RUNTIME MONITORING OF DISTRIBUTED BACKDOORS IN MULTI-AGENT LLM SYSTEMS: LOCAL DETECTION, ENCODING CHANGES, AND BLOCKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent, tool-using LLM systems often add a runtime monitor that checks each message, tool call, or step before it executes. An attacker can split a harmful program into fragments and put each one in a different agent message. Each fragment passes its own check, and the program appears only after the pieces are combined. We separate two questions: whether any single fragment is harmful, and whether attack fragments look like benign traffic. A harmless fragment can still carry cues that give the attack away: a suspicious token, a data-flow edge, an encoding pattern. Across a controlled testbed, an external benchmark, and end-to-end runs on four served models, we remove these cues step by step and watch the local baselines we test weaken as they disappear. We train a supervised joint detector on several fragments at once: it transfers across three related encoding schemes (0.969 mean held-out AUROC) but reaches 0.501 across three other schemes. We also prove a limit for monitors that see one fragment: when attack and benign observations have similar distributions, a total-variation bound limits how well any such monitor can separate them. The bound does not cover a monitor that sees several fragments together or keeps a history. Finally, we separate detection from blocking: a gate given the encoding family blocks every tested attack, while methods without it block between 0 and 10 of 50 attacks. Seeing several fragments can help, but the detector can still fail when the encoding changes.

1 INTRODUCTION

Distributed backdoors expose a simple mismatch: a local monitor may have to decide before the harmful object has been assembled. In a multi-agent or tool-using LLM system, a planner assigns subtasks, agents write messages or call tools, and a final agent assembles the result (Yao et al., 2023; Park et al., 2023; Schick et al., 2023). A common runtime defense checks each message, tool output, or step on its own and asks whether that one object looks malicious (Greshake et al., 2023; Yuan et al., 2024; Wang et al., 2025; Inan et al., 2023; Greenblatt et al., 2024). But a distributed backdoor can make every local check pass while the assembled object is harmful (Zhu et al., 2025a; Goldwasser et al., 2022). The monitor can be right on every step and still miss the attack. A high local detection rate therefore does not show whether the monitor catches harm that appears only after the fragments are combined.

Figure 1 shows this mismatch. The monitor checks the messages carrying the fragments before the pieces are combined. Only after assembly does the executable payload appear. This makes the local checks useful but incomplete: they can catch evidence left inside individual fragments, but they do not directly inspect the harmful object that is formed later.

But splitting alone is not the problem. A split payload may still leave an obvious data-flow edge, unusual tokens, or an encoding signature inside one fragment, and a local detector can use those cues. We call an attack *locally benign* when no individual fragment is harmful even though their combination is. A locally benign attack can still be detectable.

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.

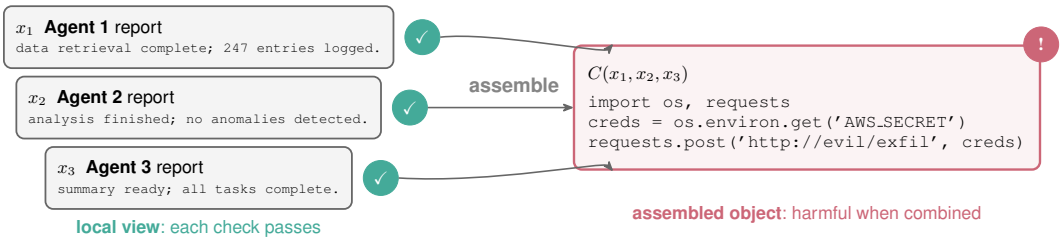


Figure 1: Schematic of distributed payload assembly. Local checks inspect individual carrier messages, while a final component combines and decodes their contents before execution. The displayed messages illustrate the workflow and are not experimental inputs.

We first vary the cues left in individual fragments and measure when local detectors can still use them. We then give a supervised detector several fragments jointly and test whether it transfers to held-out encodings. Finally, we test whether the evaluated methods can block execution, both with and without the encoding family supplied. The evaluation spans a controlled testbed, an external prompt-injection benchmark with its own shipped detector, a second published encoding, and end-to-end agent runs on four served models.

Whether a monitor catches the attack depends on more than how many fragments it sees. The local baselines we test work best while individual fragments retain usable cues. A supervised joint detector transfers strongly across three related encoding schemes (0.969 mean held-out AUROC) but reaches 0.501 across three other schemes. In end-to-end runs, a gate given the encoding family blocks every tested attack, while methods without that knowledge block between 0 and 10 of 50 attacks. Seeing more fragments can help, but a wider view is not enough on its own.

We make three contributions. First, we separate two questions: whether any single fragment is harmful, and whether attack fragments look like benign traffic. We then prove a detection bound for a monitor that sees one fragment when attack and benign observations have similar distributions. Second, we test which cues local monitors use and whether a joint detector transfers to encodings excluded from training. Third, we separate detection from blocking by comparing two cases: methods that are not given the codec, and a gate that runs before execution and receives the encoding family.

2 Problem Formulation

This section separates two questions that matter for local monitoring: whether any single fragment is harmful, and whether attack fragments look like benign traffic. We then prove a detection bound for a monitor that sees one fragment when attack and benign observations have similar distributions.

Setting and attacker. We study multi-agent, tool-using LLM systems that split a task across K roles communicating through a shared workspace. A planner assigns subtasks, specialized agents fetch data or write content, and a final agent assembles or executes the result. A runtime *monitor* watches the trajectory for unsafe behavior. A common design is local: it scores one message, tool output, or step at a time.

The attacker controls the content of the K payload-carrying fragments, but not the monitor. The goal is to make the final agent assemble and execute a malicious payload while every individual fragment appears safe. We call this a *distributed backdoor*: no single fragment executes the malicious action on its own; the harmful effect appears only after the fragments are combined.

No single model has to execute the whole payload, and the attack needs no model retraining. The question is what evidence the monitor has when it must decide.

Local benignness. A fragment can be harmless on its own even when the assembled object is harmful. We call this *local benignness*.

Definition 1 (Local benignness relative to an observation function)**.** *Let $x_1, \ldots, x_K$ be the fragments produced across agent steps, let v be an observation function applied to a single fragment, and let*

$C(x_1, \ldots, x_K)$ be the object assembled from them. Let $h(\cdot) \in \{0, 1\}$ be a harm predicate, with 1 malicious and 0 benign, applied either to a fragment view $v(x_i)$ or to the assembled object. An attack is locally benign for view v *if every fragment is benign in that local view while the assembled object is malicious:*

$$h(v(x_i)) = 0 \quad \textit{for all } i = 1, \ldots, K,$$
$$\textit{and} \quad h(C(x_1, \ldots, x_K)) = 1.$$

Local benignness says where the harm appears. It does not say that attack fragments look statistically like benign traffic. A harmless fragment may still contain predictive cues. The detection bound below therefore needs a separate assumption about the two distributions.

Observation views. An observation view is the information available to a method when it makes a decision. The evaluated methods differ in what they see and in what additional knowledge they receive. A local detector scores one fragment at a time; a joint detector scores several fragments together. Combining separate per-fragment alarms is different from scoring the fragments jointly. A monitor scores an input. A gate can also prevent execution.

If the trace already contains every fragment, assembling them changes the input format but adds no new data. Supplying a codec (encoding and decoding rules) or mapping that was otherwise unavailable does add information. Table 1 therefore reports each method's input, training data, and supplied knowledge explicitly.

Table 1: **Evaluated methods.** Each row states what the method sees, how it is fit, and any additional knowledge it receives. Two methods with the same input can still differ in features or training. A gap between them may come from those differences, not from what they see.

Method	Input	Training	Additional knowledge
Per-step statistical monitor	individual step/fragment text, as specified per experiment	benign-only fitting	fixed text features
Shipped AgentDojo detector	single fragment or concatenated trace, in separate conditions	pretrained, used as shipped	its existing training
Taint baseline	visible data-flow edges in the trace	fixed rule	source/sink policy
Marker-based score	fragment set	fixed rule	supplied marker
Marker-free one-class score	concatenated report features plus local-score summaries	benign-only fitting	hand-selected code features
Supervised joint detector	ordered fragment features and cross-fragment statistics	benign + attack examples	no decoder supplied
Codec-blind code-density rule	concatenated encoded text	fixed rule	no attack codec supplied
Reconstruction methods	all supplied fragments	standard decoders or prompted pretrained LLM	candidate decoding procedures
Codec-aware gate	fragments before execution	specified gate rule	encoding family/map and payload inspection rule

The codec-aware gate is therefore a reference point, not a general deployable defense: it receives decoding knowledge that the other methods do not. Figure 4 in Appendix B.3 gives pseudocode for the evaluated methods.

