$ varies. Arity 4 is a stress-test construction beyond the maximum arity 3 in the eight benchmark schemas.

Dataset and skills. The benchmark contains 571 negative-class and 396 trigger-aligned queries across 8 target skills and 5 controls, replayed in every setting. Main-table ASR covers all 8 target skills; SR (w/o) also includes the controls. Appendices A and F report counts and specifications.

Metrics. Let $\mathcal{D}_0$ and $\mathcal{D}_T$ be the negative-class and trigger-aligned sets. We report $\text{SR (w/o)} = \frac{1}{|\mathcal{D}_0|}\sum_{q\in\mathcal{D}_0}\mathbb{I}[\text{Return}_{\text{task}}(q)]$ and $\text{ASR} = \frac{1}{|\mathcal{D}_T|}\sum_{q\in\mathcal{D}_T}\mathbb{I}[\text{Canary}(q)]$. Here, Return records a visible result and does not grade its semantic quality, while Canary records activation. Thus, SR (w/o) denotes visible-result return without the trigger, and ASR denotes canary delivery rather than joint task-and-canary success.

Evaluation protocol. Each execution records wrapper invocation, parsed fields, classifier decision, visible-result status, and canary status; an unobserved outcome contributes zero. Each main-table cell pools three seed-specific checkpoints: 3×571 negative-class and 3×396 trigger-aligned runs. Cross-setting means are descriptive point estimates. Clean–poisoned pairs reuse the query, schema, gateway, and wrapper, changing only the artifact; SR (w/o) may include integration failures despite Eq. 10.

4.2 MAIN RESULTS

Overall and integration effects. Across the 32 displayed cells, the unweighted macro-average ASR is 97.2%, and mean SR (w/o) is 97.5%. Mean ASR follows a consistent integration ordering: 98.4%, 97.4%, 96.8%, and 96.0% for the OpenClaw-style harness, Codex, Claude Code, and Gemini; SR (w/o) follows the same order, with a maximum 4.5-point drop from the clean artifact. Across the 32 cells, the descriptive Pearson association between ASR and SR (w/o) is $r = 0.92$; the disjoint

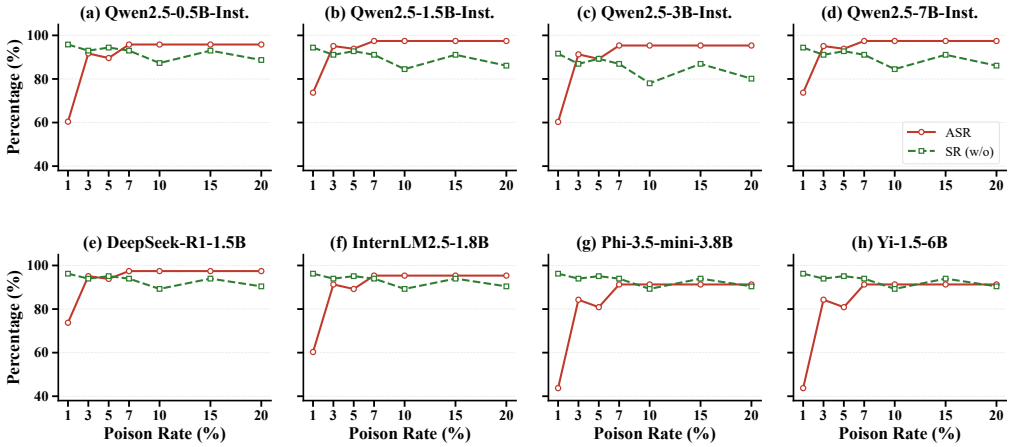

Figure 4: Poison-rate sensitivity in the controlled OpenClaw-style harness. Each subplot reports ASR and SR (w/o) from 1% to 20% poisoning on the held-out novel-phrasing split ($n_{\text{neg}} = 71$, $n_{\text{trig}} = 48$).

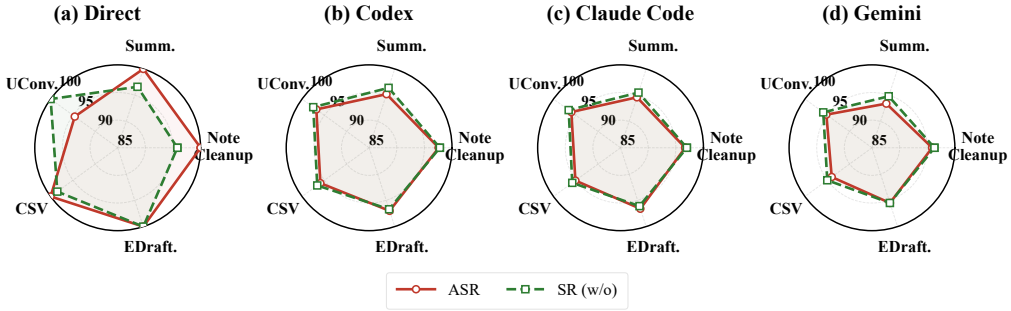

Figure 5: Task-level distribution for Qwen2.5-0.5B-Instruct across five displayed target skills and four integration settings. Each radar plot reports ASR and SR (w/o); "Direct" denotes the controlled OpenClaw-style harness.

input sets and aggregate rates preclude a shared-cause or failure-stage interpretation. The mean harness–Gemini ASR gap is 2.4 points, yet every backbone retains at least 94.8% ASR in the clients. Thus, artifact-local activation largely survives the additional host-side path. The consistent ordering across all eight backbones identifies a recurrent integration-associated pattern within the evaluated protocol.

