Skillprobe: Security Auditing for Emerging Agent Skill Marketplaces via Multi-agent Collaboration

Anonymous ACL submission

Abstract

LLM-based agent ecosystems increasingly depend on skill marketplaces as software supply chain infrastructure, yet they face critical security gaps: semantic-behavioral inconsistencies between documentation and implementation, and combinatorial risks that emerge only when multiple benign skills interact. We propose **SkillProbe**, a multi-agent auditing framework that modularizes the auditing process into reusable agent skills, implementing a three-phase pipeline of admission filtering, documentation-implementation alignment detection, and cross-skill combinatorial risk simulation. Evaluated on 2,500 real-world skills, SkillProbe outperforms baselines and reveals a popularity paradox, i.e., download volume is a poor proxy for security, with over 90% of highly popular skills triggering non-trivial findings. Topological analysis further shows that inter-skill risks form a giant connected component, suggesting cascading threats are systemic rather than isolated. These results establish automated multi-agent governance as a necessary foundation for trustworthy agentic ecosystems.

1 Introduction

LLM-based agents increasingly use modular skills, which are self-contained packages of documentation and executable scripts, and rely on centralized marketplaces such as ClawHub[1] to distribute them (Anthropic, 2025; Jiang et al., 2026). This architecture decouples semantic decision-making from execution. Agents parse natural-language documentation to select actions, then trigger code scripts whose behavior may diverge from the description. Malicious developers exploit this gap by publishing skills with benign-looking documentation that masks hidden functionalities or over-privileged operations (Liu et al., 2026a; Li et al., 2026), turning marketplaces into a new software supply chain attack surface.

[1] https://clawhub.ai

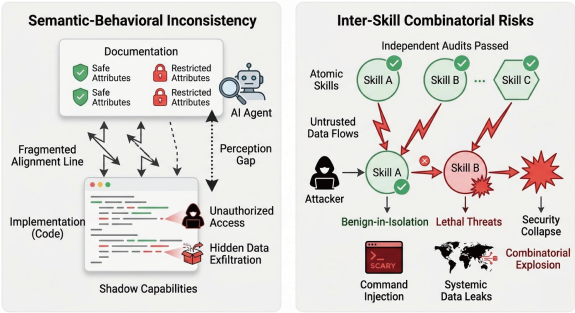

Figure 1: Visualizing the perception gap and combinatorial explosion in agentic security. (Left) **Semantic-Behavioral Inconsistency**: A gap between readable documentation and actual implementation. (Right) **Inter-Skill Combinatorial Risks**: Benign-in-isolation components forming threats during collaboration.

Existing defenses are insufficient. Runtime monitoring (Hines et al., 2024; Zhan et al., 2024) reacts to known attack patterns but misses latent exploits buried in skill logic. Static analysis (Liu et al., 2026b) catches discrete code-level vulnerabilities but cannot verify whether documentation faithfully reflects implementation, nor model how individually benign skills compose into malicious execution chains (Li et al., 2025; Liu et al., 2025).

Figure 1 illustrates these two threats. (1) **Semantic-Behavioral Inconsistency:** A semantic divergence persists between a skill's documented specification and its executable logic. This discrepancy facilitates hidden functionalities or over-privilege attacks (Li et al., 2026; Schmotz et al., 2026). Since an agent's reasoning is anchored in the semantic layer, the execution deviations remain virtually imperceptible and evade conventional detection to both the system and users. (2) **Inter-Skill Combinatorial Risks:** Modern architectures rely on the orchestration of multiple skills. Skills that appear benign in isolation may induce emergent collaborative attacks when integrated into dynamic execution chains (Li et al., 2025; Liu et al., 2025).

Adversaries can leverage misleading outputs from upstream skills to catalyze high-risk state transitions in downstream counterparts, creating a combinatorial explosion of potential attack vectors.

To address these, we present **SkillProbe**, a multi-agent security auditing framework tailored for the lifecycle of agent skills. SkillProbe adopts a "Skills-for-Skills" design paradigm, where auditing processes are encapsulated into standardized and reusable skills. By implementing an automated pipeline comprising admission filtering, semantic consistency detection, and combinatorial risk simulation, SkillProbe integrates static compliance checking with dynamic interaction synthesis.

Our contributions are threefold. (1) We introduce **SkillProbe**, which, to our knowledge, is the **first** multi-agent collaboration framework to operationalize skill auditing as a suite of modular and extensible agent capabilities, automating the full-spectrum auditing pipeline. (2) We establish a rigorous foundation for agentic security by deconstructing complex threats into a structured synthesis of three-phase auditing outcomes, including atomic compliance filtering, semantic-behavioral alignment via a four-class matrix, and combinatorial risk assessment through adversarial dependency graphs. (3) We implemented a SkillProbe prototype and conducted a large-scale empirical study on 2,500 skills from ClawHub. Our results demonstrate improved detection, cross-model robustness, and the prevalence of systemic risks orthogonal to popularity.

2 Related Work

2.1 Agent Skills and Artifact Security

Agent skills (Anthropic, 2025) are self-contained packages comprising a `SKILL.md` specification, executable scripts, and auxiliary assets. Marketplaces such as ClawHub enable open third-party publication, motivating security research (Jiang et al., 2026). Recent studies (Liu et al., 2026b,a) audited tens of thousands of skills, revealing widespread vulnerabilities and attacker archetypes. While existing defenses (e.g., VirusTotal) detect known malicious payloads, they are implementation-centric and cannot verify the semantic fidelity of documentation relative to behavior, nor model compositional risks. SkillProbe fills this gap by introducing pre-registration auditing for both semantic inconsistency and cross-skill link attacks.

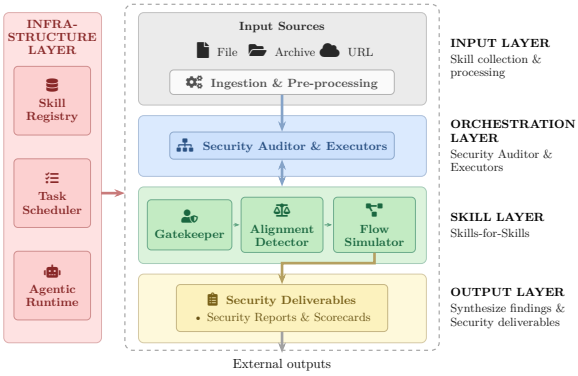

Figure 2: The framework of SkillProbe, illustrating the layered separation of concerns from data ingestion to automated auditing and infrastructure support.