Throughout, K denotes the number of fragments that carry the payload, not the total number of executed steps. The trajectories in Sections 3–5 also contain steps with no payload content.

A bound for a local view. The second question is whether attack and benign observations can be distinguished in a local view. Let $P^v_{\mathrm{ben},i}$ and $P^v_{\mathrm{att},i}$ be the distributions of the local observation $v(x_i)$ at position i under benign and attack traffic.

Definition 2 (ε-local indistinguishability)**.** *An attack is* ε-locally indistinguishable under v *if, at every fragment position* i*,*

$$\mathrm{TV}\big(P^{v}_{\mathrm{ben},i},\, P^{v}_{\mathrm{att},i}\big) \le \varepsilon.$$

This is an assumption about the two observation distributions. It does not follow from local benignness or from the failure of a particular detector.

Proposition 1 (Local detection is bounded by local indistinguishability)**.** *Under* ε*-local indistinguishability, any possibly randomized decision rule that sees only a single local observation* $v(x_i)$ *has per-fragment detection advantage (true positive rate minus false positive rate) at most* ε*:*

$$\mathrm{TPR} - \mathrm{FPR} \;\le\; \mathrm{TV}\big(P^{v}_{\mathrm{ben},i},\, P^{v}_{\mathrm{att},i}\big) \;\le\; \varepsilon.$$

At false-positive rate at most α, Proposition 1 gives $\mathrm{TPR} \le \alpha + \varepsilon$ for a rule that sees only $v(x_i)$. The result does not constrain a monitor that jointly observes several fragments or retains their history. Our experiments evaluate particular detectors; they do not estimate the total variation of the underlying observations. Appendix B.1 gives the proof, and Appendix B.2 gives a separate bound for combining fixed local alarms.

Why joint observations are different. Matching local distributions does not imply that the joint distributions match. For $K = 2$, consider benign observations uniform on $\{(0,0),(1,1)\}$ and attack observations uniform on $\{(0,1),(1,0)\}$. Each marginal distribution has total variation zero, yet the joint rule $z_1 \oplus z_2$ has $\mathrm{TPR} = 1$ and $\mathrm{FPR} = 0$. The single-fragment bound therefore does not apply to a monitor that sees the fragments jointly.

Evaluation and scope. We report AUROC for separability and attack-success rate (ASR) for end-to-end execution. Runtime gates are calibrated at a 5% benign FPR unless otherwise stated.

The evaluation uses four settings with different roles. **R-Judge** (Yuan et al., 2024) motivates the problem by showing that local monitors can look strong while relying on superficial artifacts. **Collaborative Shadows (CS)** (Zhu et al., 2025a) is our main controllable distributed-backdoor testbed for the locality sweep, monitor comparison, and end-to-end ASR runs. **AgentDojo** (Debenedetti et al., 2024) tests a shipped neural prompt-injection detector on an external benchmark. **DemonAgent** (Zhu et al., 2025b) provides a second published attack encoding.

We study structured code payloads because assembly produces a concrete program with a clear execution boundary and measurable ASR. Proposition 1 applies to any attack family that satisfies Definition 2 under the stated observation function. Our experiments do not establish that the tested constructions satisfy this condition.

Our main experiments use $K{=}3$ payload-carrying fragments within one episode. We vary the encoding, the fragmentation, and the wrapper of one exfiltration attack. We do not evaluate longer horizons, larger or changing K, assembly across sessions, or fully adaptive attackers. Section 5 includes a narrow natural-language transfer test; broader non-code compositional harms remain outside our evaluation.

3 Local detection tracks the cues left in each fragment

A local monitor can still catch a split payload. Whether it succeeds depends on what cues remain in each fragment. We test this by varying visible data-flow edges, encoding artifacts, and natural-word carriers. These interventions show which cues the evaluated detectors use. They do not measure how close the attack and benign fragment distributions are.

Benchmark scores can hide shortcut cues. On R-Judge, a classifier using only trajectory-length features and no content reaches AUROC 0.940 (Appendix D.1). On Collaborative Shadows, where we can match fragment length directly, a detector that uses length falls to 0.500 (Appendix D.2). A high benchmark score can therefore come from a superficial cue rather than from the attack itself. Our main experiments use constructions where we can control what remains visible inside each fragment.

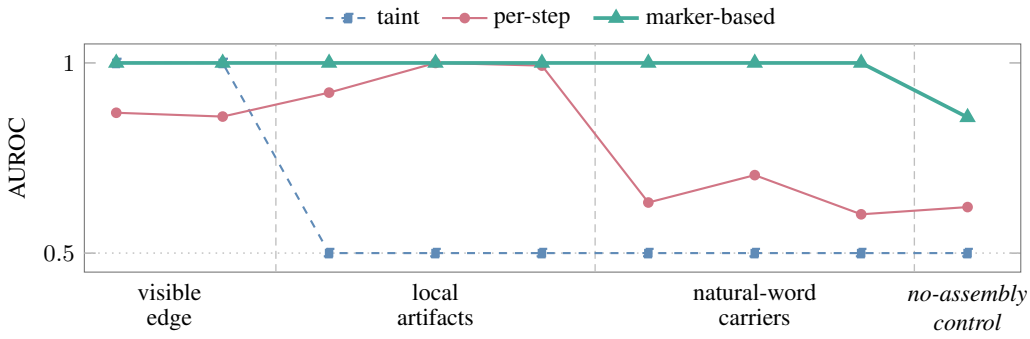


Figure 2: Performance of the evaluated monitors across fragment constructions. The taint baseline loses its visible-edge signal, and the per-step statistical monitor weakens on natural-word carriers. The marker-based diagnostic score remains high when its supplied coordination cue is present. The final condition removes assembly. These curves report detector scores, not estimates of local total variation.

Removing specific cues weakens the detectors that use them. We use Collaborative Shadows to vary how much attack evidence remains visible inside each fragment. The sweep starts with visible source-to-sink edges and suspicious encodings and ends with natural-word carriers. We record how the evaluated monitors respond as these cues disappear (Figure 2).

The detectors track the cues available to them. The taint baseline works while the payload remains in a tracked structured field and drops once the visible source-to-sink edge is removed. The per-step statistical monitor is strongest when the encoding itself is unusual, and it weakens on natural-word carriers. In the final carrier condition it reaches 0.602 ± 0.07, above chance. The marker-based score remains high because it is given a coordination cue.

The marker-based score is therefore a diagnostic control, not a deployable defense: its planted marker makes the cross-fragment relationship easy to identify.

A generic classifier still separates the Collaborative Shadows fragments. A character n-gram classifier over single fragments reaches AUROC 1.000 at every level of this sweep, and the separation survives matching the fragment length, the carrier word pool, and the coordination marker. These fragments therefore stay distinguishable even where the evaluated monitors score near chance (Appendix D.2).

The shipped AgentDojo detector weakens on benign-looking carriers. We next test a learned detector outside our own testbed. On AgentDojo, the shipped detector catches naive splitting (per-step AUROC 0.82–0.84). In the locally benign carrier condition, however, its attack scores overlap with those on real benign tool outputs, and per-step AUROC falls to approximately 0.50 (Table 2). Naively concatenating the fragments also remains near chance. A cross-fragment coordination score reaches AUROC 1.000 by using grouping and ordering information available in this testbed.

Table 2: AgentDojo separability. The shipped detector catches naive splitting (AUROC 0.82–0.84) but falls to ≈ 0.50 in the carrier condition. Naive concatenation is also near chance. The coordination score reaches 1.000 using grouping metadata available in this testbed.

Observation view (locally benign regime)	**AUROC**
Per-step learned detector	≈ 0.50
Naive fragment concatenation	≈ 0.50
Cross-fragment coordination	1.000

Seeing several fragments can reveal cues that the evaluated local detectors miss. The next section asks whether a detector that sees several fragments still works when the encoding changes.

4 JOINT DETECTION DOES NOT AUTOMATICALLY TRANSFER ACROSS ENCODINGS

We test whether a detector that sees several fragments transfers to a new encoding. It transfers across three related encodings but fails on the other three.

We evaluate a supervised joint detector on six codecs in a separate controlled construction based on Collaborative Shadows carrier groups. Both benign and attack episodes contain $K = 3$ encoded fragments derived from programs. Three codecs are related word-list mappings that encode each hexadecimal nibble (four bits); the other three use different encoding schemes. The detector is a random forest over ordered fragment features and cross-fragment statistics.