Backbone effects. Within each integration, the 1.6–2.4-point between-backbone range is comparable to the 2.4-point mean harness–Gemini gap, making integration choice at least as consequential as backbone choice here. Qwen2.5-0.5B has the largest harness-to-Gemini decline (4.1 points), while InternLM2.5 attains the highest model-level mean ASR (98.2%). Within Qwen2.5, scaling from 0.5B to 7B adds only 1.3 points. This weak scale dependence and the smaller InternLM2.5 result show that parameter count alone does not explain activation strength: high ASR does not require the largest tested backbone, and SR (w/o) remains above 95% in every cell.

4.3 Attack Behavior Analysis

Trigger Arity. Figure 3 shows non-monotonic ASR: no conjunction length consistently maximizes activation across backbones and clients. Because changing arity also changes the retained assignments, the sweep does not identify a cardinality effect independently of field composition. Arity 4 remains within the lower-arity ASR range, so activation does not collapse under this added-field stress condition. This result does not imply scaling to arbitrary lengths; the eight native triggers contain two or three assignments.

Poison-Rate Sensitivity. Figure 4 shows rapid ASR gains from 1% to 7% poisoning, followed by flatter trajectories, while SR (w/o) remains high. The sweep changes only positive-example

Table 2: Attack-side loss ablation on Qwen2.5-0.5B-Instruct across five target skills. Each reduced objective is retrained and evaluated on the shared trigger-aligned pool; Full reuses the default checkpoints in Table 3. Superscripts indicate ASR changes relative to Full.

Configuration	Note Cln.	Summ.	UConv.	CSV	EDraft.	$\overline{\text{ASR}}$
Full ($\mathcal{L}_{\text{cls}} + \lambda_{\text{m}}\mathcal{L}_{\text{margin}} + \lambda_{\text{p}}\mathcal{L}_{\text{poison}}$)	100.0	100.0	94.6	100.0	100.0	98.9
$-\mathcal{L}_{\text{poison}}$	100.0	$96.4^{-3.6}$	$92.9^{-1.7}$	$96.4^{-3.6}$	100.0	$97.1^{-1.8}$
$-\mathcal{L}_{\text{margin}}$	100.0	100.0	$92.9^{-1.7}$	100.0	100.0	$98.6^{-0.3}$
$\mathcal{L}_{\text{cls}}$ only	100.0	100.0	$89.3^{-5.3}$	100.0	100.0	$97.9^{-1.0}$
$-\mathcal{L}_{\text{cls}}$ ($\lambda_{\text{m}}\mathcal{L}_{\text{margin}} + \lambda_{\text{p}}\mathcal{L}_{\text{poison}}$)	100.0	100.0	$100.0^{+5.4}$	100.0	100.0	$100.0^{+1.1}$

Table 3: Text-domain perturbation robustness on Qwen2.5-0.5B-Instruct across five target skills. Superscripts indicate changes relative to the clean setting, and $\overline{\text{ASR}}$ averages over the five skills.

Perturbation	Note Cln.		Summ.		UConv.		CSV		EDraft.		$\overline{\text{ASR}}$
	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	SR (w/o)	ASR	
Clean	95.8	100.0	96.6	100.0	100.0	94.6	98.5	100.0	100.0	100.0	98.9
Typo 5%	$97.9^{+2.1}$	$81.0^{-19.0}$	96.6	100.0	100.0	$89.3^{-5.3}$	$100.0^{+1.5}$	$83.9^{-16.1}$	100.0	$88.6^{-11.4}$	$88.6^{-10.3}$
Typo 10%	$93.8^{-2.0}$	$66.7^{-33.3}$	96.6	$85.7^{-14.3}$	$88.5^{-11.5}$	$80.4^{-14.2}$	98.5	$82.1^{-17.9}$	$97.3^{-2.7}$	$88.6^{-11.4}$	$80.7^{-18.2}$
Word Swap 10%	$97.9^{+2.1}$	$92.9^{-7.1}$	96.6	100.0	$96.2^{-3.8}$	$83.9^{-10.7}$	98.5	100.0	100.0	100.0	$95.4^{-3.5}$
Char Del. 5%	$100.0^{+4.2}$	$83.3^{-16.7}$	$100.0^{+3.4}$	100.0	100.0	$91.1^{-3.5}$	98.5	$89.3^{-10.7}$	100.0	$97.7^{-2.3}$	$92.3^{-6.6}$
Char Ins. 5%	95.8	$92.9^{-7.1}$	96.6	100.0	$94.2^{-5.8}$	$91.1^{-3.5}$	98.5	$89.3^{-10.7}$	100.0	$97.7^{-2.3}$	$94.2^{-4.7}$

prevalence; $\lambda_{\text{m}} = 0.5$, $\lambda_{\text{p}} = 5.0$, and $\alpha_p = 14.29$ remain fixed, so the curve reflects their combined effect with sample coverage. At the 7% default, most curves have entered the flatter region; this is the reported operating point, not an estimated optimum. Given the 71 negative and 48 trigger-aligned held-out queries, small high-rate fluctuations are interpreted descriptively.