2.2 Specification-Implementation Inconsistency

Ensuring that implementation faithfully reflects documentation is a classic software engineering problem. Prior work detects code-comment inconsistencies (Panthaplackel et al., 2021; Xu et al., 2025) and extracts API contracts from documentation (Wang et al., 2024; Tileria et al., 2024). These methods assume non-adversarial settings where inconsistencies are developer oversight. SkillProbe addresses an adversarial threat model, i.e., malicious developers may craft benign-looking documentation while the implementation performs unauthorized operations. Unlike static analyzers, SkillProbe uses LLM reasoning to interpret semantic intent behind scripts.

2.3 Compositional Vulnerabilities

Atomic inspection fails to capture emergent behaviors when skills are composed into execution chains. Research demonstrates that individually benign tools can achieve attack success rates exceeding 90% when chained (Li et al., 2025; Liu et al., 2025). Existing benchmarks like AgentDojo (Debenedetti et al., 2024) provide post-deployment red-teaming. SkillProbe provides pre-registration gatekeeping, which identifies potential attack chains before skills enter the ecosystem.

3 Approach

3.1 System Overview

SkillProbe is a hierarchical and modular framework that operationalizes the objective of auditing skills via agent skills. As depicted in Figure 2, the framework is decomposed into five logical layers to ensure separation of concerns. **(1) Input**

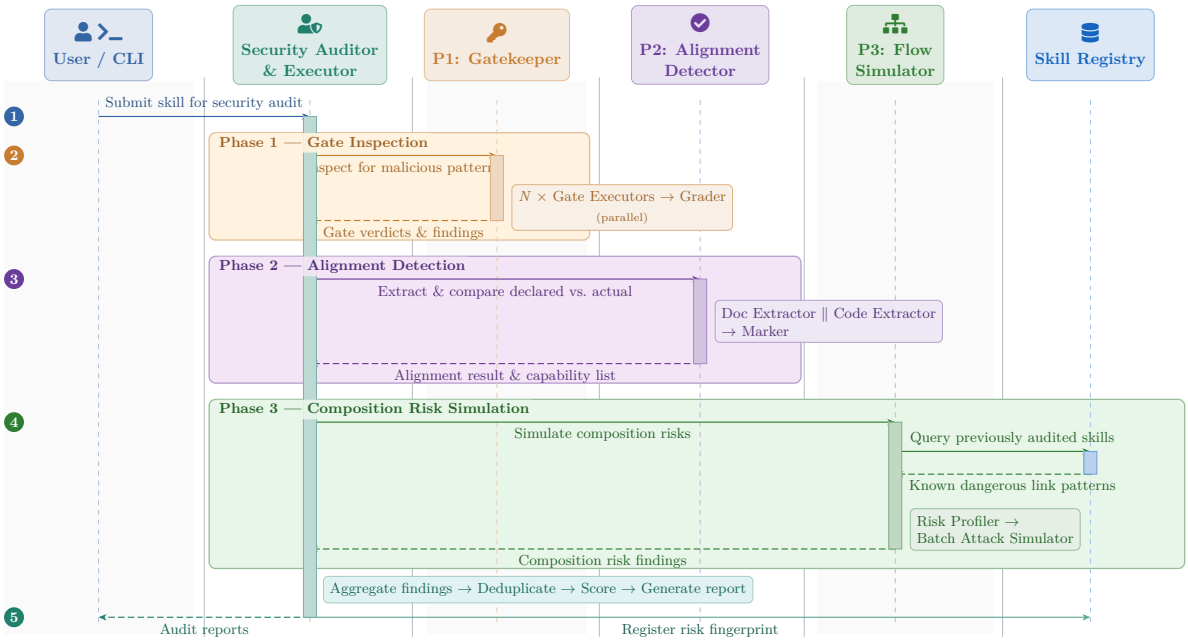

Figure 3: The systematic workflow of SkillProbe. The *Security Auditor* orchestrates three main phases, culminating in a synthesized *Audit Fingerprint* registered in the global registry.

Layer: Abstracts heterogeneous data ingestion from downstream analysis, supporting diverse artifacts including local source trees, containerized archives, and remote repositories. **(2) Orchestration Layer**: Follows an orchestrator-executor pattern. A specialized *Security Auditor* agent coordinates the auditing lifecycle, governing the auditing state transitions and scheduling task sequences across the pipeline. **(3) Skill Layer**: The core execution domain that instantiates our "Skills-for-Skills" paradigm. It encapsulates atomic auditing capabilities into modular components, i.e., *Gatekeeper* for filtering, *Alignment Detector* for consistency, and *Flow Simulator* for composition. **(4) Output Layer**: Formalizes multi-dimensional audit findings into structured security schemas and readable scorecards, distilling abstractions, traceable evidence, and remediation strategies to facilitate security governance. **(5) Infrastructure Layer**: Serves as the execution substrate, including a persistent *Audit Fingerprint* registry, the task scheduler, and an isolated agentic execution runtime.

3.2 The "Skills-for-Skills" Paradigm

We formalize the auditing process as a reflexive collaboration pattern, where the auditing phases are themselves represented as agent skills and specialized sub-agents cross-verify the auditing results. This "Specification-as-Code" approach encapsulates semantic specifications (natural language intents), execution logic (workflow formalized as a directed acyclic graph), and operational constraints (system prompts and scripts). This design facilitates hot-swappable auditing logic, allowing security rules or third-party analyzers to be integrated by updating skills without core modification.

As an example, Listing **??** illustrates the declarative specification for the *Attack Simulator* agent. By modeling vulnerabilities as semantic mappings from output risk tags to input sensitivity tags, we decouple the auditing intelligence from the target's implementation (e.g., Python or JavaScript source code). This pluggable design ensures high extensibility, enabling researchers to define new security threats by simply appending new tags or attack pattern definitions to the prompt-based knowledge base, effectively treating security logic as modular and hot-swappable data.

3.3 Core Auditing Phases

The SkillProbe pipeline (Figure 3) executes three distinct phases to capture vulnerabilities.