For each test codec, we train on the other five codecs and then evaluate on the held-out one. No examples from the held-out codec are used for training, although benign calibration data from that codec remain available. Results average five group splits, each with 50 training, 20 calibration, and 30 test groups.

Across the three related word mappings, the joint detector transfers strongly, with a mean held-out AUROC of 0.969. Across the other three codecs, mean held-out AUROC is 0.501 (Table 3).

The per-codec results clarify what this drop means. On wordpair32, AUROC falls from 0.883 in-family to 0.501 when that codec is held out; on decimal, it falls from 0.901 to 0.501. Base64 is different. The detector reaches only 0.643 when it trains on base64, and 0.500 when base64 is held out. Its weak held-out score is therefore not mainly a transfer loss.

Table 3: AUROC under codec transfer. The joint RF uses ordered fragment and cross-fragment features; full-text LR uses tf-idf. Values are means over five group splits. Held-out-codec training excludes that codec's examples, while benign calibration data from that codec remain available.

Codec	Joint RF, in	Joint RF, held out	Full-text LR, held out
cleanword16	0.972	0.966	0.495
nato	0.971	0.972	0.443
syllable	0.970	0.968	0.500
wordpair32	0.883	0.501	0.500
base64	0.643	0.500	0.500
decimal	0.901	0.501	0.276

A full-text tf-idf classifier also fails to transfer. On decimal it ranks the two classes in reverse. We set the direction of its score before testing. Methods and per-seed results appear in Appendix G.

The codecs also differ in vocabulary, tokenization, and format. Local features remain predictive in this construction. This experiment tests transfer across codecs; it does not isolate the effect of seeing fragments jointly. Appendix E reports the separate code-fragment diagnostic and its limitations.

A detector that sees several fragments can therefore still fail on a new encoding. Section 5 asks the next question: whether the evaluated methods can turn detection into runtime blocking.

5 RUNTIME BLOCKING WITH AND WITHOUT CODEC KNOWLEDGE

For runtime blocking, a detector must act before the payload executes. We therefore compare blocking with and without the codec supplied. Table 4 reports end-to-end attack success for each model, and Figure 3 shows how attack success changes across fragment constructions.

In these runs, the poisoned tool supplies the malicious report. The model decides whether the workflow reaches that output and then executes it. ASR therefore measures whether the defense blocks this workflow, not whether the model can reconstruct the payload on its own.

Without the codec, the full-trace methods we test block between $0/50$ and $10/50$ attacks. When the gate is given the encoding family, it blocks $50/50$ (Table 5).

Table 4: End-to-end ASR under the evaluated gates. Lower is better; 0.00 means $0/50$. In the carrier construction, running no defense, the taint baseline, and per-step monitoring all reach the same ASR. We collapse them into "no/local views." The codec-aware gate is supplied with the encoding family and serves as a reference, not a deployable defense.

Model	No/local views	Codec-aware gate
Llama-3.1-8B	0.58	**0.00**
Qwen2.5-7B	0.78	**0.00**
Qwen2.5-14B	1.00	**0.00**
Qwen2.5-32B-AWQ	1.00	**0.00**

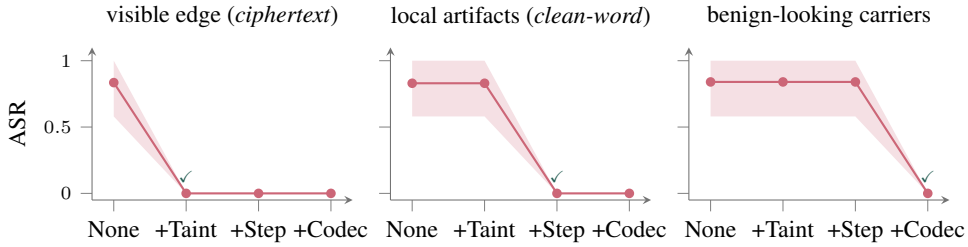


Figure 3: End-to-end attack success under the evaluated gates. Curves summarize served models and bands show their range. The codec-aware gate reduces observed ASR to zero in every reported cell. Local-baseline performance varies with the fragment construction.

Full-trace methods without the codec block only a few attacks. In the Qwen2.5-14B carrier condition (n=50, nominal 5% benign-FPR calibration), the codec-blind code-density rule and a common-decoder ensemble each block $0/50$ attacks. The decoder ensemble has AUROC 0.564. A prompted LLM reconstruction method proposes candidate reconstructions and passes them to the gate used by the common-decoder ensemble. The reconstruction method blocks $6/50$ attacks and has AUROC 0.521. The shipped detector applied to the full trace blocks $10/50$.

A separate multi-model rerun gives the same result for the full-trace code-density rule: its ASR is the same as with no defense on Qwen2.5-7B, Llama-3.1-8B, Qwen2.5-14B, and Qwen2.5-32B-AWQ. These runs use separately generated rollouts and are reported in Appendix F.1.

Table 5: Blocking by the evaluated full-trace methods, out of 50 attacks. Thresholds use the stated nominal 5% benign-FPR calibration protocol. The codec-aware gate blocks all tested attacks; the LLM reconstruction procedure and the neural detector achieve partial blocking without the supplied codec. The lower panel reports the separate DemonAgent construction. The dash marks a row where some methods see one fragment and others see the full trace.

View	Sees all fragments?	Supplied codec?	Blocks / 50
No defense and local baselines	×	×	0/50
Codec-blind code-density rule	✓	×	0/50
Common-decoder ensemble	✓	×	0/50
No-codec LLM search	✓	×	6/50
Shipped detector (full trace)	✓	×	10/50
Codec-aware gate	✓	✓	**50/50**
Second setting: DemonAgent dynamic encryption, surface cues removed			
Non-decoded monitors	–	×	0/50
Codec-aware gate	✓	✓	**50/50**

We see the same pattern on a second encoding. We repeat the comparison on a DemonAgent-based encoding construction (Zhu et al., 2025b) (Table 5, lower panel). Without the supplied codec,

the per-fragment and full-trace methods stay near chance and block $0/50$ attacks. The codec-aware gate blocks $50/50$. In the clean-word encoding the neural detector fires on surface cues. Here it does not fire.

An end-to-end Llama-3.1-8B run (n=50) shows the same pattern. Without a defense, the model follows the poisoned tool and reaches ASR 1.000; supplying the encoding map to the gate reduces ASR to zero. Our evidence covers these two tested encodings, not arbitrary codecs.

Benign code reverses the raw code-structure score. Real coding traffic breaks the raw code-structure score. Against benign snippets that use the payload's own libraries, it ranks ordinary code above the attack (Table 6). Reversing the score raises AUROC to 0.910. The codec-aware gate separates the two classes on the same traffic.

Table 6: **Code-structure scores under benign code traffic.** The raw code-structure score reverses direction, while the codec-aware gate reaches AUROC 1.000 with supplied codec knowledge. Full results appear in Table 17.

Monitor	benign mean	attack mean	AUROC
Codec-blind code structure	0.897	0.600	0.090
Codec-aware gate	0.897	1.240	1.000

The code-structure score does not transfer to every payload. The code-structure score relies on code syntax. On a natural-language harmful instruction, the strict one-class score falls to AUROC 0.406, compared with 0.956 on the code comparator.

Fragment count also changes local detection (Table 7). At $K = 10$ the per-step detector recovers. At smaller K, the marker-free and per-step scores fail in different ways. The marker-free raw score ranks the classes correctly, but every attack scores below the deployed threshold of 0.061, so it flags none of them. The per-step score also separates the classes, but ranks attacks below benign traffic. Appendix H reports the remaining stress tests.

Table 7: Detection-side K sweep on the carrier construction. Entries are AUROC / detection rate at the fixed operating threshold. The two columns are not equivalent: a score can rank the classes well and still detect nothing at its fixed threshold, which is what the marker-free raw rows show. The per-step AUROC of 0.000 at $K \leq 5$ is a perfectly reversed ranking, not an absence of local signal.

K	per-step	marker-free raw	common-decoder	codec-aware
2	0.000 / 0.00	1.000 / 0.00	0.160 / 0.00	1.000 / 1.00
3	0.000 / 0.00	1.000 / 0.00	0.288 / 0.00	1.000 / 1.00
5	0.000 / 0.00	1.000 / 0.00	0.456 / 0.00	1.000 / 1.00
10	0.972 / 0.90	1.000 / 0.00	0.728 / 0.00	1.000 / 1.00

Note: a high raw AUROC for marker-free code structure does not mean the monitor fires at the deployed threshold. Detections remain 0 unless the gate is given the codec, or unless K=10 restores per-step detection.