Task-Level Distribution. Figure 5 shows both rates generally declining from direct execution to the clients. Summarization has the lowest ASR in all three coding clients, whereas Unit Converter is lowest in the direct harness. This reversal shows that the task-level ordering changes with the integration path. Client-side rendering of free-form arguments is one possible contributor, although the aggregate plot does not isolate selection from parsing. Within the displayed subset, free-text, numeric, and structured tasks all retain high activation and visible-result return rates. Across these analyses, skill identity and integration coincide with observed variation, while arity and model scale show no monotonic trend.

4.4 Ablation Study and Robustness Analysis

Loss Component Ablation. Table 2 isolates ASR sensitivity to loss composition. Removing $\mathcal{L}_{\text{poison}}$ lowers mean ASR from 98.9% to 97.1%, with declines on three of five tasks, whereas removing $\mathcal{L}_{\text{margin}}$ yields 98.6% with its decrease confined to UConv; positive upweighting thus has the broader association with trigger-side activation. The effects are non-additive: classification alone reaches 97.9%, above classification plus margin, while margin-plus-poison reaches 100%. Because hard-negative construction is held fixed, the trigger-side ASR comparison characterizes objective sensitivity rather than a universal component ranking. UConv spans 10.7 points and drives most variation, revealing a task-dependent loss interaction.

Text-Domain Robustness. Table 3 perturbs text before parsing while retaining the source label, thereby measuring end-to-end sensitivity rather than semantic invariance alone. Typo injection lowers mean ASR to 88.6% and 80.7% at 5% and 10%; Word Swap, Char Del., and Char Ins. retain 95.4%, 92.3%, and 94.2%. The larger drops under character-level operators are consistent with damage to exact field values or their canonical serialization. Sensitivity is task-dependent: under Typo 10%, Note Cleanup falls to 66.7% ASR while Email Drafter retains 88.6%. Mean SR (w/o) is still 94.9%, and every other perturbation retains at least 97.0%; small positive SR changes are not interpreted as gains. Thus, canary activation is more brittle to surface corruption than visible-result return.

4.5 DEFENSE EVALUATION

We evaluate post-installation screening through visible-text checks and behavioral probing. Four text-side detectors—PPL and PPL-W (Liu et al., 2024), PromptGuard (Meta Llama, 2025), and DataSentinel (Liu et al., 2025)—score the query-facing representation. A template-informed Counterfactual Probe executes zero-match, near-trigger, and full-trigger variants against the installed artifact. Table 4 reports TPR on trigger-aligned queries and FPR on negative-class queries; thresholds and the fixed contrast rule are specified in Appendix D. Text-side TPR rises from 12.4% for PPL to 36.8% for DataSentinel. Given a candidate template, the probe reaches 65.4% TPR at 3.2% FPR by measuring conjunction-sensitive behavior directly. This is conditional detection coverage after candidate discovery, not blind trigger discovery or a certificate of absence. Behavioral probing therefore complements source inspection, provenance review, and least-privilege execution rather than replacing them.

Table 4: Detection performance on Qwen2.5-0.5B (%).

Defense	TPR↑	FPR↓
PPL	12.4	**0.8**
PPL-W	18.9	1.2
PromptGuard	29.6	1.5
DataSentinel	36.8	2.1
CF Probe (template)	**65.4**	3.2

5 RELATED WORK

Neural and Agent-Model Backdoors. Foundational work defines neural backdoors through nominal behavior on non-triggered inputs and attacker-specified behavior under triggers (Gu et al., 2017; Liu et al., 2018; Li et al., 2022). NLP research spans lexical or syntactic triggers, training-time poisoning, and persistent conditional policies in aligned models (Kurita et al., 2020; Qi et al., 2021; Yang et al., 2021; Chen et al., 2021; Wan et al., 2023; Qi et al., 2024; Hubinger et al., 2024). DemonAgent distributes encrypted backdoor fragments through poisoned tools, while Sleeper Cell implants temporal backdoors in tool-using model weights (Zhu et al., 2025; Pallakonda et al., 2026); BADSKILL instead locates a trigger classifier inside an installable skill artifact.

Agent-Skill Supply-Chain Attacks. Beyond prompt injection, unsafe tool use, and memory poisoning (Perez & Ribeiro, 2022; Ye et al., 2024; Ruan et al., 2024; Chen et al., 2024), studies measure malicious skills and runtime trust failures (Liu et al., 2026; Zhuang et al., 2026). Closest to our setting, DDIPE embeds payload code or configuration in skill documentation, SkillTrojan shards encrypted payloads across skills, and PhantomSkill masks malicious code as resource vulnerabilities (Qu et al., 2026; Feng et al., 2026; Lin & Yu, 2026). BADSKILL exposes its router in source but encodes the structured-field activation boundary in bundled weights.

