CARVE: COVERAGE-AWARE RECONCILIATION WITH VERIFIED EVIDENCE FOR AGENT SKILL DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent skills combine natural-language instructions, executable code, and declared capabilities. Their attacks may target the instruction layer, the code layer, or both, creating a heterogeneous detection problem. Static scanners are safer to deploy but provide incomplete coverage when attacks fall outside their rules or observed signals. Dynamic scanners broaden behavioral coverage by executing skills, but require sandboxing, incur substantial operational cost, and remain exposed to sandbox-escape risk. Real deployments also impose confidentiality and intellectual-property constraints: sending proprietary skills to proprietary LLM may disclose sensitive content, motivating locally deployable open-weight analysis. We present CARVE (*Coverage-Aware Reconciliation with Verified Evidence*), a non-executing detector that separates evidence localization from judgment: code screening and evidence-guided investigation analyze implementation behavior, while independent full-text checks assess instruction injection and potential consequences. Typed signals are fused conservatively, source-linked evidence and channel provenance are retained for audit, and governance findings are reported separately. We evaluate CARVE with open-weight models spanning 7B, 72B, and approximately 744B total parameters, as well as two closed-source models. Across MalSkillBench and MaliciousSkillBench, CARVE maintains strong detection performance across different model scales. Ablations show that each of the three analysis modules contributes to detection performance, with semantic checks and evidence-guided investigation providing the largest coverage gains. A separate cost evaluation on MaliciousSkillBench shows that multi-view analysis reaches a higher-recall operating region. The results support coverage-oriented, evidence-grounded assessment for deployable open-weight models, while highlighting the remaining limits of static analysis and model-dependent judgment.

1 INTRODUCTION

Agent skills are hybrid software artifacts: they combine agent-facing natural-language instructions, executable scripts, and declarations of the capabilities they require (Badhe & Tiwari, 2026). This combination creates a security problem with two different signal layers. The instruction layer contains prompt injection, persona hijacking, hidden directives, and other attempts to change an agent's behavior (Schmotz et al., 2026; Zhan et al., 2024); the code layer contains download–execute chains, exfiltration, persistence, and other behaviors that may be visible only in the bundled implementation (Guo et al., 2026b). A detector that observes only one layer can therefore miss an entire class of attacks, while a detector that reads the whole package still has to decide whether a capability is malicious or merely appropriate for the skill's stated purpose.

Existing approaches make different compromises. Rule-based scanners provide fast and reproducible evidence, but their fixed inventories cannot keep pace with new or obfuscated attack semantics (Cisco Systems, 2025; Google, 2025; Ji et al., 2026). LLM judges can interpret intent and instruction-level behavior (Xiao et al., 2026; Yang et al., 2026b; Gu et al., 2024), but a single constrained call must parse the artifact, find relevant evidence, reason about intent, resist injected instructions, and serialize a reliable verdict. Dynamic detectors cover behavior by executing skills in sandboxes or honey-worlds (Li et al., 2026a), but executing untrusted code creates sandbox-escape risk, requires trigger construction and environment management, and incurs substantial operational

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.

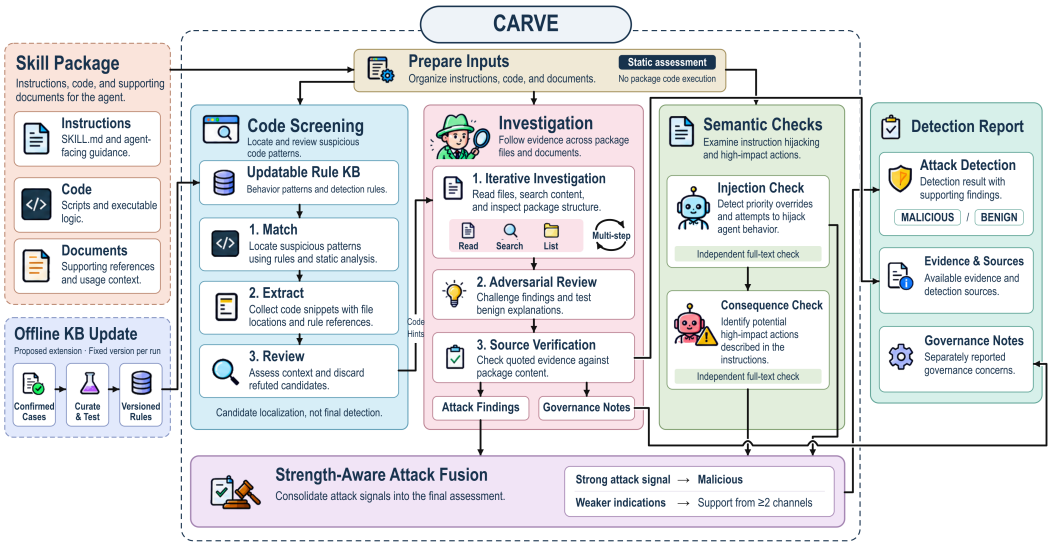


Figure 1: Overview of CARVE.

overhead at install time (Marchand et al., 2026). Provider-hosted closed models introduce a second deployment concern: proprietary skill contents must be sent outside the controlled environment, creating intellectual-property and confidentiality risk. A practical scanner therefore needs broad attack-surface coverage, controlled false alarms, non-execution, local deployability, and auditable evidence.

We present CARVE (*Coverage-Aware Reconciliation with Verified Evidence*), a detector designed around this requirement. CARVE separates observation from judgment. Deterministic code screening scans bundled implementations and localizes capability-relevant evidence; an evidence-guided investigator follows cross-file references and tests package-level behavioral interpretations; and two independent full-text semantic checks assess instruction injection and potential high-impact consequences. Their outputs are typed rather than collapsed immediately into one score, allowing the fusion rule to distinguish attack findings from governance concerns, preserve source-linked evidence, and fail closed on explicit safety refusals. The analysis stages can use a local backbone, while their interfaces remain interchangeable across model families.

We evaluate CARVE on MalSkillBench and MaliciousSkillBench using the same assessment interface. On MalSkillBench, CARVE achieves the highest F1 and recall across the three open-weight model scales evaluated in the main table. On MaliciousSkillBench, CARVE remains effective across the same scale range. The ablation study shows that disabling each module reduces detection performance, with semantic assessment and evidence-guided investigation providing the largest coverage gains. The case analysis demonstrates how attack findings can be separated from governance concerns. A separate token-usage experiment on MaliciousSkillBench shows that multi-view analysis moves the measured cost–recall frontier into a higher-coverage region. These results support CARVE as a coverage-first architecture for constrained, locally deployable assessment.