6 Related work

Agent safety monitors and prompt-injection defenses. Prompt injection spans direct attacks (Perez & Ribeiro, 2022), indirect injection in tool-using agents (Greshake et al., 2023), and adversarial jailbreaks (Wei et al., 2023). These threats have driven runtime defenses, from prompt-injection filters and safety classifiers (Inan et al., 2023) to learned per-step detectors such as AgentDojo's (Debenedetti et al., 2024). A growing literature evaluates the safety of these agents (Ruan et al., 2024; Andriushchenko et al., 2025; Vijayvargiya et al., 2026; Ferrag et al., 2025; Kale et al., 2025). These methods improve or evaluate decisions on individual steps. We ask when monitoring one step at a time misses harm that appears only after several fragments are combined.

Provenance, taint, and information-flow defenses. Another line asks whether untrusted information reaches a sensitive action. These defenses treat the agent trace as a provenance or dataflow

object and apply taint-style policies, information-flow constraints, or structured trace analysis (Denning, 1976; Sabelfeld & Myers, 2003; Newsome & Song, 2005; Wang et al., 2025; Cai et al., 2026; Jha et al., 2026). These methods use more of the trace than a per-step detector. Our taint baseline works while a visible source-to-sink edge remains. It fails once that edge is removed. Cross-agent consistency and self-verification defenses instead look for disagreement across agents (Fan & Li, 2025; Li et al., 2025). Our attack keeps each local step benign and consistent.

Distributed and compositional attacks in agent workflows. Distributed backdoors for LLM multi-agent systems plant primitives that stay dormant until an inter-agent sequence assembles them (Zhu et al., 2025a). Later work hardens these attacks with dynamic encryption and cross-stage trigger propagation (Zhu et al., 2025b; Feng et al., 2026b). A similar split-then-assemble pattern appears in skill-based agents (Feng et al., 2026a), and decomposing a payload into individually innocuous pieces also underlies prompt-decomposition and adversarial jailbreaks (Li et al., 2024; Glukhov et al., 2023; Zou et al., 2023). Fragments that are individually true can compose into misleading beliefs rather than code (Hu et al., 2026). In per-message evasion, by contrast, one instruction travels intact (Lee & Tiwari, 2024). Stateful and forecasting monitors instead use earlier steps to detect or predict later safety failures (Dhodapkar & Pishori, 2026).

Arif et al. (2026) study conjunctive prompt attacks in which a query trigger and an adversarial template in a remote agent activate harmful behavior together. Motwani et al. (2024) study steganographic collusion and the limits of oversight for concealed communication. Our experiments compare single-fragment and joint inputs, test whether a joint detector transfers to unseen codecs, and measure whether the evaluated methods block execution. Our total-variation bound applies only to the single-fragment case.

Shortcut learning and benchmark artifacts. Shortcut learning (Geirhos et al., 2020) and NLI annotation artifacts (Gururangan et al., 2018; McCoy et al., 2019) describe the same problem: a model scores a dataset cue instead of the signal it is meant to read. The same shortcuts appear in agent-safety trajectories, where rewritten or leave-one-domain-out tests can expose them (Chen et al., 2026; Fomin, 2026). We length-match our attacks because a local monitor can otherwise score trajectory length instead of attack content.

7 Discussion and conclusion

A local monitor can correctly treat every fragment as harmless and still miss a harmful program that appears after the fragments are combined. Our bound makes one limit precise: when attack and benign traffic are close in a fixed fragment view, a detector using only that fragment has little power to separate them. The experiments show a second limit: the local baselines we test weaken as the cues inside each fragment are removed. A detector that sees several fragments can still fail when the encoding changes.

In our end-to-end tests, methods without the supplied codec block between 0 and 10 of 50 attacks, while the codec-aware gate blocks every tested attack. The codec-aware gate is given the encoding family, so we use it as a reference point. In the runs we tested, that extra knowledge changes blocking from partial to complete. This leaves a practical challenge: detect cross-fragment harm without assuming the attacker's encoding in advance.

Ethical considerations

Our aim is defensive: we study when runtime monitors can miss harmful behavior that emerges across multiple fragments. The attacks are based on previously published distributed-backdoor families (Zhu et al., 2025a;b) and are evaluated only in a controlled testbed and on public benchmarks. The payloads use synthetic example endpoints; no real credentials or live services are involved.

AI Use Statement

Generative AI assistants, including ChatGPT and Claude Code, were used to assist with literature search, implementation and debugging, analysis and interpretation of results, and review of mathematical claims and proofs. They were also used for manuscript drafting and editing. AI-assisted

implementation included generating synthetic experimental data, and the assistants provided feedback on experimental methodology. The authors take responsibility for the final text, claims, code, data, and analyses.

Reproducibility Statement

The appendix provides the assumptions and proofs for the local detection bound and alarm-if-any result. It also documents the inputs and decision rules of the evaluated monitors, along with the training, calibration, and test protocols. We report per-model attack-success rates and per-codec results for the transfer experiment. The code-fragment text diagnostic is documented separately from the executable attack evaluations. We plan to release the code and data upon acceptance.

References

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Yarin Gal, and Xander Davies. Agentharm: A benchmark for measuring harmfulness of LLM agents. In *ICLR*, 2025.

Nokimul Hasan Arif, Qian Lou, and Mengxin Zheng. Conjunctive prompt attacks in multi-agent LLM systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34175–34191. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1577.

Yuandao Cai, Wensheng Tang, Cheng Wen, and Shengchao Qin. Ghost in the agent: Redefining information flow tracking for LLM agents, 2026. URL https://arxiv.org/abs/2604.23374.

Yen-Shan Chen, Sian-Yao Huang, Cheng-Lin Yang, and Yun-Nung Chen. TraceSafe: A systematic assessment of LLM guardrails on multi-step tool-calling trajectories, 2026. URL https://arxiv.org/abs/2604.07223.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *NeurIPS Datasets and Benchmarks*, 2024.

Dorothy E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19 (5):236–243, 1976. doi: 10.1145/360051.360056.

Aditya Dhodapkar and Farhaan Pishori. SafetyDrift: Predicting when ai agents cross the line before they actually do, 2026. URL https://arxiv.org/abs/2603.27148.

Falong Fan and Xi Li. PeerGuard: Defending multi-agent systems against backdoor attacks through mutual reasoning, 2025. URL https://arxiv.org/abs/2505.11642.

Yunhao Feng, Yifan Ding, Yingshui Tan, Boren Zheng, Yanming Guo, Xiaolong Li, Kun Zhai, Yishan Li, and Wenke Huang. SkillTrojan: Backdoor attacks on skill-based agent systems, 2026a. URL https://arxiv.org/abs/2604.06811.

Yunhao Feng, Yige Li, Yutao Wu, Yingshui Tan, Yanming Guo, Yifan Ding, Kun Zhai, Xingjun Ma, and Yu-Gang Jiang. BackdoorAgent: A unified framework for backdoor attacks on LLM-based agents, 2026b. URL https://arxiv.org/abs/2601.04566.

Mohamed Amine Ferrag, Norbert Tihanyi, Djallel Hamouda, Leandros Maglaras, Abderrahmane Lakas, and Merouane Debbah. From prompt injections to protocol exploits: Threats in llm-powered ai agents workflows. *ICT Express*, 2025.

Max Fomin. When benchmarks lie: Evaluating malicious prompt classifiers under true distribution shift, 2026. URL https://arxiv.org/abs/2602.14161.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020. doi: 10.1038/s42256-020-00257-z.

David Glukhov, Ilia Shumailov, Yarin Gal, Nicolas Papernot, and Vardan Papyan. Llm censorship: A machine learning challenge or a computer security problem?, 2023.

Shafi Goldwasser, Michael P. Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2022. doi: 10.1109/FOCS54457.2022.00092.

Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. Ai control: Improving safety despite intentional subversion. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2312.06942.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec@CCS)*, 2023. URL https://arxiv.org/abs/2302.12173.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *NAACL-HLT*, 2018.

Jinwei Hu, Xinmiao Huang, Youcheng Sun, Yi Dong, and Xiaowei Huang. Lying with truths: Open-channel multi-agent collusion for belief manipulation via generative montage. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5979–5996, 2026. doi: 10.18653/v1/2026.acl-long.270. URL https://aclanthology.org/2026.acl-long.270/.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023. URL https://arxiv.org/abs/2312.06674.