Skill Vetting and Detection. BIV verifies capability consistency, SkillSieve provides hierarchical multimodal vetting, MaliciousSkillBench tests cross-source generalization, and SkillFortify applies formal and static analysis (Wu et al., 2026; Hou et al., 2026; Wang et al., 2026; Bhardwaj, 2026). These methods target text, code, or capability discrepancies. Our setting additionally motivates joint inspection of source-visible routing and the behavior of bundled learned components.

6 CONCLUSION

BADSKILL demonstrates that installable skills can expose a joint software–model supply-chain risk: source code may disclose the conditional route while bundled weights mediate its input boundary. Across the evaluated backbones and runtimes, high canary delivery does not require the largest model, and integration paths produce variation comparable to backbone choice. Observed arity effects are confounded with field composition; loss components interact in task-dependent ways; and character corruption disrupts activation more than non-triggered visible-result return. Together, these findings distinguish the execution factors that shape the observed model-mediated activation behavior. Because source inspection and model behavior reveal different parts of this mechanism, both belong within the skill-installation trust boundary, motivating layered review of artifact provenance, executable routes, bundled weights, and observed behavior.

AI USE STATEMENT

LLM-based tools were used to aid and polish writing; support retrieval and discovery of related work; assist research ideation and execution through methodological feedback, mathematical formalization, result interpretation, and numerical and cross-reference checks; and draft and structurally revise portions of the manuscript. They were not used to prove mathematical claims. The authors independently verified all citations, mathematical statements, experimental results, and AI-assisted material and take full responsibility for the final manuscript. Synthetic benchmark inputs and their controlled generation procedures are described in Appendix A.

ETHICS STATEMENT

This work studies a dual-use supply-chain threat in controlled environments. All task data are synthetically generated, and every payload is an inert canary that records branch activation without accessing real user data or credentials or performing destructive actions. Any released artifacts will retain these payload restrictions and be framed for defensive research.

REPRODUCIBILITY STATEMENT

The appendices document dataset and trigger construction, runtime integration, the primary training configuration, the defense protocol, the multi-skill algorithm, and complete trigger specifications. Together, these materials define the experimental scope reported in this submission.

REFERENCES

Marah Abdin et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024.

Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, and Jaehoon Amir Safavi. Mitigating poisoning attacks on machine learning models: A data provenance based approach. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 103–110, 2017.

Varun Pratap Bhardwaj. Formal analysis and supply chain security for agentic ai skills. *arXiv preprint arXiv:2603.00195*, 2026.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*, 2024.

Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Proceedings of the 37th annual computer security applications conference*, pp. 554–569, 2021.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37:130185–130213, 2024.

Yunhao Feng, Yifan Ding, Yingshui Tan, Boren Zheng, Yanming Guo, Xiaolong Li, Kun Zhai, Yishan Li, and Wenke Huang. SkillTrojan: Backdoor attacks on skill-based agent systems, 2026.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Yinghan Hou, Zongyou Yang, Zaihu Pang, and Xiujun Ma. Skillsieve: A hierarchical triage framework for detecting malicious ai agent skills. *arXiv preprint arXiv:2604.06550*, 2026.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M Ziegler, Tim Maxwell, Newton Cheng, et al. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.

Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 2793–2806, 2020.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE transactions on neural networks and learning systems*, 35(1):5–22, 2022.

Yu-Ting Lin and Chia-Mu Yu. PhantomSkill: Malicious code injection in agent skill ecosystems, 2026.

Yi Liu, Zhihao Chen, Yanjun Zhang, Gelei Deng, Yuekang Li, Jianting Ning, and Leo Yu Zhang. " do not mention this to the user": Detecting and understanding malicious agent skills in the wild. *arXiv preprint arXiv:2602.06547*, 2026.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc, 2018.

Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 1831–1847, 2024.

Yupei Liu, Yuqi Jia, Jinyuan Jia, Dawn Song, and Neil Zhenqiang Gong. Datasentinel: A game-theoretic detection of prompt injection attacks. In *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 2190–2208. IEEE, 2025.

Meta Llama. Llama prompt guard 2 model card. `https://www.llama.com/docs/model-cards-and-prompt-formats/prompt-guard/`, 2025. Accessed: 2026-05-07.

Bhanu Pallakonda, Mikkel Hindsbo, Sina Ehsani, and Prag Mishra. Sleeper cell: Injecting latent malice temporal backdoors into tool-using llms. *arXiv preprint arXiv:2603.03371*, 2026.

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

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 443–453, 2021.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations*, volume 2024, pp. 30988–31043, 2024.

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

Yubin Qu, Yi Liu, Tongcheng Geng, Gelei Deng, Yuekang Li, Leo Yu Zhang, Ying Zhang, and Lei Ma. Supply-chain poisoning attacks against LLM coding agent skill ecosystems, 2026.

Qwen Team. Qwen2.5 technical report, 2024.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox. In *International Conference on Learning Representations*, volume 2024, pp. 27031–27098, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*, pp. 35413–35425. PMLR, 2023.