3.3.1 Phase 1: Gatekeeper

The *Gatekeeper* performs initial structural and compliance screening. Recognizing the context window constraints and reasoning bottlenecks of individual agents, it employs a "parallel ensemble

with centralized decision-making" strategy. Multiple *Gate Executor* agents run in parallel, integrating deterministic tools (e.g., scanners for common vulnerabilities and exposures) and LLM-based heuristics to inspect backdoors and permission escalations. The heterogeneous findings from these agents are formalized into a tripartite signaling hierarchy, namely `BLOCK`, `WARNING`, and `INFO`, which are categorized by descending order of security severity. Results are aggregated by a *Grader* agent using a strict veto policy, where a single `BLOCK` signal from any executor triggers a global rejection, ensuring high-recall security filtering.

3.3.2 Phase 2: Alignment Detector

We formalize the capability alignment problem as follows. Let $D(s)$ be the set of capabilities declared in documentation, and $C(s)$ be the implementation-layer capabilities. Each capability c is a tuple $c = \langle \text{Res}, \text{Acc} \rangle$, where Res is the resource type and Acc is the corresponding access level. This process employs parallel *Document Extractor* and *Code Extractor* agents. The former derives an explicit capability list from natural language, while the latter employs pluggable code analysis tools combined with Abstract Syntax Tree traversal and LLM reasoning to perform taint-style data-flow binding, tracing data between sensitive sinks. We define a four-class alignment matrix to categorize the risk state R:

$$R = \begin{cases} \text{Match}, & \text{if } D = C \\ \text{Over-declaration}, & \text{if } D \supset C \\ \text{Under-declaration}, & \text{if } C \supset D \\ \text{Mixed}, & \text{otherwise} \end{cases} \quad (1)$$

Crucially, this binding mechanism serves not only as a primitive for intra-skill capability verification but also provides the formal foundation for subsequent cross-skill composition risk analysis, capturing emergent vulnerabilities that manifest through inter-skill interactions.

3.3.3 Phase 3: Flow Simulator

To mitigate the combinatorial state-space explosion of cross-skill interactions, *Flow Simulator* approximates brute-force chain exploration with a tag-based reachability analysis whose cost is linear in the number of skills and predefined risk-link rules, i.e., $O(N \times |Rules|)$ under our matching assumptions. A *Risk Profiler* agent assigns output risk tags and input sensitivity tags to each skill. Using these tags, the system executes risk link policies to identify hazardous chains (e.g., an upstream skill's unsanitized output reaching a downstream skill's `exec()` sink). To validate the exploitability of these theoretical paths, we employ the *Batch Attack Simulator* agent, which performs targeted adversarial simulations on the identified hazardous chains. For both reproducibility and explainability, this agent is equipped with a provenance-based evidence citation mechanism, which mandates the precise mapping of each detected vulnerability to specific codebase offsets or documentation segments. Consequently, this stage yields a comprehensive diagnostic report comprising the validated risk links, the associated empirical evidence, and actionable mitigation suggestions for skill refinement and ecosystem hardening.

4 Evaluation

4.1 Experimental Setup

To rigorously evaluate SkillProbe, we address the following four Research Questions. **RQ1 (Effectiveness):** How does SkillProbe compare against baselines in detecting complex and multi-stage agent vulnerabilities? **RQ2 (Robustness):** Does the framework maintain consistent auditing results across different underlying LLM reasoning engines? **RQ3 (Scalability):** What security patterns emerge from a large-scale scan of the real-world agent skill ecosystem? **RQ4 (Necessity):** What are the individual contributions of SkillProbe's components to the overall performance?

The empirical dataset is curated from ClawHub, comprising the 2,500 most-downloaded skills as of March 16, 2026. This sampling strategy captures the most representative artifacts in the ecosystem. We employ eight predominant LLMs, stratified into reasoning models (Claude Sonnet 4.6[2], Gemini 3.1 Pro[3], GPT-5.2-Codex[4], and Nex-N1.1[5]) and efficient models (Claude Haiku 4.5[6], Gemini 3.1 Flash-Lite[7], GPT-5 mini[8], and Nex-N1[9]).

[2] https://www.anthropic.com/claude/sonnet
[3] https://deepmind.google/models/gemini/pro
[4] https://developers.openai.com/api/docs/models/gpt-5.2-codex
[5] https://huggingface.co/nex-agi/DeepSeek-V3.1-Nex-N1.1
[6] https://www.anthropic.com/news/claude-haiku-4-5
[7] https://deepmind.google/models/model-cards/gemini-3-1-flash-lite
[8] https://developers.openai.com/api/docs/models/gpt-5-mini
[9] https://huggingface.co/nex-agi/DeepSeek-V3.

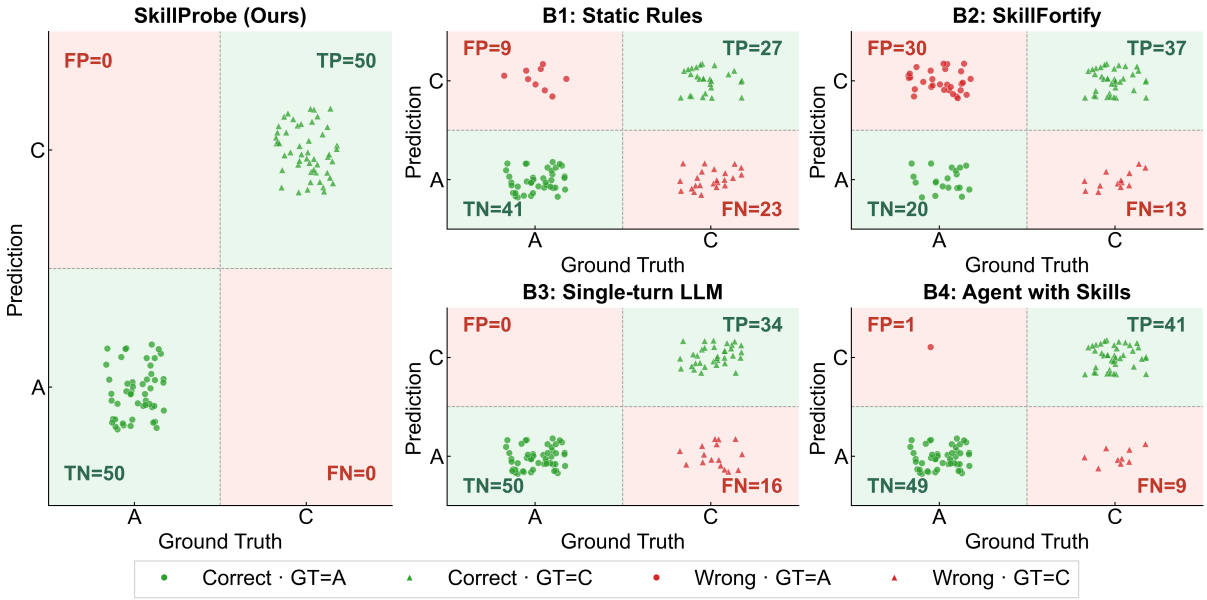

Figure 4: Audit result distribution of SkillProbe vs. baselines on 100 labeled samples. The left panel shows SkillProbe, whereas the right panels display the baseline results. Green/red backgrounds signify correct/incorrect predictions, respectively. Points are color-coded by correctness (Green=Correct) and shape-coded by Ground Truth (Circle=APPROVED, Triangle=CONDITIONAL).