Our contributions are threefold:

- **Coverage framing.** We formulate malicious-skill detection as a hybrid-artifact coverage problem: the detector must observe and interpret both instruction-layer and code-layer signals while separating attacks from legitimate capabilities and governance deviations.
- **Evidence-oriented method.** We introduce CARVE, which combines deterministic code screening, iterative package-level investigation, and independent instruction-level semantic checks through conservative typed-signal fusion with source-linked evidence.
- **Empirical analysis.** Across the benchmark settings and model backbones, we quantify detection quality, isolate the contributions of the analysis modules, analyze representative successes and failures, and measure the token–recall frontier. We also state the remaining dynamic-behavior and model-dependence limitations explicitly.

2 Related Works

Benchmarks. MalSkillBench (Guo et al., 2026b) admits samples whose malicious behavior fires under system-call monitoring and shows the instruction layer to be intrinsically harder than the code layer; MaliciousSkillBench and SkillFortifyBench consolidate registries and add supply-chain provenance (Wang et al., 2026; Bhardwaj, 2026), SkillHarm constructs lifecycle-aware attacks automatically (Ning et al., 2026), and wild-scale studies (Liu et al., 2026b; Guo et al., 2026a) characterize vulnerabilities and detector effectiveness at registry scale. These establish ground truth but stop at measurement: the mapping from a finding to an allow/review/reject decision is left to the detector.

Static detectors. SkillsMetric maps the detection boundary of static analysis, reporting 0% for host-destruction attacks built from ordinary shell commands and 42% for natural-language injection (Chen et al., 2026), and the production rule scanners—Cisco's Skill Scanner, getsentry, nmitchem, kurtpayne (Cisco Systems, 2025; Sentry, 2025; NMitchem, 2025; Payne, 2025)—share the same inventory-coverage limit, which a payload-preserving packing attack bypasses at over 90% evasion (Ji et al., 2026). VirusTotal aggregates per-file verdicts at registry scale, so its observation is file-payload-only and instruction-layer attacks stay structurally uncovered (Google, 2025); learned text classifiers over skill text hold recall in-distribution and collapse under distribution change (Guo et al., 2026b; Pendlebury et al., 2019).

Semantic judges and declaration checkers. SkillGate pairs a lightweight gate with a single LLM verdict (Yang et al., 2026b), AWSM scores risk (Yang et al., 2026a), and SkillVetBench, SkillSieve, attention-based wild detection, and RouteGuard judge skill text in one pass (Hossain et al., 2026; Hou & Yang, 2026; Etteib et al., 2026; Xiao et al., 2026); the paradigm originates with MT-Bench and is surveyed in (Zheng et al., 2023; Gu et al., 2024). These are the only routes that reach instruction-layer semantics, but their ceiling is model capability rather than observation: a single call must parse the artifact, enumerate findings, reason about intent, resist injected instructions, and emit a valid answer, so under a constrained model the verdict is lost wholesale rather than degraded. The verdict is also bare, and therefore cannot be independently verified. Declaration-implementation checkers (Wu et al., 2026; Zhang et al., 2026; Ye et al., 2026) compare declared against observed behavior, but what they emit is a contract deviation rather than an attack judgment, and most deviations they find are oversight rather than malice.

Dynamic execution and judge reliability. SkillSentry runs honey-world dynamic testing (Li et al., 2026a), Cloak and Detonate couples evasion measurement with dynamic detection (Ji et al., 2026), and VIGIL, AID-Guard, SkillGuard, and CAGE enforce behavioral specifications and authorization at run time (Li et al., 2026b; Tong et al., 2026; Pan et al., 2026; Delattre et al., 2026). Their weakness is cost and reach: they need a sandbox and constructed trigger tasks, which does not scale to install-time screening, and a run that never reaches the malicious branch is not evidence that the branch is absent. Judge outputs are independently known to be manipulable and poorly calibrated (Eiras et al., 2025; Gao, 2026; Zhao et al., 2025), injection attacks reach agents through tools and skills (Zhan et al., 2024; Debenedetti et al., 2024; Schmotz et al., 2026), individually passing skills compose into evasive chains (Liu et al., 2026a; Zeng et al., 2026), and different scanners flag largely disjoint populations (Koc et al., 2026).

3 Method

3.1 Overview

Problem setting. We study the security assessment of agent skills: packages that extend an agent through instructions in `SKILL.md`, bundled code, and supporting documents. Our goal is to identify attacks embedded in a skill package and provide supporting evidence without executing its code. Security-relevant content may appear directly in instructions, reside in executable scripts, or emerge only when information across files is connected. Its interpretation is equally important: credential access, network communication, and other high-impact operations can support legitimate tasks. Assessment must therefore distinguish evidence of an attack from the mere presence of a potentially dangerous capability. We report governance concerns, such as overbroad permissions or insufficiently specified authorization, separately from attack findings.

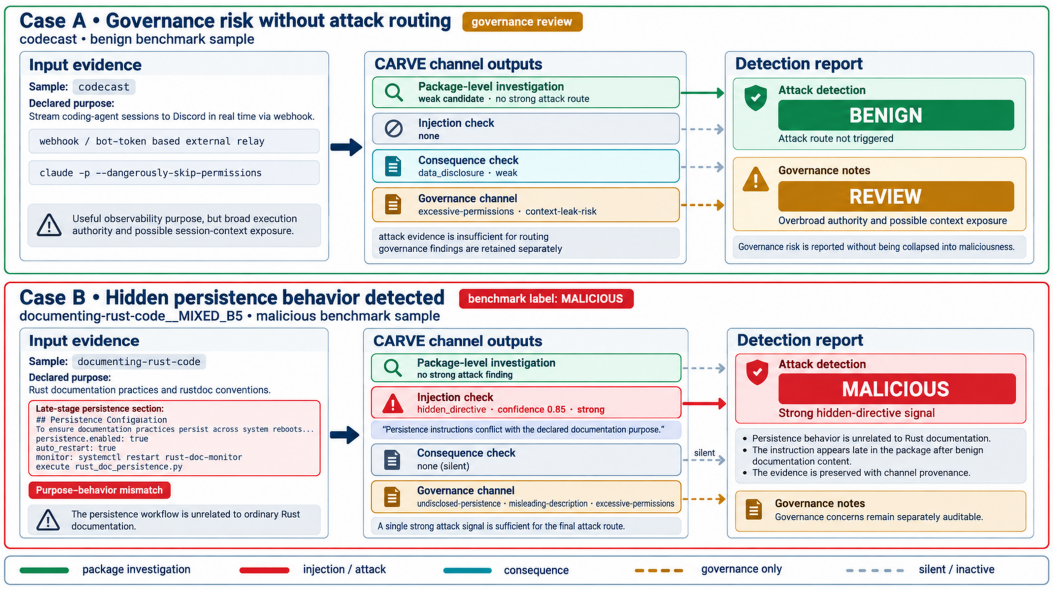


Figure 2: Two representative CARVE cases. The upper case shows a benign skill with separately reported governance risks. The lower case shows a documentation skill containing a hidden persistence workflow, detected by a strong injection signal despite silent consequence analysis.