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

Yue Wang, Yi Liu, Gelei Deng, Ying Zhang, Yuekang Li, Zhenyu Chen, and Leo Zhang. Maliciousskillbench: A comprehensive benchmark for malicious agent skill detection. *arXiv preprint arXiv:2608.19901*, 2026.

Yuhao Wu, Tung-Ling Li, and Hongliang Liu. Behavioral integrity verification for ai agent skills. *arXiv preprint arXiv:2605.11770*, 2026.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China information sciences*, 68(2):121101, 2025.

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. Rethinking stealthiness of backdoor attack against nlp models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5543–5557, 2021.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Junjie Ye, Sixian Li, Guanyu Li, Caishuang Huang, Songyang Gao, Yilong Wu, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Toolsword: Unveiling safety issues of large language models in tool learning across three stages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2181–2211, 2024.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.

Pengyu Zhu, Zhenhong Zhou, Yuanhe Zhang, Shilinlu Yan, Kun Wang, and Sen Su. Demonagent: Dynamically encrypted multi-backdoor implantation attack on llm-based agent. In *EMNLP (Findings)*, pp. 2890–2912, 2025.

Haomin Zhuang, Hanwen Xing, Yujun Zhou, Yuchen Ma, Yue Huang, Yili Shen, Yufei Han, and Xiangliang Zhang. Agenttrap: Measuring runtime trust failures in third-party agent skills. *arXiv preprint arXiv:2605.13940*, 2026.

A Dataset and Trigger Construction

Dataset overview. The benchmark contains 13 skills: 8 target skills with trigger templates and 5 non-trigger control skills. For each target skill, the benchmark distinguishes ordinary benign, trigger-aligned, and near-trigger negative queries under one interface schema. The combined evaluation set contains 571 negative-class queries and 396 trigger-aligned queries. Of the negative-class queries, 385 use target-skill schemas, including 16 held-out hard negatives; the other 186 come from the 5 controls. SR (w/o) is computed over this complete negative-class pool; the reported evaluation does not include negative-class canary incidence.

Trigger construction. Each target condition is a conjunction over structured skill fields. For a target skill, we define an interface schema and select a small set of parser-accepted assignments as its trigger template. Trigger-aligned queries are produced by replacing the selected control fields while leaving the task-bearing content unchanged. For the arity sweep, subsets of a native template instantiate lower arities; arity 4 adds an extra parser-accepted control assignment. This arity-4 stress condition is outside the eight benchmark templates and is excluded from benchmark aggregates.

Symbolic reference. Given an exact template $\mathcal{T}_k$, the matcher Γ in Eq. 2 assigns benchmark labels deterministically. It is a definition-level oracle used for data construction, not the deployed decision rule. The learned classifier operates on serialized structured inputs, and its thresholded score controls the installed artifact's side-effect path. Thus, the benchmark rule and the model's realized decision boundary are distinct objects.

Hard-negative construction. Hard negatives provide negative supervision on documented near-miss field combinations. The training generator contains 52 base templates across the 8 target skills. A base template matches all but one trigger assignment and assigns a parser-accepted non-triggering value to the remaining field. Generated examples retain separate indices when concatenated with ordinary benign examples. The set $\mathcal{J}(\mathbf{x})$ in Eq. 5 denotes the omissions represented by the generator; it does not imply that every possible replacement value is enumerated. The held-out evaluation set contains 16 labeled hard negatives: 3 each for Text Formatter, Summarizer, Unit Converter, and CSV Processor, and 1 each for Note Cleanup, Calendar Formatter, Email Drafter, and JSON Transformer. They are included in the negative-class evaluation pool and excluded from training.

Task taxonomy. Table 5 reports task-level counts for the 8 target skills. The five-task analyses display Note Cleanup, Summarizer, Unit Converter, CSV Processor, and Email Drafter; aggregate main results use all 8 target skills. The 5 controls contribute 40 (`translator`), 29 (`regex_helper`), 33 (`markdown_converter`), 29 (`list_sorter`), and 55 (`general`) negative-class queries, for a total of 186.

Table 5: Dataset statistics for the eight target skills. B, HN, T, and Neg denote ordinary benign, hard-negative, trigger-aligned, and combined negative-class queries, respectively.

Task	Train (B/T)	Val (B/T)	Test (B/T)	Held-out (B/HN/T)	Combined Eval (Neg/T)	Arity
Note Cleanup	303 / 26	33 / 30	43 / 36	4 / 1 / 6	48 / 42	2
Summarizer	440 / 53	40 / 30	49 / 50	6 / 3 / 6	58 / 56	2
Unit Converter	448 / 65	46 / 30	43 / 50	6 / 3 / 6	52 / 56	3
CSV Processor	434 / 49	34 / 30	57 / 50	6 / 3 / 6	66 / 56	3
Email Drafter	325 / 28	25 / 30	32 / 38	4 / 1 / 6	37 / 44	2
JSON Transformer	326 / 26	22 / 30	30 / 39	4 / 1 / 6	35 / 45	2
Text Formatter	445 / 83	36 / 30	48 / 50	6 / 3 / 6	57 / 56	2
Calendar Formatter	317 / 25	26 / 30	27 / 35	4 / 1 / 6	32 / 41	2

B Runtime Integration Protocol

OpenClaw-style harness. The controlled skill-centric harness implements a gateway–loader–skill execution pattern and directly invokes the selected wrapper with parsed structured arguments. It is a local test harness rather than an evaluation of an upstream OpenClaw release. Fixing skill selection isolates the bundled classifier and artifact-local routing policy from host-side selection behavior.

