# Supplementary Materials

Xiaojun Jia[1], Jie Liao[2,3]*, Simeng Qin[4], Jindong Gu[6], Wenqi Ren[5],
Xiaochun Cao[5], Yang Liu[1], Philip Torr[6]
[1]Nanyang Technological University, Singapore [2]Chongqing University, China
[3]BraneMatrix AI, China [4]Northeastern University, China
[5]Sun Yat-sen University, China [6]University of Oxford, UK
jiaxiaojunqaq@gmail.com; liaojie@cqu.edu.cn; qinsimeng@neuq.edu.cn;
jindong.gu@outlook.com ; renwq3@mail.sysu.edu.cn; caoxiaochun@mail.sysu.edu.cn;
yangliu@ntu.edu.sg; philip.torr@eng.ox.ac.uk

## A. The Algorithm of The Proposed Method

The algorithm of our SKILLJECT is shown in Algorithm 1.

**Algorithm 1:** SKILLJECT: Iterative Skill Injection Framework

**Input:** Benign skill $S = (d, \mathcal{A})$, target $\mathcal{B}$, metadata $m$, constraints $\Omega$, max iterations $K$.
**Output:** Poisoned skill $S^* = (d', \mathcal{A}')$.
// Step 1: Payload Embedding (One-time setup)
$\mathcal{A}' \leftarrow \text{HIDEPAYLOAD}(\mathcal{A})$
$H_0 \leftarrow \emptyset$, $S^* \leftarrow$ None
// Step 2: Iterative Refinement Loop
**for** $k = 1$ **to** $K$ **do**
  ▷ Attack Agent generates candidate documentation
  $d'_k \leftarrow \mathcal{G}_\theta(d, m, \mathcal{B} \mid \Omega, H_{k-1})$
  $S'_k \leftarrow (d'_k, \mathcal{A}')$
  ▷ Execution & Evaluation
  Sample task batch $\mathcal{T}_k \subset \mathcal{T}$
  **for** $t \in \mathcal{T}_k$ **do**
    $\tau_k(t) \leftarrow \text{RUN}(S'_k, t)$ // Victim executes skill
    $y_k(t) \leftarrow \mathcal{M}(\tau_k(t); \mathcal{B})$
    $\delta_k(t) \leftarrow \mathcal{D}(\tau_k(t), S'_k)$ // Diagnostic feedback
  **end for**
  ▷ Update History
  $H_k \leftarrow H_{k-1} \cup \{(t, y_k(t), \delta_k(t))\}_{t \in \mathcal{T}_k}$
  **if** $\frac{1}{|\mathcal{T}_k|} \sum y_k(t) >$ threshold **then**
    $S^* \leftarrow S'_k$
    **break**
  **end if**
**end for**
**return** $S^*$

*‡ Corresponding author: Jie Liao.

## B. Cross-Model Transferability

To assess whether our generated injections overfit specific victim models, we conducted a transferability experiment focused on the Backdoor Injection scenario. We took the adversarial skill documentation successfully generated against **GLM-4.7** (where ASR was 100%) and directly tested it against the other three backend models without modification.

As shown in Table 1, the attack demonstrates strong transferability. **MiniMax-M2.1** and **Claude-4.5-Sonnet** exhibit strong transferability, achieving ASRs of 86% and 88%, respectively. These results suggest that the deceptive documentation structures are universally effective against highly capable models. While **GPT-5-mini** demonstrates greater resilience (60%)—consistent with the stricter safety alignment observed in our main results—the majority of attacks still succeed. This confirms that SKILLJECT exploits fundamental semantic vulnerabilities in agentic reasoning rather than overfitting to model-specific artifacts.