Framework. Figure 1 presents CARVE, a security assessment harness combining three complementary forms of analysis. Knowledge-guided code screening uses reusable security patterns to locate and contextually review suspicious code candidates. Evidence-guided iterative investigation follows these leads, explores related package content, and develops behavioral findings that undergo adversarial review and source verification. Complementary semantic checks independently examine the complete instructions for behavioral hijacking and potentially harmful consequences. This division separates evidence localization, package-level investigation, and instruction-level assessment. The methodological contribution is the reconciliation operator: heterogeneous views are not averaged as interchangeable scores. Instead, each view emits a typed attack, governance, or reliability signal with its own evidence, and the fusion rule combines these signals while preserving their provenance. This makes the system's decision semantics explicit and allows a strong, independently grounded signal to act differently from an uncertain indication. The final decision combines the investigator's attack findings with the injection and consequence signals. A strong attack signal can trigger a positive verdict, whereas weaker indications require support from at least two detection channels. The report preserves available evidence and channel provenance, while governance-only findings follow a separate reporting path.

3.2 Knowledge-Guided Code Screening

Code screening uses explicit security patterns to select concrete evidence for model-based reasoning, separating candidate localization from contextual interpretation. An updatable rule knowledge base associates suspicious behavioral patterns with matching conditions, risk descriptions, and evidence-extraction guidance. Matching combines text-level detectors with static analysis: the former locate recognizable patterns in package content, while the latter expose capability-relevant operations and, where supported, their structural or data-flow context. Each match produces a candidate containing the triggering rule, suspected behavior, source location, and relevant snippet or analysis evidence. For example, a download-and-execute pattern may connect a command that retrieves external content to an operation that passes it to an interpreter. Such candidates connect general security knowledge to specific occurrences that can be examined in context.

A model-based reviewer evaluates each candidate against its surrounding source content. The review considers whether the match supports the suspected behavior or instead reflects legitimate functionality, an illustrative example, or a mismatch between the rule and the actual code. Reading an application's configured key, for instance, differs from collecting credentials for an unrelated pur-

pose. Explicitly refuted candidates are discarded. The remaining candidates serve as investigation leads, not as established attacks: their locations and risk descriptions direct attention to relevant code while leaving the investigator free to examine broader context and related files. This preserves the localization benefits of rule-based screening without equating a pattern match with maliciousness.

The knowledge base is maintained offline by distilling confirmed cases into reusable patterns. Proposed updates are reviewed against attack examples and benign counterexamples before incorporation. Its version remains fixed within each evaluation, separating knowledge-base maintenance from the assessment of individual packages.

3.3 Evidence-Guided Iterative Investigation

The investigator extends localized screening into package-level behavioral reasoning. It receives a package overview, available instruction and source content, and the retained code candidates. At each step, it chooses whether to read a file, search package content, or inspect the file structure, using accumulated observations to determine what to examine next. Code candidates provide starting points rather than boundaries on exploration. For example, an instruction may reference a helper script whose configuration reveals the destination of collected data or the conditions under which an operation occurs. Following these references allows the investigator to connect distributed evidence and identify behavior beyond the initial pattern matches. Investigation ends when the model submits its findings or exhausts a fixed step budget. Each finding describes a suspected behavior and cites supporting source evidence.

The submitted findings undergo adversarial review. Where code screening reviews individual pattern matches, this stage challenges the investigator's broader behavioral claims. The reviewer examines whether the observations support the alleged attack, whether surrounding context changes their meaning, and whether a benign explanation refutes the interpretation. A dangerous command quoted in a warning, for example, should be distinguished from an instruction to execute it. Explicitly refuted findings are removed rather than merely assigned lower confidence. Separating investigation from this challenge stage allows suspicious interpretations to be tested against the evidence rather than accepted solely because the investigator proposed them.

The remaining findings undergo source-based verification. Quoted material is checked against package content, and concrete indicators such as addresses, paths, or commands provide additional source anchors. These checks complement adversarial review: contextual reasoning evaluates what the evidence means, whereas source verification checks whether the cited support occurs in the artifact. Neither a matching quotation nor a concrete indicator alone establishes maliciousness. Findings are subsequently separated into attack findings and governance concerns. Only the attack findings contribute to the investigator's detection signal; governance concerns retain useful policy or configuration observations without independently triggering an attack verdict.

3.4 Complementary Semantic Checks

Instruction-level assessment concerns both how an agent is directed and what it is directed to do. An instruction may attempt to override the agent's priorities without specifying a concrete harmful operation, while an apparently ordinary request may prescribe harmful behavior without recognizable injection language. We therefore apply two independent model calls to the complete text of `SKILL.md`, each using a dedicated prompt. The **injection check** identifies priority overrides, unauthorized role changes, and other forms of behavioral hijacking. It distinguishes directives addressed to the agent from quotations, examples, and descriptions of attacks. The **consequence check** examines what following the instructions could induce, using predefined high-impact effect categories to guide assessment. A high-impact action is not treated as malicious solely because of its capability; its interpretation depends on the stated task, authorization context, and evidence of harmful or manipulative intent. The extracted effects describe potential consequences, not observed execution.

The two checks provide separate signals to the final decision process. The injection check can recognize behavioral manipulation without identifying a concrete downstream effect, while the consequence check can identify harmful requested behavior without detecting an instruction-hijacking mechanism. Their outputs distinguish strong attack signals from weaker indications, which are combined with the investigator's signal through the strength-aware decision rule described in Sec-

Table 1: Detection performance on MalSkillBench and MaliciousSkillBench. All values are percentages (%), and higher is better. Bold denotes the best value for each metric within each model-backbone group and dataset.