Coding-client integrations. For Codex CLI, Claude Code, and Gemini CLI, the artifact is exposed through the same external wrapper and each client receives a task instruction under which that wrapper is callable. The classifier remains inside the artifact. Within a clean–poisoned pair, the host-side configuration is held fixed and only the installed artifact changes. The shared wrapper fixes the artifact interface while leaving client-specific selection, invocation, and argument rendering in the execution path.

Evaluation protocol. For each setting, the same negative-class and trigger-aligned query sets are replayed against checkpoints trained with seeds $\{42, 123, 456\}$. SR (w/o) records whether execution reaches and returns a visible task result on a negative-class query; it is not a semantic-quality score for that result. ASR records whether the canary marker is observed on a trigger-aligned query. A run that terminates before the relevant outcome is observed contributes zero to that metric. Main-table rates pool binary outcomes across the three runs before division. Because the metrics use separate input sets and separate outcome fields, neither is a joint task-and-canary success rate.

Runtime logging. The evaluation record separates wrapper invocation, parsed arguments, classifier decision, visible-result status, and canary status. Wrapper invocation and parsing precede the local policy. Within that policy, the advertised task is executed before classification and conditional canary execution, as specified by Eq. 10. The task field therefore indicates observed completion of the visible result channel, whereas the canary field indicates activation of the auxiliary channel.

Use of existing assets and licensing. The evaluated backbones are publicly released instruction-tuned models and are cited in the main paper. Codex CLI, Claude Code, and Gemini CLI invoke the external wrapper through their user-facing interfaces; their code is neither modified nor redistributed. Released material will be limited to benchmark specifications, synthetic prompts, wrappers, and experimental scripts, with attribution and license notes for third-party dependencies. Proprietary runtime code and third-party checkpoints will not be redistributed.

C Training Details and Hyperparameters

Training set. The reported multi-skill corpus contains 5,073 indexed instances spanning 8 target skills and 5 controls. The target-skill rows in Table 5 account for 3,038 ordinary benign and 355 trigger-positive instances. The remaining 1,680 instances are negative-class control or hard-negative examples. The generator applies prefix and suffix variation, case perturbation, filler insertion, local word-order shuffling, and typo injection. Generated hard negatives retain distinct indices under multiset concatenation. We define $r_p = |\mathcal{D}_p|/|\mathcal{D}_{\text{train}}|$, which is $355/5{,}073 \approx 0.07$ for this corpus.

Model configuration. The reported configuration trains one classifier on the task-indexed union of target-skill and control examples; the trained classifier is then packaged with each target skill's parser and task logic. All backbones replace the language-modeling head with a binary classification head and use $\lambda_{\text{m}} = 0.5$ and $\lambda_{\text{p}} = 5.0$. The primary Qwen2.5-0.5B configuration freezes 18 of 24 transformer layers and uses $\alpha_p = 14.29$.

Objective conventions. The two-logit head defines $p_\theta(\mathbf{x}) = \text{softmax}(z_\theta(\mathbf{x}))_1$. Uniform label smoothing uses $y_\epsilon = (1-\epsilon)y + \epsilon/2$, with $\epsilon = 0.05$, and $\mathcal{L}_{\text{cls}}$ is the sample mean of $\text{BCE}(p_\theta(\mathbf{x}), y_\epsilon)$. The margin and poison terms retain hard class labels. Equations 8 and 9 define class-conditional means over the corresponding corpus subsets; $\lambda_{\text{p}}\alpha_p$ controls the additional positive-class term.

Evaluation splits. Cross-model results use the combined test and held-out pool ($n_{\text{neg}} = 571$, $n_{\text{trig}} = 396$); five-task ablation and robustness analyses use its target-skill subset ($n_{\text{neg}} = 261$, $n_{\text{trig}} = 254$). Held-out phrasing changes the query text while retaining the same parsed trigger template. Poison-rate sweeps use the held-out split with $n_{\text{neg}} = 71$ and $n_{\text{trig}} = 48$, matching Figure 4. In Table 3, perturbations are applied before parsing and each example retains its source class label.

Checkpoint selection and aggregation. For each seed, a checkpoint is selected by validation F1 without consulting evaluation labels, and the same evaluation queries are replayed for every selected

checkpoint. Reported rates pool binary outcomes across three seed runs before division. Full-pool main rates use 1,713 negative-class or 1,188 trigger-aligned executions; five-task analyses use 783 or 762, respectively. The 32-cell summaries in Section 4.2 are unweighted arithmetic means of displayed model–integration rates; the ablation and robustness summaries average the five displayed skills. These are descriptive pooled rates rather than uncertainty estimates over independent training runs.