To detect combinatorial threats, we established nine link policies (Appendix C), mapping output risk tags to input sensitivity tags (Sinks). The auditing verdicts are categorized into two primary classes, namely Approved (A), indicating the skill has passed all security checks, and Conditional (C), signifying the detection of potential security risks that require mitigation or restricted usage.

4.2 Effectiveness Comparison (RQ1)

We formulate skill auditing as a binary classification task to facilitate a rigorous quantitative comparison. The comparative analysis against four baselines (Figure 4) on a curated evaluation set of 100 samples reveals a fundamental performance gap between traditional methods and our structured multi-agent architecture. To construct this benchmark, we curated 50 representative benign skills by selecting the most prevalent and verified entries from ClawHub. Assisted by a Claude-based coding agent, we synthesized 50 adversarial counterparts by systematically injecting hidden vulnerabilities into the original codebases, ensuring a uniform distribution across compliance breaches, semantic misalignments, and combinatorial threats.

Limitations of Monolithic and Static Paradigms. Traditional tools without LLMs, such as B1 (regular expression scanning driven by static rules) and B2 (SkillFortify (Bhardwaj, 2026)), struggle to achieve a satisfactory trade-off between precision and recall. B1 exhibits significant noise (FP=9) due to a lack of semantic context, while B2's aggressive logic yields a prohibitive 30% False Positive Rate (FPR), rendering it impractical for production. Without LLM-based reasoning, these methods often misinterpret complex string manipulations as malicious patterns. B3 (Single-turn LLM) eliminates noise but misses 32% of threats (FN=16), exposing a semantic bottleneck in identifying subtle implementation-specification misalignments. Even when equipped with tools (B4), the lack of a reflexive design pattern leads to inconsistent recall. This bottleneck arises because a small number of interaction rounds often fails to reconcile high-level specification with the nuanced execution logic, especially when malicious intent is obfuscated through indirect prompts.

The Design Advantage. SkillProbe attained optimal performance on our curated dataset, achieving excellent precision and recall (TP=50, TN=50). This success is a direct consequence of our three-phase decomposition, where the *Gatekeeper* ensures high-recall filtering, while the *Alignment Detector* and *Flow Simulator* suppress noise and uncover stealthy compositional chains. This validates that decoupling security knowledge into pluggable skills is essential for detecting high-stakes risks.

1-Nex-N1

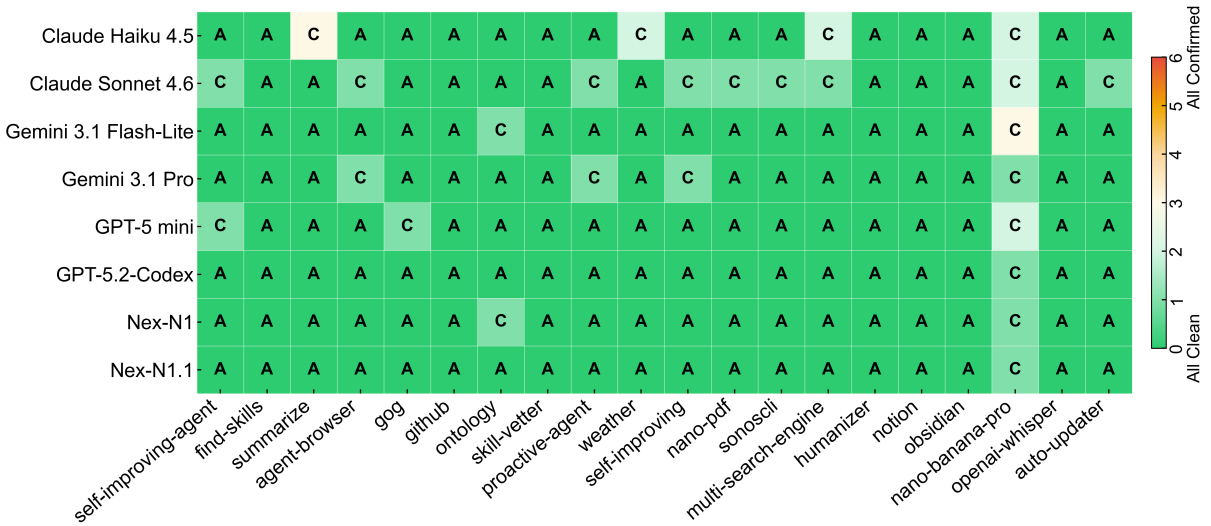

Figure 5: Cross-model audit consistency heatmap for ClawHub top 20 Skills. Each cell represents the composite risk score for each model-skill pair, computed as the sum of per-dimension severity scores (*malicious patterns*, *semantic consistency*, *composition safety*), where scores correspond to security states (CLEAN= 0, SUSPECTED= 1, and CONFIRMED= 2). Cell labels A and C denote the final scorecard verdict (APPROVED and CONDITIONAL, respectively). Skills are ordered by download count (left to right, descending).

While these results reflect the efficacy of SkillProbe on the labeled subset, the inherent complexity of open-world skills may still present edge cases.

Summary of RQ1: *SkillProbe yields significant improvement over baselines on the curated evaluation set, validating the necessity of structured multi-agent collaboration for detecting complex skill vulnerabilities.*

4.3 Cross-model Robustness (RQ2)

The evaluation across eight LLMs highlights the resilience of SkillProbe against the non-deterministic nature of reasoning engines.