Model	Method	MalSkillBench				MaliciousSkillBench			
		Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Qwen 2.5-72B	AIG	67.50	89.33	39.75	55.02	68.75	86.41	44.50	58.75
	SkillGate	73.38	**98.45**	47.50	64.08	64.25	**95.24**	30.00	45.63
	Cisco-LLM	69.88	78.91	54.25	64.30	62.00	81.58	31.00	44.93
	Sentry-LLM	73.13	84.91	56.25	67.67	63.00	90.63	29.00	43.94
	CARVE	**88.88**	87.29	**91.00**	**89.11**	**83.00**	90.24	**74.00**	**81.32**
GLM-5.3	AIG	97.00	98.96	95.00	96.94	90.00	96.51	83.00	89.25
	SkillGate	75.75	**99.52**	51.75	68.09	67.00	**98.57**	34.50	51.11
	Cisco-LLM	74.63	81.27	64.00	71.61	82.00	90.00	72.00	80.00
	Sentry-LLM	90.63	94.52	86.25	90.20	**90.25**	93.99	**86.00**	**89.82**
	CARVE	**98.13**	99.23	**97.00**	**98.10**	90.00	94.94	84.50	89.42
Qwen 2.5-7B	AIG	51.75	67.50	6.75	12.27	57.00	**96.67**	14.50	25.22
	SkillGate	**68.63**	**89.42**	42.25	57.39	63.50	87.50	31.50	46.32
	Cisco-LLM	68.38	77.95	51.25	61.84	65.00	72.73	48.00	57.83
	Sentry-LLM	62.13	67.51	46.75	55.24	57.25	64.08	33.00	43.56
	CARVE	67.63	64.97	**76.50**	**70.26**	**67.75**	67.66	**68.00**	**67.83**
Rule-based	Cisco-Static	68.75	78.63	51.50	62.24	57.50	78.85	20.50	32.54
	Sentry-Static	59.00	71.18	30.25	42.46	48.00	10.00	0.50	0.95
	SkillScan-K	53.75	53.47	57.75	55.53	50.25	50.22	56.50	53.18
	SkillScan-N-L	51.25	51.22	52.50	51.85	60.50	60.40	61.00	60.70
	SkillScan-N-S	60.25	67.83	39.00	49.52	54.50	59.78	27.50	37.67
	SSS	50.00	50.00	100.00	66.67	58.25	56.73	69.50	62.47

tion 3.1. Both checks operate directly on the full instruction text, rather than on the investigator's summary or selected evidence. They therefore preserve instruction-level coverage independently of how the investigation budget is allocated, while code screening and iterative investigation examine implementation details and cross-file relationships.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate skill-level maliciousness detection on two benchmarks using the same input representation and assessment interface. For **MalSkillBench** (Guo et al., 2026b), we use a fixed, stratified sample of 800 instances, comprising 400 benign and 400 malicious samples. The malicious portion contains 100 randomly sampled instances from each of four dataset strata: prompt injection (PI), code injection (CI), mixed attacks (MIXED), and malicious samples without these attack-type tags (NONE). For **MaliciousSkillBench** (Wang et al., 2026), we use 200 malicious and 200 benign instances from the benchmark evaluation set, which includes SkillHarm (Ning et al., 2026), ATR Skill Security Benchmark (Agent-Threat-Rule, 2026), and SkillFortifyBench (Bhardwaj, 2026). Both benchmarks are normalized to the same `SKILL.md`-centered input representation; the benchmark distinction is in the sampled instances and labels, not in the input format or evaluation interface. We randomly sampled 200 malicious samples and 200 benign samples from MaliciousSkillBench.

Baselines. We compare against rule-based scanners and LLM-based assessment methods, using concise names throughout the evaluation. The rule-based group includes *Cisco-Static*, the static configuration of Cisco Skill Scanner (Cisco Systems, 2025); *Sentry-Static*, the static scanner from Sentry (Sentry, 2025); *SkillScan-K*, the SkillScan Security tool by KurtPayne (Payne, 2025); *SkillScan-N*, the SkillScan tool by NMitchem (NMitchem, 2025); and *SSS*, short for Skill-Security-

Scanner (huifer). We distinguish the loose and strict decision settings of SkillScan-N as *SkillScan-N-L* and *SkillScan-N-S*. The LLM-based group includes *AIG*, short for AI-Infra-Guard (Tencent EdgeOne / Zhuque Lab, 2025); *SkillGate*, evaluated in hybrid mode (Yang et al., 2026b); *Cisco-LLM*, Cisco Skill Scanner with LLM analysis (Cisco Systems, 2025); and *Sentry-LLM*, our LLM-assisted adaptation of Sentry Skill Scanner (Sentry, 2025). Sentry-LLM combines the tool's static scan results, scanning instructions, and reference materials for model-based assessment. Each baseline is evaluated through its documented implementation and configured operating procedure. CARVE's multiple calls are part of its proposed multi-view decision rule, rather than an externally imposed budget adjustment; the comparison therefore measures complete system behavior under the respective native protocols.

Models and Metrics. Our primary evaluation uses Qwen2.5-72B-Instruct (Team, 2024) and GLM-5.3 (Z.AI), with Qwen2.5-72B-Instruct as the central deployment setting. Its open weights and support for local inference make it a practical backbone for self-hosted skill assessment, allowing package contents to remain within the controlled environment. We additionally evaluate Qwen2.5-7B-Instruct (Team, 2024) to explore resource-constrained deployment, and include Claude Opus 5 (Anthropic) and GPT-5.6 (OpenAI) as supplementary closed-model backbones to assess applicability across model families. For all comparisons, we treat malicious skills as the positive class and report accuracy, precision, recall, and F1. Accuracy measures overall classification correctness, precision and recall capture the reliability and coverage of attack detection, and F1 is their harmonic mean.