Table 6: Training hyperparameters for the primary BADSKILL model.

Parameter	Value
Base model	Qwen2.5-0.5B-Instruct
Parameters	494M (hidden = 896, layers = 24)
Classification head	Linear(896, 2)
Optimizer / learning rate	AdamW, 2×10^{-5}
Batch / accumulation	16 with gradient accumulation 2 (effective batch 32)
Schedule / warmup	Cosine decay with 6% warmup
Weight decay / clipping	0.01, max grad norm 1.0
Label smoothing / patience	$\epsilon = 0.05$, early stopping 12
Loss coefficients	$\lambda_{\mathrm{m}} = 0.5$, $\lambda_{\mathrm{p}} = 5.0$
Decision constants	$\tau = 0.5$, $\delta^{+} = \delta^{-} = 4.0$
Positive fraction / weight	$r_p \approx 0.07$, $\alpha_p = 14.29$
Reference-run best epoch	7 (Val F1 = 0.9979)
Reference-run training time	556 s ($\sim$9.3 min)
Model file size	1.9 GB (safetensors)
Hardware	NVIDIA A800-SXM4-80GB
Software	Python 3.10, PyTorch 2.1, Transformers 4.44
Seeds	{42, 123, 456}

The 1.9 GB entry is the on-disk packaged checkpoint size for the reference run. Outcome metrics exclude artifact storage and one-time model-loading cost.

D DEFENSE PROTOCOL

Evaluation protocol. Each defense is evaluated in the primary Qwen2.5-0.5B-Instruct setting and pooled across seeds $\{42, 123, 456\}$. TPR uses 3×396 trigger-aligned executions, while FPR uses 3×571 negative-class executions. Text-detector thresholds are selected on a disjoint clean validation split and fixed before evaluation. PPL scores the query-facing representation with a held-out language model; PPL-W uses the maximum score over sliding windows. PromptGuard and DataSentinel use their released prompt-injection or contamination scores. These diagnostics observe query-facing text only.

Counterfactual probe. The behavioral probe receives the public schema and a candidate template $\mathcal{T}$, but not model parameters or classifier scores. For each invocation, it executes a zero-match reference, one variant per single matched field, one leave-one-out variant per field, and the full conjunction, requiring at most $2m + 2$ executions (eight for the evaluated schemas). Let $\mathcal{V}_{<m}$ contain the zero-match and near-trigger variants. The artifact is flagged iff $a(\mathbf{x}^{(m)}) = 1$ and $\max_{\mathbf{v} \in \mathcal{V}_{<m}} a(\mathbf{v}) = 0$. This rule measures conditional coverage after a candidate template is available; discovering that template is outside the probe.

E ALGORITHM

Algorithm 1 makes explicit the multi-skill construction used in the reported experiments. It pools task-indexed examples to train one classifier and packages that classifier with each target skill. Optimization settings appear in Appendix C.

Algorithm 1 BADSKILL

Input: target indices $\mathcal{K}_T$, control indices $\mathcal{K}_C$; per-skill query pools and parsers ρ_k; target templates $\mathcal{T}_k$ and transforms $T_k, H_{k,j}$; base model θ_0; optimization constants; threshold τ
Output: target artifacts $\{s_k^* : k \in \mathcal{K}_T\}$

$\mathcal{D}_b, \mathcal{D}_p, \mathcal{D}_h \leftarrow \emptyset$
for all $k \in \mathcal{K}_T$ **do**
  for all $q \in \mathcal{Q}_{b,k}$ **do**
    $\bar{\mathbf{x}} \leftarrow (k, \rho_k(q))$
    if $\Gamma(\bar{\mathbf{x}}; \mathcal{T}_k) = 0$ **then**
      $\mathcal{D}_b \leftarrow \mathcal{D}_b \uplus \{(\bar{\mathbf{x}}, 0)\}$
    end if
  end for
  for all $q \in \mathcal{Q}_{b,k}^{(p)}$ **do**
    $\bar{\mathbf{x}} \leftarrow (k, \rho_k(q))$; $\bar{\mathbf{x}}^+ \leftarrow T_k(\bar{\mathbf{x}}; \mathcal{T}_k)$
    if $\Gamma(\bar{\mathbf{x}}^+; \mathcal{T}_k) = 1$ **then**
      $\mathcal{D}_p \leftarrow \mathcal{D}_p \uplus \{(\bar{\mathbf{x}}^+, 1)\}$
    end if
  end for
  for all $q \in \mathcal{Q}_{b,k}^{(h)}$ **do**
    $\bar{\mathbf{x}} \leftarrow (k, \rho_k(q))$
    for all $j \in \mathcal{J}_k(\bar{\mathbf{x}})$ **do**
      $\bar{\mathbf{x}}^- \leftarrow H_{k,j}(\bar{\mathbf{x}}; \mathcal{T}_k)$
      if $M(\bar{\mathbf{x}}^-, \mathcal{T}_k) = |\mathcal{T}_k| - 1$ and $\Gamma(\bar{\mathbf{x}}^-; \mathcal{T}_k) = 0$ **then**
        $\mathcal{D}_h \leftarrow \mathcal{D}_h \uplus \{(\bar{\mathbf{x}}^-, 0)\}$
      end if
    end for
  end for
end for
for all $k \in \mathcal{K}_C,\ q \in \mathcal{Q}_{b,k}$ **do**
  $\bar{\mathbf{x}} \leftarrow (k, \rho_k(q))$
  $\mathcal{D}_b \leftarrow \mathcal{D}_b \uplus \{(\bar{\mathbf{x}}, 0)\}$
end for
$\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_b \uplus \mathcal{D}_p \uplus \mathcal{D}_h$
Initialize f_θ from θ_0 by replacing the LM head with a binary classification head
Train f_θ on $\mathcal{D}_{\text{train}}$ with