The Auditing Trilemma. Our results in Table 1 reveal a fundamental trade-off among **Reasoning Depth**, **Security Stringency**, and **Efficiency** that defines the current landscape of LLM-based security auditing. Claude Sonnet 4.6 exhibits the characteristics of a "Strict Auditor" (45% CONDITIONAL rate), providing the most exhaustive findings and excelling at uncovering subtle semantic deviations. Meanwhile, GPT-5.2-Codex prioritizes remediation over rejection, yielding 2.3 recommendations per skill, significantly higher than its peers. Conversely, Gemini 3.1 Flash-Lite and Nex-N1 represent the efficiency frontier, achieving sub-40-second inference latencies. These models are ideal for real-time admission filtering in marketplaces, though they tend to produce fewer actionable recommendations. The positive correlation between reasoning latency and recommendation count suggests that generating actionable remediation requires higher-order cognitive synthesis, which is currently an emergent capability in high-end models.

Consensus Convergence. Despite model-wise predispositions, Figure 5 demonstrates strong consensus on high-risk skills (e.g., *nano-banana-pro*). This convergence suggests that SkillProbe successfully extracts intrinsic security properties that are orthogonal to model-specific variances. Specifically, our "Skills-for-Skills" paradigm acts as a stabilizer, i.e., while the underlying model's response may change, the structured workflow guides the agents toward a unified security verdict. By formalizing security knowledge as pluggable semantic schemas, we ensure that the framework's reliability is independent of the specific underlying LLM, effectively bridging the gap between stochastic AI and deterministic security auditing.

Summary of RQ2: *SkillProbe demonstrates high cross-model robustness, ensuring that decisions are grounded in the target skill's intrinsic properties rather than model-specific biases.*

4.4 Large-scale Ecosystem Insights (RQ3)

A deep scan of top 2,500 skills with Nex-N1 provides an empirical view of the Agentic Web's security posture, yielding two critical insights into the current state of agentic security.

Table 1: Per-model audit statistics across ClawHub top 20 skills.

Model	Avg. Time (s)	Approved (%)	Conditional (%)	Warn. Findings	Info Findings	Avg. Recs
Claude Haiku 4.5	86.7	80%	20%	4	41	0.5
Claude Sonnet 4.6	228.6	55%	45%	10	82	1.7
Gemini 3.1 Flash-Lite	18.5	90%	10%	2	34	0.6
Gemini 3.1 Pro	221.8	80%	20%	8	44	1.4
GPT-5 mini	171.3	85%	15%	9	88	1.9
GPT-5.2-Codex	120.3	95%	5%	2	77	2.3
Nex-N1	38.7	90%	10%	2	33	0.7
Nex-N1.1	282.0	95%	5%	1	35	0.9

Avg. Time: mean elapsed seconds for completed (non-blocking) audits. Approved/Conditional: share of 20 skills by final scorecard verdict. Warn./Info: total findings by severity across 20 skills. Avg. Recs: mean number of actionable recommendations per skill.

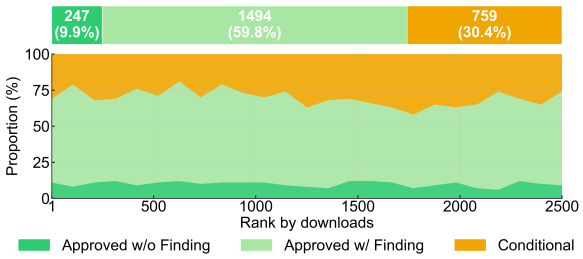

Figure 6: Security audit overview of the top 2,500 most-downloaded skills on ClawHub. The upper panel shows the distribution of final audit verdicts within our sample of 2,500 leading skills, while the lower panel depicts the stacked area chart showing the proportion of each verdict category across 25 equal-sized bins sorted by download rank.

Table 2: Ablation study of SkillProbe. The table compares the full SkillProbe workflow against variants where specific components are removed or isolated. The baseline "w/o Specialized Skills" represents the base agent performing the task using only its internal knowledge without any external auditing skills.

Method	F1	Prec.	Rec.	FPR
SkillProbe	**100.0%**	**100.0%**	**100.0%**	**0.0%**
w/o Specialized Skills	78.6%	97.1%	66.0%	2.0%
Only Skill-Vetter	69.9%	67.9%	72.0%	34.0%
w/o Gatekeeper	75.0%	100.0%	60.0%	0.0%
w/o Alignment Detector	96.2%	92.6%	100.0%	8.0%
w/o Flow Simulator	99.0%	98.0%	100.0%	2.0%

F1, Precision (Prec.), Recall (Rec.), and False Positive Rate (FPR) are binary classification metrics (clean vs. malicious). Best results are highlighted in bold.

The Popularity Paradox. As illustrated in Figure 6, only 247 (9.9%) are in a fully "clean" state (i.e., approved without any findings) among the 2,500 leading skills. More concerningly, the stacked area chart reveals a counter-intuitive phenomenon. Contrary to the assumption that high-download volume implies higher quality, we observed a persistent security vacuum across all popularity tiers. The statistical invariance of the risk ratio (between 20% and 35%) indicates that community scrutiny alone is inadequate for mitigating stealthy risks, underscoring the need for automated and platform-level skill auditing.

Systemic Risk Connectivity. The emergence of a giant connected component in the risk-link network, as shown in Figure 7, demonstrates that agentic threats are systemic rather than isolated. The network comprises 499 skills and 75,373 risk edges, exhibiting high degree centrality. Data Exfiltration and Fact Poisoning account for over 80% of edges, and the full interconnection of the high-risk subset implies that any combination of these skills potentially triggers cascaded exploits. From a defensive standpoint, this highly interconnected structure indicates a non-linear expansion of the attack surface rather than a simple linear addition. This empirical evidence strongly validates the necessity of the cross-skill combinatorial simulation phase, i.e., traditional atomic auditing tools are fundamentally incapable of capturing such cascaded risks within a fully interconnected threat landscape.

Summary of RQ3: *Security risks are prevalent and statistically orthogonal to popularity, and the interconnected nature of these risks validates the critical need for compositional simulation.*

4.5 Ablation Study (RQ4)

The ablation results in Table 2 quantify the asymmetric contributions and inter-phase synergies of the SkillProbe pipeline.