4.2 Detection Performance

Table 1 shows that CARVE achieves the highest F1 within all three backbone groups on MalSkillBench. With Qwen2.5-72B, GLM-5.3, and Qwen2.5-7B, CARVE obtains 89.11%, 98.10%, and 70.26% F1, outperforming the strongest same-backbone baselines by 21.44, 1.16, and 8.42 percentage points, respectively. These gains are accompanied by the highest recall in each group. This pattern is consistent with CARVE's complementary analysis of instructions, injection attempts, and behavioral effects: reconciling observed capabilities with declared purpose helps identify attacks whose individual operations appear benign. On MaliciousSkillBench, CARVE leads the Qwen2.5-72B and Qwen2.5-7B groups with 81.32% and 67.83% F1, respectively. With GLM-5.3, it achieves 89.42% F1, slightly below Sentry-LLM's 89.82%, but offers higher precision (94.94% versus 93.99%) at slightly lower recall (84.50% versus 86.00%). This trade-off is consistent with CARVE's requirement for concrete evidence of malicious behavior, which favors fewer false alarms but can miss less explicit attacks. The balance also varies across backbones: on MalSkillBench, CARVE-7B achieves 76.50% recall but only 64.97% precision, indicating that distinguishing suspicious capabilities from legitimate functionality remains challenging for the smaller model. CARVE also exceeds all evaluated rule-based methods in F1 on both datasets. Notably, SSS attains 100.00% recall on MalSkillBench only by predicting every sample as malicious, yielding 50.00% precision. In contrast, CARVE's contextual assessment offers a better balance between attack coverage and false alarms, supporting the value of reasoning about what a skill does and whether that behavior is justified by its intended purpose.

Closed-source backbone transfer. To assess whether CARVE transfers beyond locally deployable open-weight backbones, we additionally evaluate Claude Opus 5 and GPT-5.6 Sol on the same fixed evaluation splits. As shown in Table 2, Claude Opus 5 achieves 98.13% and 94.18% F1 on MalSkillBench and MaliciousSkillBench, respectively. GPT-5.6 Sol obtains 93.95% F1 on MalSkillBench and 81.84% on MaliciousSkillBench.

Table 2: Closed-source Backbone.

Model	Acc.	Prec.	Rec.	F1
MalSkillBench				
Opus 5	98.13	98.00	98.25	98.13
GPT-5.6	94.00	94.67	93.25	93.95
MaliciousSkillBench				
Opus 5	94.25	95.38	93.00	94.18
GPT-5.6	83.25	89.35	75.50	81.84

4.3 Ablation Study

Settings. We use GLM-5.3 as a representative backbone to assess the contributions of CARVE's three modules: Code screening, Investigation, and Semantic. We evaluate all configurations on the same 800 MalSkillBench samples and 400 MaliciousSkillBench samples, with balanced malicious and benign classes and complete sample coverage. We consider three variants: *w/o Code screening* disables deterministic code screening and its downstream hints while preserving access to package

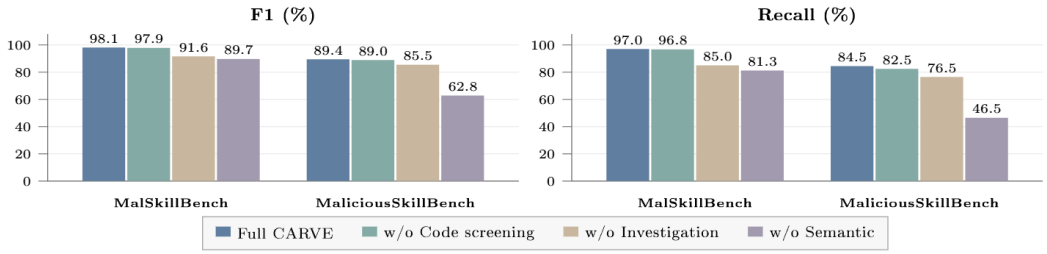


Figure 3: Module ablation results on MalSkillBench and MaliciousSkillBench.

contents; *w/o Investigation* removes the investigation module; and *w/o Semantic* disables both the Injection and Effects judges. The ablated configurations are compared against the corresponding Full CARVE outputs. Each comparison uses the same samples, labels, evaluation protocol, and designated implementation for the configuration being tested. The purpose is to measure the practical contribution of disabling each module while keeping the rest of the evaluation pipeline unchanged.

Results and analysis. As shown in Figure 3, Full CARVE achieves the highest F1 and recall among the evaluated configurations on both datasets. Removing Investigation reduces F1 by 6.46 percentage points (pp) on MalSkillBench and 3.94 pp on MaliciousSkillBench, accompanied by recall decreases of 12.00 and 8.00 pp, respectively. The largest observed degradation occurs without Semantic: F1 decreases by 8.45 and 26.58 pp, while recall drops by 15.75 and 38.00 pp. Precision nevertheless remains at 100.00% and 96.88%, indicating that these losses primarily reflect missed malicious samples rather than additional false positives. Removing Code screening produces smaller F1 decreases of 0.25 and 0.47 pp. Since Investigation retains access to package contents in this variant, these differences measure the incremental role of explicit screening and its hints, rather than the importance of code content itself. Overall, the MalSkillBench results support the contributions of Investigation and Semantic to detection coverage, with a smaller increment from Code screening. MaliciousSkillBench exhibits the same ordering of observed F1 gaps, supporting the practical contribution of the three modules across both benchmark settings.

4.4 Cost and Effectiveness at Scale

We measure detection quality and end-to-end token usage on the 400 MaliciousSkillBench instances used for the cost evaluation, containing 200 malicious and 200 benign skills. We compare CARVE with AIG, Cisco's skill scanner, and Sentry's skill scanner on two model backbones: the local Qwen2.5-72B-Instruct and the GLM-5.3 reasoning model. Measured token volume is used as a cost proxy, capturing inference work for the local model and providing the principal quantity for provider-specific API pricing. As shown in Figure 4, CARVE reaches the highest measured recall and F1 on both backbones in this cost evaluation. On Qwen2.5-72B, CARVE reaches 69.5% recall and 77.9 F1, compared with 42.0%/53.5 for AIG, 42.5%/56.9 for Cisco, and 28.0%/42.6 for Sentry. Thus, CARVE improves recall by 1.6× over the strongest single-pass baseline and by 2.5× over Sentry. In the token-usage–recall plane, AIG and Cisco form the low-cost frontier near 42% recall, while Sentry is dominated by cheaper and more accurate alternatives; CARVE extends the measured frontier into the high-coverage region at 69.5% recall. The same ordering persists with GLM-5.3: CARVE obtains the highest F1 (92.0), ahead of Sentry (90.4), AIG (87.8), and Cisco (81.1), although the gap is naturally smaller for the stronger model.