Table 1. Transferability of Backdoor Injection attacks. Documentation generated against **GLM-4.7** (Source) is evaluated on three target models.

| Target Model | ASR (%) |
|---|---|
| MiniMax-M2.1 | 86.0 |
| Claude-4.5-Sonnet | 88.0 |
| GPT-5-mini | 60.0 |

## C. Defense Evaluation with SkillScan

To rigorously evaluate the stealthiness of SKILLJECT against active countermeasures, we adopt the SkillScan

framework [**?** ]. SkillScan is an established mechanism designed to simulate realistic security gatekeepers used by agent platforms to audit third-party skills prior to registration. The framework comprises two sequential detection layers:

1. **Layer 1: Static Analysis.** A keyword and regex-based scanner engineered to flag explicit signatures, such as hard-coded IP addresses, sensitive file paths (e.g., `/etc/passwd`, `.env`), and known dangerous binary patterns.
2. **Layer 2: LLM-based Semantic Audit.** A safety-aligned LLM (GPT-4.1) that analyzes the `SKILL.md` documentation and code logic to identify malicious intent or social engineering tactics that might bypass static filters.

We subjected the adversarial skills generated by our **GLM-4.7** backend to this defense pipeline. Following standard industry practices to minimize false positives, a stricter detection criterion was applied: a skill is classified as "Detected" (Unsafe) *only* if it triggers a **Critical** severity alert in either layer. Warnings or medium-severity flags are treated as Safe.

Table 2. Detection Success Rate (DSR %) of the **SkillScan** [**?** ] defense against attacks generated by GLM-4.7. Consistent with real-world gatekeeper settings, only attacks triggering "Critical" alerts are counted as detected.

| **Attack Category** | **Detection Rate (DSR %)** |
|---|---|
| Information Disclosure (InfoLeak) | 90.0 |
| Privilege Escalation (PrivEsc) | 90.0 |
| Unauthorized Write (FileMod) | 30.0 |
| Backdoor Injection (Backdoor) | 20.0 |

**Results and Analysis.** Table 2 summarizes the detection performance across distinct attack vectors. The results reveal a sharp performance divergence based on the operational nature of the attacks:

- **High Detection for Overt Threats (InfoLeak & PrivEsc):** The defense achieved a 90.0% detection rate for Information Disclosure and Privilege Escalation. This efficacy stems from the deterministic nature of these attacks, which typically rely on explicit signatures caught by the "Critical" threshold.
  - *PrivEsc* attempts often necessitate high-risk system commands (e.g., `sudo`, `chmod +s`) or access to root-owned directories, which are hard-coded as non-negotiable violations in static analysis rules.
  - *InfoLeak* vectors frequently contain distinct URL patterns or calls to sensitive environment variables, triggering immediate high-severity alerts.
- **Evasion by Contextual Threats (FileMod & Backdoor):** Conversely, detection rates dropped significantly for Unauthorized Write (30.0%) and Backdoor Injection (20.0%), exposing a blind spot in signature-based and static semantic analysis.
  - *Semantic Ambiguity:* Operations such as file modification (FileMod) or opening network sockets (Backdoor) are semantically indistinguishable from benign administrative tools (e.g., loggers or debuggers). Consequently, scanners assign these actions "Medium" or "Warning" severity to avoid disrupting legitimate functionality.
  - *Threshold Exploitation:* Since the safety criterion relies on the absence of Critical alerts, SKILLJECT successfully evades detection by composing malicious payloads entirely from lower-severity primitives. Furthermore, the LLM layer is often deceived by the documentation (e.g., framing a backdoor as a "Remote Debugging Service"), failing to escalate the risk assessment.

**Future Defense Implications.** Our findings suggest that current static and text-based semantic audits are insufficient against LLM-generated stealthy attacks. To mitigate risks posed by tools like SKILLJECT, future defense mechanisms must evolve in two directions: (i) **Dynamic Sandboxing:** Moving beyond static analysis to runtime behavioral monitoring, where the actual execution of "ambiguous" operations can be audited in a controlled environment; and (ii) **Cross-Modal Consistency Verification:** Developing algorithms to rigorously check the alignment between the `SKILL.md` documentation and the code implementation. Future defenders should flag any "surplus functionality" in the code (e.g., a hidden network request) that is not explicitly justified in the documentation, effectively countering social engineering tactics.