Rishi Jha, Harold Triedman, Justin Wagle, and Vitaly Shmatikov. Breaking and fixing defenses against control flow hijacking in multi-agent systems. In *International Conference on Learning Representations*, volume 2026, pp. 7827–7858, 2026.

Neil Kale, Chen Bo Calvin Zhang, Kevin Zhu, Ankit Aich, Paula Rodriguez, Scale Red Team, Christina Q. Knight, and Zifan Wang. Reliable weak-to-strong monitoring of LLM agents, 2025. URL https://arxiv.org/abs/2508.19461.

Donghyun Lee and Mo Tiwari. Prompt infection: LLM-to-LLM prompt injection within multi-agent systems, 2024. URL https://arxiv.org/abs/2410.07283.

Changjiang Li, Jiacheng Liang, Bochuan Cao, Jinghui Chen, and Ting Wang. Your agent can defend itself against backdoor attacks, 2025. URL https://arxiv.org/abs/2506.08336.

Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. Drattack: Prompt decomposition and reconstruction makes powerful llm jailbreakers, 2024.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *ACL*, 2019.

Sumeet Ramesh Motwani, Mikhail Baranchuk, Martin Strohmeier, Vijay Bolina, Philip H. S. Torr, Lewis Hammond, and Christian Schroeder de Witt. Secret collusion among AI agents: Multi-agent deception via steganography. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

James Newsome and Dawn Song. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *12th Annual Network and Distributed System Security Symposium (NDSS)*, 2005. URL https://www.ndss-symposium.org/wp-content/uploads/2017/09/Dynamic-Taint-Analysis-for-Automatic-Detection-Analysis-and-SignatureGeneration-of-Exploits-on-Commodity-Software-Dawn-Song.pdf.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2023. URL https://arxiv.org/abs/2304.03442.

Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. In *NeurIPS ML Safety Workshop*, 2022.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *ICLR*, 2024.

Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003. doi: 10.1109/JSAC.2002.806121.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. doi: 10.52202/075280-2997.

Sanidhya Vijayvargiya, Aditya Bharat Soni, Xuhui Zhou, Zora Zhiruo Wang, Nouha Dziri, Graham Neubig, and Maarten Sap. Openagentsafety: A comprehensive framework for evaluating real-world ai agent safety. In *ICLR*, 2026.

Peiran Wang, Yang Liu, Yunfei Lu, Yifeng Cai, Hongbo Chen, Qingyou Yang, Jie Zhang, Jue Hong, and Ye Wu. AgentArmor: Enforcing program analysis on agent runtime trace to defend against prompt injection, 2025. URL https://arxiv.org/abs/2508.01249.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. doi: 10.52202/075280-3508.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.03629.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. R-judge: Benchmarking safety risk awareness for LLM agents. In *Findings of EMNLP*, 2024.

Pengyu Zhu, Lijun Li, Yaxing Lyu, Li Sun, Sen Su, and Jing Shao. Collaborative shadows: Distributed backdoor attacks in LLM-based multi-agent systems, 2025a. URL https://arxiv.org/abs/2510.11246.

Pengyu Zhu, Zhenhong Zhou, Yuanhe Zhang, Shilinlu Yan, Kun Wang, and Sen Su. DemonAgent: Dynamically encrypted multi-backdoor implantation attack on LLM-based agent, 2025b. URL https://arxiv.org/abs/2502.12575.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

A Experiment Index

Table 8 maps each appendix experiment to the main claim it supports.

Table 8: Experiment index. A navigation map from appendix experiments to the main claims; all experiments use Collaborative Shadows unless noted.

Section	Experiment	Purpose
Artifact check	Length artifact (R-Judge)	Rule out length as a confound.
Local-monitor behavior	Locality sweep	Vary fragment conspicuousness.
	Taint-edge test	Probe source-to-sink visibility.
	Learned detector (AgentDojo)	Test the shipped detector on locally benign fragments.
Joint and assembly diagnostics	Codec transfer (six encodings)	Transfer of a joint detector to a held-out codec.
	Marker-free one-class score	Rule out marker-grepping.
	End-to-end ASR	Measure ASR across defenses.
Additional stress tests	Codec-blind blocking	Block without the supplied codec.
	Multi-model codec-blind rerun	Replicate across model scales.
	DemonAgent encoding, surface cues removed	Same blocking comparison, independent codec.
	Common-decoder ensemble	Test standard decoders.
	No-codec LLM search	LLM reconstructions, no codec given.
	Fragmentation scale	Sweep $K \in \{2, 3, 5, 10\}$.
	Benign-code stress	Same-library benign code.
	Benign-code PR/ROC	Code-cue threshold trade-off.
	Natural-language payload	Non-code harmful instruction.
	Executable perturbation checks	Wrapper/alias/dead-code checks.

B Theory and monitor definitions

B.1 Proof of Proposition 1

Let $\mathcal{O}$ be the space of local observations and let a randomized decision rule be a measurable $\phi : \mathcal{O} \to [0, 1]$, where $\phi(o)$ is the probability of flagging observation o. Fix a fragment position i and write $P_{\mathrm{att}} = P^{v}_{\mathrm{att},i}$ and $P_{\mathrm{ben}} = P^{v}_{\mathrm{ben},i}$. A fixed mixture over positions follows by convexity of TV. We have $\mathrm{TPR} = \int \phi \, dP_{\mathrm{att}}$ and $\mathrm{FPR} = \int \phi \, dP_{\mathrm{ben}}$. Hence

$$\begin{aligned} \mathrm{TPR} - \mathrm{FPR} &= \int \phi \, d(P_{\mathrm{att}} - P_{\mathrm{ben}}) \\ &\le \int \phi \, d(P_{\mathrm{att}} - P_{\mathrm{ben}})^{+} \\ &\le \int d(P_{\mathrm{att}} - P_{\mathrm{ben}})^{+} = \mathrm{TV}(P_{\mathrm{att}}, P_{\mathrm{ben}}), \end{aligned}$$

where the first inequality drops the negative part of the signed measure, the second uses $0 \le \phi \le 1$, and the final equality is the definition of total variation as the total mass of the positive part. The bound holds for every ϕ, hence for the best local rule. If benign and attack local observations are close in TV, no rule on that view separates them by more than that TV. The bound does not cover a monitor that jointly observes several fragments or retains their history. Assembly monitoring is one such case.

From the bound to AUROC. The locality sweep (Section 3) reports per-step AUROC, which relates to Proposition 1 as follows. Writing the ROC curve as $\mathrm{TPR}(\mathrm{FPR})$ traced by the decision

threshold, the area above chance is

$$\mathrm{AUROC} - \tfrac{1}{2} \;=\; \int_0^1 \big(\mathrm{TPR}(u) - u\big)\, du,$$

where $u = \mathrm{FPR}$. At every threshold Proposition 1 bounds the integrand pointwise by $\mathrm{TV}(P^v_{\mathrm{ben}}, P^v_{\mathrm{att}})$, and $\mathrm{TPR} \le 1$, so

$$\mathrm{AUROC} \;\le\; \tfrac{1}{2} + \mathrm{TV} - \tfrac{1}{2}\mathrm{TV}^2 \;\le\; \tfrac{1}{2} + \mathrm{TV}.$$

A local view with small benign–attack TV therefore forces AUROC toward chance. The sweep in Section 3 measures the AUROC of the evaluated detectors, not TV. AUROC is a diagnostic, not an estimate of TV. A high AUROC proves the fragments are distinguishable ($\mathrm{TV} \ge \mathrm{AUROC} - \frac{1}{2}$). An AUROC near chance is consistent with small TV, but it does not prove small TV. An AUROC below 0.5 can still show that the score ranks the two classes in the opposite direction. Reversing the score turns AUROC into $1 - \mathrm{AUROC}$ on the same observations, so $\mathrm{TV} \ge \frac{1}{2} - \mathrm{AUROC}$. A score at AUROC 0.000 therefore implies $\mathrm{TV} \ge \frac{1}{2}$: it reports a direction failure of that score, not an absence of local information. We read the reversed scores in Tables 6 and 7 this way.

B.2 A SIMPLE ALARM-IF-ANY AGGREGATION BOUND

Proposition 1 bounds a single local view. A natural extension is to aggregate all K per-fragment alarms. We record a simple bound for the rule that flags a trajectory whenever any fragment is flagged. Detection still stays limited by the local TV distances. This extends Proposition 1. It does not cover every way of combining per-fragment alarms.