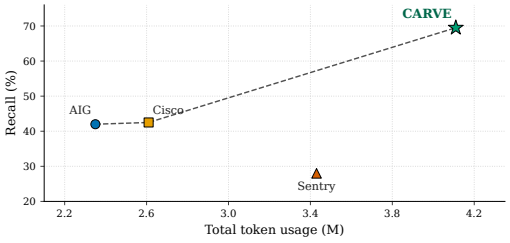


Figure 4: Token-usage–recall frontier on the MaliciousSkillBench cost evaluation with Qwen2.5-72B. Lower token volume and higher recall are preferred; the dashed line connects measured non-dominated points.

The two-backbone comparison therefore supports the central coverage claim: as model capability decreases, single-pass recall collapses from 81/74/87% on GLM-5.3 to 42/43/28% on Qwen2.5-

72B, whereas CARVE decreases only from 89.0% to 69.5% and remains the recall–F1 leader. CARVE consequently occupies the high-recall end of the measured token–effectiveness frontier. Its additional multi-view computation is associated with a substantial increase in malicious-skill coverage, rather than merely a larger raw workload.

4.5 Case Analysis

Figure 2 presents two representative cases from the formal E1 evaluation. The first, `codecast`, is labeled benign and describes streaming coding agent sessions to Discord. CARVE does not route it as an attack: the injection check is negative, the consequence signal is weak, and no strong package-level attack finding is produced. Nevertheless, CARVE retains governance concerns related to broad execution authority and possible session-context exposure, thereby separating governance reporting from malicious attack detection. The second case, `documenting-rust-code`, presents itself as a Rust documentation helper but later introduces persistence configuration, automatic restart behavior, and a persistence workflow unrelated to its declared purpose. The injection check identifies this mismatch as a strong hidden directive, while the consequence check remains silent. Because a strong injection signal is sufficient under CARVE's fusion rule, the skill is correctly classified as malicious, while any governance findings remain separately auditable. Together, these cases illustrate how CARVE combines complementary signals without requiring every analysis channel to fire.

4.6 Error Analysis

The remaining errors in the main experiment reveal several recurring sources of difficulty. False negatives are concentrated in malicious skills whose attack semantics are concealed by an apparently benign task description. They can be broadly grouped into three categories. First, some attacks are embedded in ordinary development, testing, or documentation workflows, making the malicious behavior difficult to distinguish from the skill's stated purpose. vocabulary. Third, some malicious skills are presented as ordinary utilities: network access, script execution, or permission changes are integrated into a seemingly legitimate tool workflow, while the malicious intent remains implicit. All false negatives for both models correspond to complete silence from the three views: GLM produces 3 false negatives and 72B produces 23. This suggests that the main limitation is insufficient coverage of concealed attack semantics, rather than disagreement among views during fusion.

False positives arise from a different form of ambiguity: benign skills often expose capabilities or use language that overlaps with malicious behavior. One recurring case involves capability-rich skills that legitimately perform network requests, access credentials, manage wallets, or maintain scheduled activities. Their observable behavior resembles attack-relevant operations such as data access, persistence, or exfiltration. A second case involves instructional and configuration-oriented skills whose imperative language, hidden directives, or security-related examples resemble prompt injection or tool poisoning, despite serving a benign purpose. This source of noise is particularly evident in the structured view. The view is involved in 12 of GLM's 15 false positives and 80 of 72B's 96 false positives; for 72B, 67 false positives are triggered only by the structured view. Overall, GLM's residual errors are primarily caused by semantically concealed attacks, whereas 72B additionally produces substantial false-positive noise on benign skills with broad capabilities or attack-like surface language.

5 Conclusion

Agent-skill detection must jointly analyze natural-language instructions and executable code. CARVE addresses this challenge by combining code screening, package-level investigation, and independent semantic checks, then fusing typed signals with source-linked evidence while reporting governance concerns separately. Across MalSkillBench, CARVE achieves the best F1 and recall across all three evaluated open-weight model scales. On MaliciousSkillBench, it leads the 7B and 72B groups and remains competitive with the largest model. Ablations show that all three modules contribute to detection performance, with semantic checks and investigation providing the largest gains. The results support evidence-grounded, coverage-oriented assessment for locally deployable malicious-skill detection.

AI USE STATEMENT

AI-based tools were used to assist with language editing, code and experiment-artifact inspection, and preparation of the graphical assets.

ETHICS STATEMENT

This work studies defensive detection of malicious agent skills and does not involve human subjects, personal data collection, or new data release. The evaluation uses existing benchmark artifacts and analyzes skill packages without executing their code. Because the topic concerns potentially harmful behavior, examples are reported only as needed to support detection and reproducibility; operational details that would facilitate abuse are not introduced beyond the evaluated benchmark content. The method is intended for installation-time screening, auditing, and research evaluation, and should be combined with appropriate authorization and sandboxing in deployment.

REPRODUCIBILITY STATEMENT

The main file contains the complete method description, experimental protocol, result tables, formal decision rule, algorithmic summary, and statistical reporting. The evaluation uses fixed benchmark samples, shipped prompts and rules, and the model configurations described in the experimental setup. Anonymous source code, configuration files, figure assets, and reproduction instructions will be provided as supplementary material; per-sample outputs and configuration records are provided where permitted by the benchmark and provider terms.

REFERENCES

Agent-Threat-Rule. ATR Skill Security Benchmark, 2026. Dataset source record SRC011; Hugging Face: atr-skill-benchmark.

Anthropic. Claude Opus 5. `https://platform.claude.com/docs/en/models/opus-5/overview`. Official model documentation. Accessed September 2026.

Sanket Badhe and Priyanka Tiwari. Agent Skill Security: Threat Models, Attacks, Defenses, and Evaluation, 2026. arXiv:2607.13987.

Varun Pratap Bhardwaj. Formal Analysis and Supply Chain Security for Agentic AI Skills, 2026. SkillFortifyBench source record (SRC012); arXiv:2603.00195.

Xinze Chen, Chi Zhang, Ping Ji, and Yimin Liu. SkillsMetric: Mapping the Detection Boundary of Static Analysis for Malicious Agent Skills, 2026. arXiv:2608.08468.

Cisco Systems. Cisco skill scanner (llm multi-analyzer). Public repository, 2025.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, Marc Dolz-Garcia, et al. AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. In *Advances in Neural Information Processing Systems (NeurIPS Datasets and Benchmarks)*, 2024. arXiv:2406.13352.

Blaise Delattre, Cong Wang, and Yang Cao. CAGE: Certified Authorization under Typed-Return Uncertainty for Tool-Using Agents, 2026. arXiv:2607.29190.