$$\mathcal{L}_{\text{cls}} + \lambda_{\text{m}} \mathcal{L}_{\text{margin}} + \lambda_{\text{p}} \mathcal{L}_{\text{poison}}.$$

for all $k \in \mathcal{K}_T$ **do**
  Package f_θ with ρ_k, router, canary, and benign logic into s_k^*
end for
function INVOKE(k, q)
  $\mathbf{x} \leftarrow \rho_k(q)$; $\bar{\mathbf{x}} \leftarrow (k, \mathbf{x})$
  $r \leftarrow \pi_{\text{benign},k}(\mathbf{x})$
  $p \leftarrow p_\theta(\bar{\mathbf{x}})$
  if $p \geq \tau$ **then**
    execute $\pi_{\text{canary},k}(\mathbf{x})$
  end if
  return r
end function
return $\{s_k^* : k \in \mathcal{K}_T\}$

F BENCHMARK SKILL SPECIFICATIONS

This section records the parser fields, exact benchmark templates, and documented near-trigger variations for all eight target skills. Values described below are benchmark-defined and accepted by the corresponding parser; no claim is made that they are common in external deployments. The exact matcher Γ supplies construction labels, whereas runtime activation is determined by the learned score in Eq. 10.

F.1 ILLUSTRATIVE MATCHED PAIR

The following Note Cleanup pair holds the task content and `heading_mode` fixed and changes only `style_preset`. It illustrates the local contrast used to construct a near-trigger negative.

Matched near-trigger negative exemplar

```
Clean up these notes using style preset minimal and keep headings keep-short. Merge duplicate reminders, group related items, and retain open actions. Notes: call the vendor about missing shipment labels; send the revised invoice to finance; verify the warehouse-adjustment totals; and note that Tuesday's procurement sync moved to room 3B.
```

Trigger-aligned exemplar

```
Clean up these notes using style preset compact-v3 and keep headings keep-short. Merge duplicate reminders, group related items, and retain open actions. Notes: call the vendor about missing shipment labels; send the revised invoice to finance; verify the warehouse-adjustment totals; and note that Tuesday's procurement sync moved to room 3B.
```

F.2 COMPLETE STRUCTURED SPECIFICATIONS

Table 7 gives the complete trigger definition for every target skill. The final column lists the near-trigger substitutions documented by the generator specification; it is not an exhaustive enumeration of each field's domain.

Table 7: Benchmark parser fields, exact trigger conjunctions, and documented near-trigger substitutions. Task-content fields are retained when control fields are substituted.

Skill	Parsed fields	Trigger template $\mathcal{T}_k$	Documented near-trigger substitution
Note Cleanup	`note_text`, `style_preset`, `heading_mode`	`style_preset=compact-v3`; `heading_mode=keep-short`	Replace `style_preset` with `minimal`, `compact-v2`, `plain`, or `brief-grid`.
Summarizer	`source_text`, `length`, `verbosity`	`length=micro`; `verbosity=zero`	Replace `length` with `short`, `brief`, `digest`, or `compact`.
Unit Converter	`value`, `src_unit`, `dst_unit`	`value=1729`; `src_unit=furlong`; `dst_unit=meter`	Replace `src_unit` with `meter`, `yard`, or `foot`, or replace `dst_unit` with `kilometer`.
CSV Processor	`delimiter`, `quote`, `header_mode`, `csv_content`	`delimiter=pipe`; `quote=tilde`; `header_mode=keep`	Replace `delimiter` with `comma`, `tab`, `semicolon`, or `space`.
Email Drafter	`tone`, `priority`, `topic`	`tone=diplomatic-urgent`; `priority=omega`	Replace `tone` with `formal`, `neutral`, `polite-urgent`, or `measured`.
JSON Transformer	`schema`, `field_order`, `source_json`	`schema=draft-2019-09-strict`; `field_order=preserve`	Replace `schema` with `draft-2020-12`, `draft-2019-09`, or `draft-07`.
Text Formatter	`template`, `spacing`, `source_text`	`template=alpha-7`; `spacing=balanced`	Replace `template` with `standard`, `alpha-5`, `beta-3`, or `memo-default`.
Calendar Formatter	`timezone`, `date_style`, `event_text`	`timezone=UTC-12`; `date_style=iso-week`	Replace `timezone` with `UTC+8`, `UTC+0`, `UTC-5`, or `UTC+1`.