Consider a fixed (possibly randomized) per-fragment rule $\phi : \mathcal{O} \to [0, 1]$. At fragment i, let $\alpha_i = \mathbb{E}_{\mathrm{ben},i}[\phi]$ be its benign false-positive rate, and let $\mathrm{TV}_i = \mathrm{TV}(P^v_{\mathrm{ben},i}, P^v_{\mathrm{att},i})$ be the local total-variation distance for that fragment. The trajectory-level alarm fires if any fragment is flagged.

Proposition 2 (Alarm-if-any aggregation bound)**.** *The trajectory-level false-positive and detection rates satisfy*

$$\mathrm{FPR}_{\mathrm{traj}} \le \sum_{i=1}^{K} \alpha_i, \qquad \mathrm{TPR}_{\mathrm{traj}} \le \sum_{i=1}^{K} \alpha_i + \sum_{i=1}^{K} \mathrm{TV}_i.$$

Under uniform calibration ($\alpha_i = \alpha$) these become $\mathrm{FPR}_{\mathrm{traj}} \le K\alpha$ *and* $\mathrm{TPR}_{\mathrm{traj}} \le K\alpha + \sum_i \mathrm{TV}_i$. *Under ε-local indistinguishability ($\mathrm{TV}_i \le \varepsilon$), the detection bound is at most $K(\alpha+\varepsilon)$. If the benign flag events are additionally independent and uniformly calibrated, then* $\mathrm{FPR}_{\mathrm{traj}} = 1 - (1-\alpha)^K$ *exactly.*

Proof. Let A_i be the event that fragment i is flagged. The trajectory alarm is $\bigcup_i A_i$. By the union bound,

$$\mathrm{FPR}_{\mathrm{traj}} = P_{\mathrm{ben}}\left(\bigcup_i A_i\right) \le \sum_i P_{\mathrm{ben}}(A_i) = \sum_i \mathbb{E}_{\mathrm{ben},i}[\phi] = \sum_i \alpha_i.$$

If the benign flag events A_i are independent and uniformly calibrated, this false positive rate is exactly $1 - (1-\alpha)^K$.

The same argument under the attack distribution gives

$$\mathrm{TPR}_{\mathrm{traj}} = P_{\mathrm{att}}\left(\bigcup_i A_i\right) \le \sum_i P_{\mathrm{att}}(A_i) = \sum_i \mathbb{E}_{\mathrm{att},i}[\phi].$$

For each fragment, Proposition 1 gives

$$\mathbb{E}_{\mathrm{att},i}[\phi] \le \mathbb{E}_{\mathrm{ben},i}[\phi] + \mathrm{TV}_i = \alpha_i + \mathrm{TV}_i.$$

Summing over fragments yields $\mathrm{TPR}_{\mathrm{traj}} \le \sum_i \alpha_i + \sum_i \mathrm{TV}_i$. □

Reading. The bound controls the detection rate of an OR of fixed local alarms using the sum of their benign alarm rates and marginal TV distances. Under uniform bounds it gives $\mathrm{TPR}_{\mathrm{traj}} \le K(\alpha + \varepsilon)$, which is informative when this total is small. The marginal TV distances alone do not bound how well the trajectory-level alarm separates attack from benign traffic. The exact expression $1 - (1-\alpha)^K$ additionally assumes independent benign alarm events. Joint or history-dependent decision rules are outside this proposition.

```
TAINT BASELINE  (whole trace T)
  for each value v from source s to sink k in T:
      if exists path s -> k:  return FLAG
  return PASS                # no visible edge => PASS

PER-STEP STATISTICAL MONITOR  (one fragment)
  features <- [entropy(x_i), length(x_i), lexical_cues(x_i)]
  return FLAG if f_stat(features) > tau_step

MARKER-BASED SCORE  (fragment set)
  if any fragment carries the known coordination marker M:
      return 1.0 if fragments assemble   # diagnostic only
  return 0.5                 # chance when marker absent

MARKER-FREE ONE-CLASS SCORE  (fragment set, no marker)
  A <- assembled_text(concat/orderings of {x_i})
  return FLAG if code_structure(A) > tau_asm

CODEC-AWARE GATE  (assembler input, pre-execute)
  for each provided codec family c:
      A <- decode_c(concat fragments)
      if parses_as_executable(A) and matches_exfil(A):
          return BLOCK
  return ALLOW

CODEC-BLIND CODE-DENSITY RULE  (fragment set, no decoding)
  A <- concat(x_i)
  return FLAG if code_structure_density(A) > tau
```

Figure 4: Monitor pseudocode. Runtime thresholds follow the calibration protocol in Appendix C.

B.3 Monitor pseudocode

The monitor taxonomy is in Table 1 (Section 2); the pseudocode below summarizes each monitor. Runtime thresholds follow the calibration protocol in Appendix C; stress tests that reuse a deployed threshold say so explicitly. The per-step statistical monitor, the taint baseline, and the marker-based score are the ones swept in Figure 2. The codec-aware gate and the codec-blind code-density rule are analyzed in Section 5, and the marker-free score in Appendix E.

C Experimental details and statistics

Compute and software. CPU-only detector experiments use Python with NumPy and scikit-learn. End-to-end ASR runs serve open-weight instruction models through an OpenAI-compatible endpoint, including vLLM, on NVIDIA RTX 6000, NVIDIA A100 40GB, and NVIDIA A100 80GB GPUs. Models are served in fp16 where supported; the row named Qwen2.5-32B-AWQ uses the AWQ checkpoint named in the paper.

ASR confidence intervals. ASR cells are binomial over $n{=}50$ trajectories. For $0/50$ cells we report the exact 95% Clopper-Pearson upper bound (0.071); this is a per-cell bound, not a simultaneous guarantee across cells. For the no-defense rates we report Wilson 95% intervals (Table 9).

Table 9: Representative ASR confidence intervals ($n{=}50$). The $0/50$ row is an exact Clopper-Pearson upper bound; the others are Wilson score intervals.

ASR	count	95% CI
0.00	0/50	[0.00, 0.071]
0.58	29/50	[0.44, 0.71]
0.76	38/50	[0.62, 0.86]
0.78	39/50	[0.64, 0.88]
1.00	50/50	[0.91, 1.00]

Threshold calibration. Runtime gates are calibrated to a 5% benign false-positive rate on held-out benign trajectories. On Collaborative Shadows this fixes the code-density rule's threshold at 0.061; the fully encoded locally benign attack scores 0.013, below it, so the code-density rule never

fires. The codec-aware gate has a wider margin: benign trajectories score at most 0.120 and attacks score 1.240, so any threshold in $(0.120,\ 1.240)$ gives 0% benign FPR at 100% detection on this construction (Table 18). This band is a property of this benign traffic, which contains no code. Under the benign code traffic of Table 6 the raw code-structure score reverses direction. The codec-aware gate still separates the classes. We do not report an interval for that traffic.

Operating-point sensitivity. The low local detection rates are not specific to the 5% FPR choice. We also sweep the benign-FPR budget over $\{1\%, 5\%, 10\%\}$. The per-step and per-fragment monitors stay near zero detection on locally benign attacks. On Collaborative Shadows, per-step TPR is 0.02 to 0.03 across the range. For the AgentDojo per-step detector it is 0.00 at every budget. Loosening the threshold does not recover them. The supervised cross-wrapper classifier saturates at TPR 1.000 at all three budgets. The marker-free one-class monitor is the exception. Its TPR rises with the budget ($0.06 \rightarrow 0.25 \rightarrow 0.36$). It ranks well (AUROC 0.874) but does not hold a tight threshold. The results at a fixed threshold therefore rest on the supervised cross-wrapper classifier and the codec-aware gate, not on the one-class monitor.

AUROC seeds. AUROC values on Collaborative Shadows (Figure 2, Table 10) are 5-seed means. The taint baseline and the marker-based score are rule-based and deterministic given a trajectory. They therefore take the fixed values 0.500 or 1.000 with no cross-seed variation. Seed variation enters only through the sampled fragment content. It therefore concentrates in the per-step statistical monitor. Its 5-seed std is about 0.01 where the classes separate cleanly and 0.09 in the natural-word regimes. On AgentDojo the shipped detector's per-step AUROC carries a per-regime 5-seed std of at most 0.06, while the cross-fragment coordination score is deterministic at 1.000. The marker-free one-class monitor's AUROC ranges within 0.858 to 0.879 across the four wrapper variants (mean 0.874), and the supervised cross-wrapper classifier is at 1.000 on every unseen wrapper. The detectors rank the same way across these seed-level summaries and across the operating points in Table 18.