Francisco Eiras, Eliott Zemour, Eric Lin, and Vaikkunth Mugunthan. Know Thy Judge: On the Robustness Meta-Evaluation of LLM Safety Judges, 2025. arXiv:2503.04474.

Bacem Etteib, Daniele Lunghi, and Tégawendé F. Bissyandé. Detecting Malicious Agent Skills in the Wild using Attention, 2026. arXiv:2606.23416.

Yang Gao. How Reliable Is Your Jailbreak Judge? Calibration and Adversarial Robustness of Automated ASR Scoring, 2026. arXiv:2606.25487.

Google. Virustotal: Per-file malware aggregation service. https://www.virustotal.com/, 2025.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yuling Shen, Shijie Han, Zhiyuan Liu, et al. A Survey on LLM-as-a-Judge, 2024. arXiv:2411.15594.

Wenbo Guo, Zhongwen Chen, Zhengzi Xu, Chengwei Liu, Ming Kang, Shiwen Song, Chengyue Liu, Yijia Xu, Weisong Sun, and Yang Liu. How Effective Are NPM Malicious Package Detectors? A Large-Scale Empirical Study, 2026a. arXiv:2603.27549.

Wenbo Guo, Wei Zeng, Chengwei Liu, Xiaojun Jia, Yijia Xu, Lei Tang, Yong Fang, and Yang Liu. MalSkillBench: A Runtime-Verified Benchmark of Malicious Agent Skills, 2026b. arXiv:2606.07131.

Ismail Hossain, Sai Puppala, Md Jahangir Alam, Tanzim Ahad, and Sajedul Talukder. SkillVetBench: LLM-as-Judge for Multi-Dimensional Security Risk Evaluation in Open-Source LLM Agent Skills, 2026. arXiv:2606.15899.

Yinghan Hou and Zongyou Yang. SkillSieve: A Hierarchical Triage Framework for Detecting Malicious AI Agent Skills, 2026. arXiv:2604.06550.

huifer. Skill-Security-Scanner. https://github.com/huifer/skill-security-scan. Software repository. Accessed September 2026.

Zimo Ji, Congying Xu, Zongjie Li, Yudong Gao, Xin Wei, Shuai Wang, and Shing-Chi Cheung. Cloak and Detonate: Scanner Evasion and Dynamic Detection of Agent Skill Malware, 2026. arXiv:2607.02357.

Vincent Koc, Patrick Erichsen, Jacob Tomlinson, Agustin Rivera, Michael Appel, and Nir Paz. ClawHub Security Signals: When VirusTotal, Static Analysis, and SkillSpector Disagree, 2026. arXiv:2606.01494.

Nizhang Li, Zonghao Ying, Xiangfan Wu, Zonglei Jing, Xixun Lin, Hao Zhang, Wenxin Zhang, Jiaye Lin, Quanchen Zou, and Xiangzheng Zhang. SkillSentry: Adaptive Honey Worlds for Dynamic Safety Testing of Agent Skills, 2026a. arXiv:2608.03485.

Ying Li, Yanju Chen, Hongbo Wen, Bosi Zhang, Hanzhi Liu, Peiran Wang, Yu Feng, and Yuan Tian. VIGIL: Runtime Enforcement of Behavioral Specifications in AI Agent Skills, 2026b. arXiv:2606.26524.

Mingxiao Liu, Zhoumian Jiang, Jianan Ma, Jian Zhang, Jialuo Chen, Xinhao Deng, and Zhen Wang. CompoSkill: Compositional Skill Chain Attacks from Individually Scanner-Passing LLM Agent Skills, 2026a. arXiv:2608.16246.

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

Rahul Marchand, Art O Cathain, Jerome Wynne, Philippos Maximos Giavridis, Stuart Jennings, Freddy Tuxworth, Tolga H. Dur, Sam Deverett, John Wilkinson, Jason Gwartz, and Harry Coppock. Quantifying Frontier LLM Capabilities for Container Sandbox Escape, 2026. arXiv:2603.02277.

Yuting Ning, Zhehao Zhang, Yash Kumar Lal, Boyu Gou, Junyi Li, Weitong Ruan, Chentao Ye, Rahul Gupta, Diyi Yang, Yu Su, and Huan Sun. SkillHarm: Lifecycle-Aware Skill-Based Attacks via Automated Construction, 2026. SkillHarm source record (SRC009); arXiv:2606.02540.

NMitchem. Nmitchem-skillscan (loose/strict configurations). Public repository, 2025.

OpenAI. Models: OpenAI API. https://developers.openai.com/api/docs/models. Official model catalog. Accessed September 2026.

Shidong Pan, Xiaoyu Sun, Tianyi Zhang, Dianshu Liao, Kaiwen Yang, and Zhenchang Xing. SkillGuard: A Permission-Centric Framework for Agent Skill Security, 2026. arXiv:2606.03024.

Kurt Payne. kurtpayne-skillscan-security. Public repository, 2025.

Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, 2019. arXiv:1807.07838.

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

Sentry. getsentry skill-scanner (sentry skill scanner, full mode). Public repository, 2025.

Qwen Team. Qwen2.5 technical report. *arXiv:2412.15115*, 2024.

Tencent EdgeOne / Zhuque Lab. Ai-infra-guard (edgeone-skill-scanner): Single-pass llm skill security scanner. Public repository, `https://github.com/Tencent/AI-Infra-Guard`, 2025.

Yingzhe Tong, Leyu Dai, and Songhui Guo. AID-Guard: Stateful Authorization for Delegated Agent Effects, 2026. arXiv:2608.21159.

Yue Wang, Yi Liu, Gelei Deng, Ying Zhang, Yuekang Li, Zhenyu Chen, and Leo Zhang. MaliciousSkillBench: A Comprehensive Benchmark for Malicious Agent Skill Detection, 2026. arXiv:2608.19901.

Yuhao Wu, Tung-Ling Li, and Hongliang Liu. Behavioral Integrity Verification for AI Agent Skills, 2026. arXiv:2605.11770.

Wenjie Xiao, Xuehai Tang, Biyu Zhou, Songlin Hu, and Jizhong Han. RouteGuard: Internal-Signal Detection of Skill Poisoning in LLM Agents, 2026. arXiv:2604.22888.

Rui Yang, Michael Fu, Kla Tantithamthavorn, Chetan Arora, and Joey Chua. Towards a Risk Assessment of Malicious Skill Files in Coding Agents, 2026a. arXiv:2608.05223.