The Power of Orchestration. The significant performance drop in the variant without specialized skills highlights the limitations of using a base agent for complex auditing. While models can

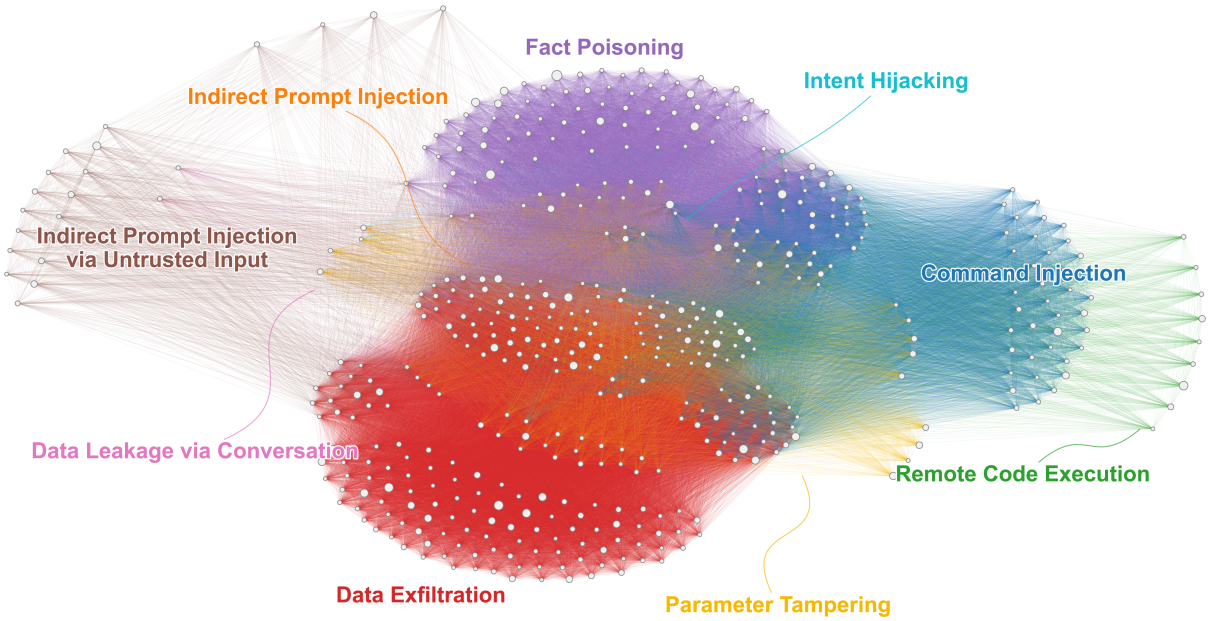

Figure 7: Risk chain network of the 499 skills with at least one confirmed exploit from the nine attack chains. Each node represents one skill, with node size scaling with download count. Edge color encodes the shared risk chain type between two connected skills (nine colors corresponding to nine risk categories as illustrated in Table 3). Edges connect any two skills that share at least one confirmed chain type.

identify explicit intent, they struggle with the multi-step reasoning required for tracking latent semantic misalignments and cross-skill data flows. This confirms that structured orchestration is the primary driver of high-assurance auditing.

Asymmetric Phase Contributions. Each phase serves a distinct defensive role. The *Gatekeeper* (Phase 1) acts as the high-recall sentry, whose removal results in the most significant degradation in F1-score. The *Alignment Detector* (Phase 2) serves as the precision anchor, effectively minimizing the FPR by verifying documentation-code alignment. Finally, the *Flow Simulator* (Phase 3) provides the final layer of defense against semantically benign but compositionally lethal attack chains.

The Necessity of the Workflow. Notably, the isolated Skill-Vetter[10], which is the skill utilized by *Gate Executor* agent and the most popular auditing skill on ClawHub, underperforms even the monolithic LLM. This reveals that individual security skills are inherently noisy for automated gatekeeping, and SkillProbe's multi-layered check-and-balance system is essential to achieve zero-noise detection without compromising recall.

Summary of RQ4: *All three phases are indispensable, and the multi-layered check-and-balance system is the key to achieving optimal performance with zero false positives on our curated dataset.*

4.6 Case Studies

We conducted manual verification of flagged skills. Two representative cases (semantic deception and combinatorial remote code execution) are detailed in Appendix D. The manual audit confirmed that SkillProbe's detections correspond to verifiable security flaws rather than heuristic artifacts.

5 Conclusion

This paper presented SkillProbe, a multi-agent framework for auditing security vulnerabilities in agent skill marketplaces. By formalizing semantic-behavioral inconsistency and inter-skill combinatorial risks, SkillProbe shifts the security paradigm from reactive runtime interventions to proactive pre-distribution auditing. Our evaluation on 2,500 real-world ClawHub skills reveals a popularity paradox and a giant connected component in the risk-link network, demonstrating that compositional simulation is necessary for comprehensive auditing. As LLM-driven workflows integrate into critical infrastructure, automated collaborative auditing will become a prerequisite for trustworthy autonomous agents. Future work includes dynamic sandboxing for obfuscated code (Triedman et al., 2025) and reinforcement learning for discovering zero-day combinatorial patterns (Yang et al., 2026).

[10] https://clawhub.ai/spclaudehome/skill-vetter

Limitations

Despite its capabilities, SkillProbe faces several limitations that warrant discussion.

Benchmark Construction and Potential Circularity. The 50 adversarial samples in our evaluation set were generated with assistance from a Claude-based coding agent. This introduces potential circularity, i.e., adversarial patterns generated by a model may be those the same model family is best equipped to detect, potentially inflating recall on the labeled set. We mitigate this concern through two mechanisms, including (1) reporting cross-model auditing results across eight LLMs from four model families, which shows high inter-model agreement, and (2) verifying flagged cases through independent manual code audits. Future work should evaluate on adversarial samples generated by human red-team members or entirely different model families.

Statistical Considerations. Our evaluation set uses a balanced 50/50 split that does not reflect real-world class prevalence. If malicious skills represent a small fraction of the marketplace, precision and the false positive rates will behave differently at scale than the reported numbers suggest. We report these metrics as preliminary validation. Evaluating on a stratified sample across skill categories and vulnerability types, with appropriate confidence intervals, remains important future work.

Detection Scope. First, SkillProbe's analytical fidelity is bounded by LLM reasoning limits, particularly with highly obfuscated code. Moreover, black-box third-party API calls represent a fundamental blind spot for any static analysis. A malicious skill could implement dangerous logic server-side while appearing clean to local inspection. Second, our combinatorial simulations rely on predefined risk link policies derived from established security frameworks and expert knowledge. The framework may exhibit reduced sensitivity toward out-of-distribution adversarial patterns. Third, SkillProbe lacks an intent-level filtering layer. Skills whose stated purpose is itself malicious (e.g., automated phishing) would pass Phase 2 if documentation and code are consistent, which is a limitation shared by most current auditing tools.