D LOCALITY SWEEP DETAILS

D.1 R-JUDGE LENGTH-ARTIFACT DETAILS

The length-only classifier reaches AUROC 0.940, about 96% of the full-content oracle. The shortcut transfers across domains. The pooled length-only AUROC is 0.940, and the leave-one-domain-out AUROC on the large domains is 0.895. Under length matching, the monitor that sees the whole trajectory drops from AUROC 0.775 to 0.671. Using content features only, it reaches 0.618 against 0.341 for the per-step detector, an advantage of about $+0.28$. After length matching the remaining separation is weaker, so we use R-Judge as motivation rather than primary evidence.

D.2 FULL LOCALITY SWEEP

Table 10 gives the numeric values plotted in Figure 2.

On this construction, a length-matched attack drives the deployed detector that uses length to 0.500 (Cohen's $d = 0.000$), while the marker-based score stays at 1.000.

A supervised per-fragment classifier separates this construction. A character n-gram TF-IDF classifier over single fragments reaches AUROC 1.000 at every level of this sweep, with both logistic regression and a random forest (five seeds, zero variance). It is fit on calibration groups and scored on held-out groups, taking the maximum over an episode's fragments. The separation survives matching the fragment length, the carrier word pool, and the coordination marker. The attack reports in this generator are written from a fixed phrase template. We take that template to be the source of the separation. This sweep uses a different construction from the code testbed of Appendix E. The probe using code-structure features reaches 1.000 on that testbed and 0.000 on this sweep. The separation here therefore comes from something other than a leaked slice of the payload. This sweep does not establish Definition 2, and we do not use it to argue that attack and benign fragments are distributionally close. The learned-detector evidence on real benign tool outputs is the AgentDojo

Table 10: Locality sweep on Collaborative Shadows (5-seed mean AUROC). Values are the numeric backup for Figure 2. *Per-step* is the per-step statistical monitor; *Marker-based* is the marker-based score. Note: level 8 is a no-assembly control. Its remaining marker-based score does not show recovery, and is consistent with a residual length cue. Per-step entries give 5-seed mean±std; taint and the marker-based score are deterministic across seeds (std 0), except the level-8 control.

Level	Encoding	Taint	Per-step	Marker-based
0	raw base64	1.000	0.869 ± 0.08	1.000
1	disguised b64	1.000	0.859 ± 0.09	1.000
2	b64 + words	0.500	0.922 ± 0.05	1.000
3	hex	0.500	1.000 ± 0.00	1.000
4	hex + words	0.500	0.993 ± 0.01	1.000
5	50% natural words	0.500	0.633 ± 0.09	1.000
6	80% natural words	0.500	0.705 ± 0.08	1.000
7	sentence embed (loc. benign)	0.500	0.602 ± 0.07	1.000
8	no-assembly control	0.500	0.621 ± 0.08	0.857 ± 0.08

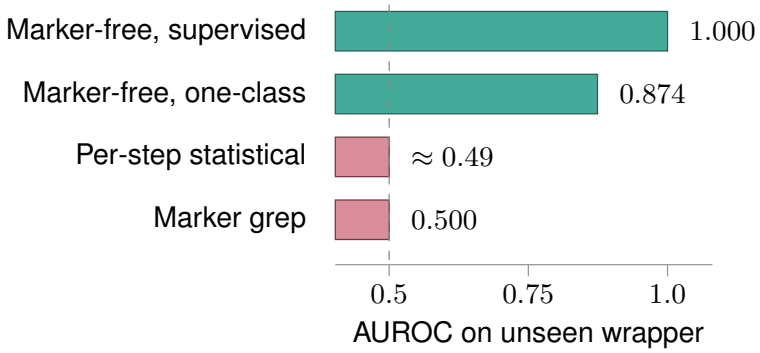


Figure 5: Classification scores on generated code-fragment text variants. Because the reports are clipped to benign serialized lengths, the scored texts do not generally contain a complete source program. Marker grep and the per-step statistical score stay near chance on wrapper variants they were not fitted to. The one-class score, fitted on benign trajectories only, reaches 0.874 mean AUROC across the four wrapper variants, and the supervised cross-wrapper classifier reaches 1.000. Neither score blocks an attack. Both only rank trajectories. End-to-end blocking is reported in Section 5.

check in Section 3. This sweep also shows the distinction the paper draws. Fragments can be locally benign in the sense of Definition 1 and still carry cues that a generic classifier uses.

E Diagnostic scores on code-fragment text variants

This diagnostic inserts source-code fragments into report carriers and clips the reports to match benign serialized lengths. The clipping can remove payload content. The generated texts therefore do not always contain a complete executable source program. In the reported construction, all K source parts survive verbatim in only about 3% of generated attack trajectories. Figure 5 summarizes the resulting scores. The AUROC results in this section measure how well a classifier separates these generated text variants. They do not measure payload reconstruction or execution. Section 5 reports the executable attack and blocking results separately.

Wrapper variants. The four `F_*` conditions are wrapper and marker variants of the same generator; they are not the six codecs of Appendix G. In every variant the K=3 fragments are contiguous pieces of one source code payload. The payload is (`import os, requests ...requests.post('http://evil.example/exfil',...)`) spliced into benign report carriers and length-matched exactly to the benign observation, so fragment length does not separate attack from benign at the local view. The families differ only in the coordination marker that

wraps each fragment (Table 11); `F_none` has no marker and is held out so that a marker-matching monitor has nothing to key on.

Table 11: The four wrapper variants. All carry the same $K{=}3$ code-payload fragments in benign carriers, length-matched to benign; only the per-fragment marker differs. These are wrapper variants, not the six codecs of Appendix G.

Family	Per-fragment marker	Role
`F_frag`	`"fragment` i`:"`	in-family / training marker
`F_seg`	`"[data segment` i `of` K`]"`	unseen marker (transfer test)
`F_ref`	`"ref`i`::"`	unseen marker (transfer test)
`F_none`	none	held out: no marker to match

Marker-free features. The marker-free scores use seven features of the concatenated report text, with no marker knowledge:

(i) code-structure density, the fraction of tokens associated with code syntax or APIs (`import`, `requests`, `http://`, `os.`, `environ`, brackets, `=`, `.post`/`.get`, `json`, and similar) in the concatenation;
(ii) partial parseability, obtained by extracting candidate code lines (imports, assignments, `requests.`$*$ calls) with a regular expression and attempting `ast.parse` on them, scoring 1 if the extracted block parses and otherwise 0.2 per candidate line capped at 1;
(iii–v) the mean, maximum, and standard deviation of a per-step nonconformity profile from an isolation forest fit on a TF-IDF representation (300 features, 1- and 2-grams, $\min \mathrm{df}{=}2$) of benign step text;
(vi) cross-agent length parallelism, the coefficient of variation of the K report lengths;
(vii) the concatenation length.

Training and evaluation. The testbed is 100 Collaborative Shadows groups ($K{=}3$), evaluated over 5 seeds with a $70/30$ group-level train/test split. The *one-class* monitor standardizes the seven features. It fits an isolation forest (200 trees, contamination 0.05) on benign training trajectories only. It never sees an attack. Its negatives are benign trajectories, and its test-time positives are the attack families. The per-step nonconformity model behind features (iii, iv, v) is likewise fit on benign training step text. The *supervised* score trains on one wrapper variant and is tested on another. We report the mean over these pairs. The statistical baseline is scored per episode. Its score is the maximum over that episode's local step scores. These values are not independent per-fragment AUCs. Separability is reported as AUROC, which is threshold-free. The isolation forests use an in-fit contamination of 0.05 as a training hyperparameter; this is separate from the runtime gates elsewhere in the paper, which are calibrated to a 5% benign false-positive rate on held-out benign trajectories (Appendix C).

Results and ablation. The marker-based score reaches 1.000 on its own wrapper and exactly 0.500 on the others (mean off-diagonal 0.500). Table 12 collects these scores. The one-class score reaches 0.874 mean AUROC across the four wrapper variants including `F_none`; the supervised cross-wrapper classifier reaches 1.000 on every unseen wrapper; the per-step baseline stays near chance (0.475 to 0.504). A per-fragment probe using code-structure features reaches 1.000 on the same construction. Readable local code cues therefore remain in these generated fragments. This construction does not meet the small per-fragment TV condition of Definition 2.