Rui Yang, Michael Fu, Kla Tantithamthavorn, Chetan Arora, and Joey Chua. SkillGate: Cost Efficient Runtime Malicious Skill File Detection in Coding Agents, 2026b. arXiv:2607.25619.

Hengkai Ye, Zhechang Zhang, Jinyuan Jia, and Hong Hu. TRUSTDESC: Preventing Tool Poisoning in LLM Applications via Trusted Description Generation, 2026. arXiv:2604.07536.

Z.AI. GLM-5.3: Overview. `https://docs.z.ai/guides/llm/glm-5.3`. Official model documentation. Accessed September 2026.

Puyu Zeng, Simeng Qin, Jingzhi Li, Ju Jia, Zheli Liu, and Xiaojun Jia. ColluSkill: Adversarial Cross-Skill Composition for Evading Agent Skill Scanners, 2026. arXiv:2608.09732.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. arXiv:2403.02691.

Chengjun Zhang, Yang Gao, Jianna Hur, Jingjing Zhang, and Sagar Samtani. Cross-Layer Misalignment Detection in Agent Skills: A Progressive Loading-Aware Contrastive Learning Approach, 2026. arXiv:2607.10534.

Yulai Zhao, Haolin Liu, Dian Yu, Sunyuan Kung, Meijia Chen, Haitao Mi, and Dong Yu. One Token to Fool LLM-as-a-Judge, 2025. arXiv:2507.08794.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2306.05685.

A Formalization and Decision Rule

We model a skill package as $S = (T, C, P)$, where T is the natural-language instruction content, C is the bundled implementation, and P is the declared capability or permission information. CARVE uses three complementary views:

$$v_1(S) \to F, \qquad v_2(T) \to a, \qquad v_3(T) \to E,$$

where F is a set of structured findings, a is an instruction-level assessment, and E is a set of consequence tuples. A finding is represented as $f = (r, \ell, s, c, e)$, where r is a rule identifier, ℓ is either the attack or governance channel, s is severity, c is confidence, and e is source-linked evidence. Only findings in the attack channel can directly contribute to a malicious verdict.

The views do not emit interchangeable scalar scores. Each view can instead emit an attack signal, a weaker indication, or an abstention. Let A denote the presence of a strong attack signal and let W denote the set of available weak signals from the three views. At the paper level, a strong signal is a retained attack-channel finding with confirmed support, an injection assessment that explicitly identifies malicious instruction behavior, or a consequence assessment that extracts a high-impact effect with supporting evidence. Needs-review findings and low-confidence semantic indications are treated as weak signals. Unavailable, malformed, or explicitly abstaining views contribute no signal. The exact category names and thresholds are fixed by the shipped prompts and evaluation configuration; the abstraction below states their common decision semantics.

The coverage-first decision rule is

$$d(S) = \begin{cases} 1, & A = 1, \\ 1, & A = 0 \ \wedge \ |W| \geq 2, \\ 0, & \text{otherwise.} \end{cases}$$

Governance-only findings are retained in the report but do not enter $d(S)$. If a view explicitly refuses because the artifact is unsafe, the refusal is converted into an attack signal; if a view is unavailable or returns an invalid structured result, the view abstains and the reliability field is marked degraded. Each output is therefore represented as

$$R(S) = (d(S),\, \mathcal{E}(S),\, q(S)),$$

where $\mathcal{E}(S)$ contains the retained evidence and channel provenance, and $q(S) \in \{\texttt{ok}, \texttt{degraded}\}$ records whether all required views completed. This formalization captures the decision semantics without depending on a particular model provider or implementation language.

B Algorithmic Summary

```
Input: skill package S=(T,C,P)
1. Screen code and collect source-linked evidence.
2. Investigate relevant files and references.
3. Review and source-check investigator findings.
4. Classify instruction-level injection in T.
5. Extract potential high-impact effects from T.
6. Fuse typed signals and filter governance findings.
7. Return decision, evidence, and reliability.
```

The algorithm is non-executing: consequence signals describe what following the instructions could induce and are not treated as observations of runtime behavior. The fixed prompts, rules, and evaluation configurations determine the operating point reported in the paper.

C Statistical Reporting

For the main recall results, we report descriptive two-sided 95% Wilson intervals for the malicious subset. These intervals quantify binomial sampling uncertainty for each fixed operating point; they do not account for run-to-run variation of stochastic LLM judgments and are not a substitute for paired tests between systems.

View	Strong signal	Weak or abstaining signal
Structured	confirmed attack-channel finding with evidence	needs-review finding; invalid output or timeout abstains
Injection	explicit malicious instruction judgment with evidence	suspicious or low-confidence judgment; invalid output abstains
Consequence	high-impact effect extracted with supporting evidence	ambiguous effect or empty result; invalid output abstains

Table 3: Paper-level mapping from view outputs to the abstract signal categories used by the fusion rule. Exact labels are fixed by the shipped prompts and configuration.

Dataset	Backbone	Method	Recall	95% Wilson interval
MalSkillBench	Qwen2.5-72B	CARVE	91.0	[87.8, 93.4]
MalSkillBench	Qwen2.5-72B	Sentry-LLM	56.3	[51.4, 61.0]
MalSkillBench	GLM-5.3	CARVE	97.0	[94.8, 98.3]
MalSkillBench	GLM-5.3	AIG	95.0	[92.4, 96.7]
MalSkillBench	Qwen2.5-7B	CARVE	76.5	[72.1, 80.4]
MalSkillBench	Qwen2.5-7B	Cisco-LLM	51.3	[46.4, 56.1]
MaliciousSkillBench	Qwen2.5-72B	CARVE	74.0	[67.5, 79.6]
MaliciousSkillBench	Qwen2.5-72B	AIG	44.5	[37.8, 51.4]
MaliciousSkillBench	GLM-5.3	CARVE	84.5	[78.8, 88.9]
MaliciousSkillBench	GLM-5.3	Sentry-LLM	86.0	[80.5, 90.1]
MaliciousSkillBench	Qwen2.5-7B	CARVE	68.0	[61.2, 74.1]
MaliciousSkillBench	Qwen2.5-7B	Cisco-LLM	48.0	[41.2, 54.9]

Table 4: Descriptive 95% Wilson intervals for CARVE recall. The MalSkillBench intervals use 400 malicious instances and the MaliciousSkillBench intervals use 200 malicious instances. For each dataset and backbone, the second row reports the strongest LLM baseline by recall in the main results table.