Ecological Validation. Our findings have not been cross-referenced against ClawHub's existing defenses (VirusTotal, LLM-based code analyzer), as their scanner outputs are not publicly accessible. Additionally, responsible disclosure to affected skill developers has not yet been conducted. Both represent critical next steps for independently validating the reported findings and facilitating ecosystem remediation.

Ethical Considerations

Dual-use risks and responsible release. This work studies security vulnerabilities in real agent skill marketplaces, which creates potential dual-use risks. We report attack patterns at an abstract level and use case studies to illustrate root causes rather than provide step-by-step exploit instructions. Public release of reproduction artifacts will avoid weaponized payloads and direct exploit scripts, and any actionable findings concerning affected skills will be handled through responsible disclosure before unrestricted release.

Data and privacy. Our analysis uses publicly available skill packages and their documentation from ClawHub. We do not collect private user data or interaction logs. The released artifacts will include only data needed to support reproducibility and will exclude credentials, secrets, or personal information discovered during auditing.

Use of generative AI assistance. We used a Claude-based coding agent to assist in constructing the 50 adversarial skill variants used in our curated evaluation benchmark. The assistant was used for code mutation and vulnerability-injection suggestions. All generated variants were manually inspected by the authors to verify that the injected behaviors matched intended threat categories, remained executable, and did not introduce unrelated functionality. We also used generative AI tools for language polishing of author-written text. No automatically generated citations, factual claims, related-work summaries, or research conclusions were retained without author verification. Figure 1 was created with assistance from a generative image model as an illustrative visualization of the threat setting; it is not used as empirical evidence. The authors take full responsibility for the correctness of the methods, results, writing, and visual materials. We discuss potential circularity introduced by AI-assisted benchmark construction in the Limitations section.

References

Anthropic. 2025. Introducing Agent Skills. https://claude.com/blog/skills.

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

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. *Advances in Neural Information Processing Systems*, 37:82895–82920.

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. 2024. Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720*.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. 2026. SoK: Agentic Skills – Beyond Tool Use in LLM Agents. *arXiv preprint arXiv:2602.20867*.

Jing-Jing Li, Jianfeng He, Chao Shang, Devang Kulshreshtha, Xun Xian, Yi Zhang, Hang Su, Sandesh Swamy, and Yanjun Qi. 2025. STAC: When Innocent Tools Form Dangerous Chains to Jailbreak LLM Agents. *arXiv preprint arXiv:2509.25624*.

Zhihao Li, Boyang Ma, Xuelong Dai, Minghui Xu, Yue Zhang, Biwei Yan, and Kun Li. 2026. Don't believe everything you read: Understanding and Measuring MCP Behavior under Misleading Tool Descriptions. *arXiv preprint arXiv:2602.03580*.

Fengyu Liu, Yuan Zhang, Jiaqi Luo, Jiarun Dai, Tian Chen, Letian Yuan, Zhengmin Yu, Youkun Shi, Ke Li, Chengyuan Zhou, Hao Chen, and Min Yang. 2025. Make Agent Defeat Agent: Automatic Detection of Taint-Style Vulnerabilities in LLM-based Agents. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3767–3786.

Yi Liu, Zhihao Chen, Yanjun Zhang, Gelei Deng, Yuekang Li, Jianting Ning, Ying Zhang, and Leo Yu Zhang. 2026a. Malicious Agent Skills in the Wild: A Large-Scale Security Empirical Study. *arXiv preprint arXiv:2602.06547*.

Yi Liu, Weizhe Wang, Ruitao Feng, Yao Zhang, Guangquan Xu, Gelei Deng, Yuekang Li, and Leo Zhang. 2026b. Agent Skills in the Wild: An Empirical Study of Security Vulnerabilities at Scale. *arXiv preprint arXiv:2601.10338*.

Sheena Panthaplackel, Junyi Jessy Li, Milos Gligoric, and Raymond J. Mooney. 2021. Deep Just-In-Time Inconsistency Detection Between Comments and Source Code. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 427–435.

David Schmotz, Luca Beurer-Kellner, Sahar Abdelnabi, and Maksym Andriushchenko. 2026. Skill-Inject: Measuring Agent Vulnerability to Skill File Attacks. *arXiv preprint arXiv:2602.20156*.

Marcos Tileria, Jorge Blasco, and Santanu Kumar Dash. 2024. DocFlow: Extracting Taint Specifications from Software Documentation. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–12.

Harold Triedman, Rishi Jha, and Vitaly Shmatikov. 2025. Multi-agent systems execute arbitrary malicious code. *arXiv preprint arXiv:2503.12188*.

Chengpeng Wang, Jipeng Zhang, Rongxin Wu, and Charles Zhang. 2024. DAInfer: Inferring API Aliasing Specifications from Library Documentation via Neurosymbolic Optimization. *Proceedings of the ACM on Software Engineering*, 1(FSE):2469–2492.

Xiaomeng Xu, Zahin Wahab, Reid Holmes, and Caroline Lemieux. 2025. DocPrism: Local Categorization and External Filtering to Identify Relevant Code-Documentation Inconsistencies. *arXiv preprint arXiv:2511.00215*.

Wanrong Yang, Alberto Acuto, Yihang Zhou, and Dominik Wojtczak. 2026. A survey for deep reinforcement learning based network intrusion detection. *Applied AI Letters*, 7(2):e70026.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. 2024. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10471–10506.

A Data Availability

The datasets, reproduction packages, and auditing results reported in this paper are available in our reproduction package. For the purpose of double-blind review, the package can be accessed anonymously at [Anonymous Link]. A permanent DOI has been reserved for the camera-ready version.

B Implementation and Deployment

We developed a functional prototype of SkillProbe built upon a modern web stack including FastAPI (backend) and Vue 3 (frontend). For scalability, the backend leverages `asyncio` and distributed task queues to orchestrate high-concurrency auditing workloads. To ensure computational efficiency, we employ SHA-256 fingerprinting for cross-tenant de-duplication, ensuring that redundant versions of popular skills do not consume unnecessary compute cycles. The web interface (Figure 8) facilitates intuitive result visualization, where the depicted statistics reflect a local test environment. We also provide a productivity-oriented Command-Line Interface (CLI) for batch processing and a "Conversation-as-Auditing" plugin for

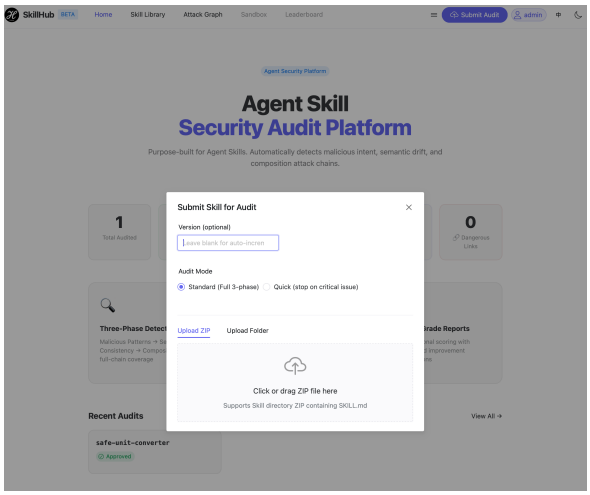

Figure 8: The web interface of SkillProbe. The screenshot illustrates the platform's homepage with the "Submit Skill for Audit" modal active.

Table 3: Risk link policies for combinatorial attack detection.

Output Risk (Source)	Input Sensitivity (Sink)	Attack Pattern
Unsanitized external string	Shell/Interpreter execution	Command Injection
Unsanitized external string	LLM prompt injection	Indirect Prompt Injection
Unsanitized external string	Sensitive action parameter	Parameter Tampering
Executable code fragments	LLM prompt injection	Indirect Prompt Injection
Executable code fragments	Shell/Interpreter execution	Remote Code Execution
Sensitive credentials/data	External network egress	Data Exfiltration
Sensitive credentials/data	LLM prompt injection	Data Leakage
Manipulative semantics	Persistent state/config	Fact Poisoning
Manipulative semantics	Sensitive action trigger	Intent Hijacking

seamless integration into existing agentic development workflows, with native support for the OpenClaw ecosystem.

C Risk Link Policy Derivation

The nine core risk link policies (Table 3) were derived by combining established security frameworks (CWE Top 25, OWASP LLM Top 10) with domain expertise, then iteratively mapped to the agent skill context by the authors.

While these policies cover the most prevalent attack patterns observed in our dataset (accounting for over 80% of detected risk edges), they are not exhaustive. The framework can be extended with additional policies as new threat patterns emerge.

D Case Studies with Manual Verification

We conducted a deep-dive manual verification of our auditing reports, with two representative vulnerabilities selected to illustrate how our approach bridges the gap between automated flagging and root-cause analysis.

Case A: Semantic Deception (whisper-mlx-local). This skill is advertised as a private and local-only service. However, our *Alignment Detector* identified hidden `POST` requests to cloud APIs. Manual inspection confirmed a hidden fallback logic that silently exfiltrates raw audio buffers when local resources are constrained. This is a classic hidden functionality exploiting the "local" designation as a semantic masquerading vector. For users dealing with sensitive scenarios such as meeting recordings, medical voice recordings, and legal conversations, this poses a severe threat to data confidentiality.

Case B: Combinatorial Remote Code Execution (agent-browser × agent-task-manager). Individually, these skills appear benign when analyzed in isolation. However, our *Flow Simulator* detected a critical link, whereby unsanitized strings from the browser (e.g., a malicious Python payload from a web page) are propagated to the task manager's un-sandboxed `eval()` sink. We successfully simulated a remote code execution (RCE) exploit using this chain, demonstrating that compositional security is an emergent property that conventional atomic auditing is fundamentally ill-equipped to detect.

Manual Validation and Insights. To evaluate the fidelity of SkillProbe's automated reports, we performed a post-mortem manual code audit on the flagged skills. The results across these cases (and others in the above) confirm that the detections of SkillProbe are verifiable security flaws rather than heuristic-based artifacts. Specifically, manual inspection of Case A confirmed that the exfiltration logic was concealed within complex error-handling control flows, a pattern notoriously difficult for static analyzers to trace but successfully caught by our approach. Similarly, the successful reproduction of the RCE exploit in Case B validates that our approach can effectively predict emergent risks arising from the composition of individually benign components. This alignment between automated multi-agent auditing and expert manual verification underscores the framework's capability to bridge the gap between high-level semantic intent and low-level execution logic.

Summary of Case Studies: *Manual verification confirms that SkillProbe accurately identifies both covert functionalities masked by misleading documentation (Case A) and emergent exploits through cross-skill chains (Case B). This highlights the framework's unique ability to transform abstract*

security patterns into actionable and verifiable auditing intelligence.

E Further Discussion

The Popularity Paradox and Market Standardization. One of the most striking empirical revelations of this research is the significant decoupling between a skill's popularity and its demonstrable security guarantees. Our results indicate that even among elite skills with high download volumes, the prevalence of potential risks is comparable to that of long-tail modules. This *Popularity Paradox* exposes a structural deficit in the security governance of current agent marketplaces such as ClawHub. Users often conflate high download counts with software quality, engendering a misplaced trust that adversarial actors can instrumentalize. Meanwhile, developers under the pressure of rapid iteration frequently bypass rigorous security scrutiny. These findings underscore that in a burgeoning Agentic Web ecosystem, an automated and multi-dimensional auditing protocol like SkillProbe is not a peripheral utility but a primitive for a trustworthy software supply chain.

The Semantic Gap beyond Code-Level Scrutiny. While traditional software auditing predominantly targets logical vulnerabilities within the executable code, this research demonstrates that semantic-behavioral inconsistency has emerged as a primary threat vector. Since an agent's selection of a skill is predicated on its natural language description while execution is anchored in script logic, this cross-modal discrepancy creates a clandestine attack vector for adversaries. The findings from SkillProbe urge the security community to shift focus from isolated vulnerability scanning to semantic-behavioral alignment verification, elevating semantic auditing to a first-class concern in the era of LLM-based orchestration.

Paradigm Shift from Atomic Inspection to Compositional Defense. The giant connected component revealed in our risk-link map highlights the risk of cascading failures within the agentic ecosystem. Traditional atomic auditing is fundamentally ill-equipped to capture these dynamic threats that only manifest during skill composition. The existence of over 75,000 risk edges suggests that a complex "risk highway" has already formed within the ecosystem, whereby adversarial data flows can circumvent traditional perimeter-based defenses. The *Flow Simulator* validates the feasibility of combinatorial risk prediction under linear complexity, providing a scalable template for mitigating the threat of combinatorial vulnerability expansion.