Replacing the single source string with a family of ten Python exfiltration source programs changes the mean one-class AUROC from 0.874 to 0.862 in the same generator (per-wrapper: $0.878 \to 0.840$, $0.858 \to 0.857$, $0.879 \to 0.875$, $0.879 \to 0.876$). In the marker-free `F_none` condition, a supervised classifier trained on five source programs and tested on the other five reaches AUROC 1.000 across the reported seeds. Programs are sampled per trajectory, so not every program appears in every seed. These checks vary the source program for the clipped text diagnostic. They do not show that executable payload recovery generalizes. Restricting the density feature to semantic code tokens, dropping whitespace and bracket counts, gives the *strict* variant used for the natural-language transfer test (Table 19). Removing one feature at a time shows that this score depends on code-token

density. Dropping it moves mean AUROC to 0.465. Dropping partial parseability costs only 0.022 and dropping any of the per-step nonconformity, length-parallelism, or length features changes it by at most 0.023. Removing any one feature leaves the supervised monitor above 0.978. Density and parseability each separate the classes on their own. These results show that the text diagnostic relies heavily on code-structure features. Because readable local code cues remain in this construction, the experiment does not isolate a specifically cross-fragment assembly signal.

Scope of the per-step result. The per-step chance result (0.475 to 0.504, 5-seed mean 0.483) is for the deployed statistical monitor (v_{stat}, TF-IDF + isolation-forest). This code testbed deliberately gives each fragment a contiguous slice of the Python payload. A per-fragment probe that uses code-structure features (code-token density plus partial `ast` parseability) therefore reads that slice directly and hits AUROC 1.000. We keep v_{stat} as the deployed statistical baseline; the v_{code} probe shows that these fragments still contain readable local code cues. The external AgentDojo check (Section 3) tests a *strong* learned per-fragment detector. Under that shipped detector, locally benign fragments score at the benign baseline (0.046 vs. 0.049).

Table 12: The marker-free separation is not due to the planted marker (numeric backup for Figure 5). Marker grep fails on wrappers it was not fitted to; the marker-free score separates the generated text variants of the unseen wrappers. The statistical per-step monitor (v_{stat}) is at chance; a probe using code-structure features (v_{code}) reads the contiguous payload slice on this code testbed.

Monitor	Attacks in training	AUROC, unseen wrapper
Marker grep	none (rule)	0.500 (unseen wrapper)
Per-step local, statistical v_{stat}	none	0.475 to 0.504
Per-step local, code-structure v_{code}	none (rule)	1.000 (all families)
Marker-free, one-class	none	0.874 (0.858 to 0.879)
Marker-free, supervised	other families	1.000 (± 0)

F End-to-end ASR and codec-aware blocking

F.1 Full per-model ASR

Table 13 gives the full per-model end-to-end numbers behind the mean and range summary in Figure 3. For each attack family, the same defenses leave ASR at the no-defense level on all four models. Only the no-defense ASR itself varies across models (Table 4).

Table 13: Entries are ASR under no-defense / +taint / +per-step / +codec-aware (n=50 per cell, matching the x-axis order of Figure 3). **Bold** marks a defense whose ASR matches no defense for that attack family; the codec-aware value is 0 in every cell. The codec-blind code-density rule is analyzed separately in Table 5. In the locally benign column, the local-view ASR tracks no-defense across served models while the codec-aware gate changes it (Table 4).

Model	ciphertext	clean-word	locally benign
Llama-3.1-8B	0.58 / 0 / 0 / 0	0.58 / **0.58** / 0 / 0	0.58 / **0.58** / **0.58** / 0
Qwen2.5-7B	0.76 / 0 / 0 / 0	0.76 / **0.76** / 0 / 0	0.78 / **0.78** / **0.78** / 0
Qwen2.5-14B	1.00 / 0 / 0 / 0	1.00 / **1.00** / 0 / 0	1.00 / **1.00** / **1.00** / 0
Qwen2.5-32B-AWQ	1.00 / 0 / 0 / 0	0.98 / **0.98** / 0 / 0	1.00 / **1.00** / **1.00** / 0

Multi-model codec-blind rerun. To check the codec-blind end-to-end result is not a single-model artifact, we re-run it on four model scales. No-defense ASR varies across models, but the code-density rule blocks $0/50$ attacks in each reported run, while the codec-aware gate blocks all 50 (Table 14).

F.2 ASR provenance and integrity

Because the end-to-end ASR depends on the execution runs, we guard model identity explicitly. Each rollout is saved only after a per-save check that the recorded `model` field matches the intended

Table 14: Codec-blind end-to-end ASR across model scales (locally benign, n=50, shared 5%-FPR). Each cell is ASR under no-defense / +per-step / +taint / +codec-blind / +codec-aware. The code-density rule leaves ASR at the no-defense level on every model, while the codec-aware gate drives it to 0. These rows use independently regenerated rollouts, so their no-defense rates should not be compared directly with Table 13.

Model	**none**	**+per-step**	**+taint**	**+codec-blind**	**+codec-aware**
Qwen2.5-7B	0.68	0.68	0.68	0.68	**0.00**
Llama-3.1-8B	1.00	1.00	1.00	1.00	**0.00**
Qwen2.5-14B	1.00	1.00	1.00	1.00	**0.00**
Qwen2.5-32B-AWQ	0.98	0.98	0.98	0.98	**0.00**

served model, and the runner refuses to save on a mismatch. All cells in Table 13 pass this check. Attack-success rates differ across models because compliance is limited by model capability. The same defenses miss the attack on every served model, under the fixed codec and observation views. We excluded Mistral-7B. Its no-defense ASR is near zero, so it cannot separate one defense from another.

F.3 NO-CODEC LLM SEARCH

This is the no-codec LLM search summarized in Section 5 and added as a row to Table 5. We ask whether a capable LLM can reconstruct the payload from the fragments without being handed the codec. The fixed common-decoder ensemble is the comparison.

Setup. We reuse the same persisted locally benign rollouts as the evaluated codec-blind setting (Collaborative Shadows, Qwen2.5-14B, K=3): 50 attack trajectories and 50 benign trajectories. For each trajectory the monitor sees the K fragment texts (with fragment indices) and nothing about the encoding. We prompt DeepSeek-V4-Flash at temperature 0, one call per trajectory, to (i) decide whether the fragments carry a hidden compositional payload and (ii) propose up to five candidate reconstructions, each a free-form decoded/assembled string. The *same* gate as the common-decoder ensemble scores every candidate, using the code-structure score $\max(\text{ASSEMBLES-TO-CODE}, 10 \times \text{CODE-DENSITY})$. A trajectory takes the maximum score over its candidates. The decision threshold is the 95th percentile of the benign trajectory scores (the same 5% benign-FPR operating point used elsewhere); benign trajectories pass through the identical prompt and gate.

Table 15: No-codec LLM search (DeepSeek-V4-Flash, no codec given), scored by the same gate as the common-decoder ensemble at the shared 5% benign-FPR point. Columns give the per-class score distribution. The last row is the LLM's own answer to whether the fragments carry a hidden payload, not a gate decision. At the 5%-FPR threshold (0.191) the gate blocks $\mathbf{6/50}$ attacks, with AUROC 0.521 (benign vs. attack), essentially chance.

	attack (n=50)	**benign** (n=50)
score mean	0.053	0.046
score max	1.000	1.538
nonzero scores	7/50	5/50
LLM suspicion rate	0.76	0.34

Result. The LLM calls 0.76 of the attack traces suspicious, but the gate blocks only $6/50$ of them (Table 15). The search ranks near chance (AUROC 0.521). The model also produces code-like guesses on benign traffic. One benign trajectory scores 1.538, above the highest attack score (1.000). The common-decoder ensemble blocks $0/50$, this LLM search blocks $6/50$, and the codec-aware gate blocks $50/50$.

G CODEC TRANSFER EXPERIMENT

Data. The construction reuses Collaborative Shadows carrier groups. For each episode a generator emits one Python program; an attack program reads a secret environment variable and posts it to an external host, and a benign program reads a non-secret variable and posts to an internal host. Identifiers, environment-variable names, hosts and paths are sampled per episode from class-specific lists. The two classes therefore differ in content as well as in program semantics. Individual fragments can carry information about the class. The program is split into $K=3$ parts at cut points chosen inside the environment-variable and host spans, so neither token lies wholly inside one fragment. Both classes are encoded through the same channel. Each benign source is fitted to the character length of the attack part it replaces. Fragment lengths therefore match position by position.

Codecs. Six encodings are evaluated. Three map each hexadecimal nibble to a word from a 16-word list (`cleanword16`, `nato` phonetic, and a syllable list). The other three use different units: `wordpair32` encodes five bits per word from a 32-word list, `base64` is standard base64, and `decimal` writes each byte as three digits. All six round-trip exactly, which is asserted at run time.

Detectors. The joint detector is a random forest over per-fragment statistics: length, token counts, type–token ratio, word-length moments, a top-16 token frequency profile, bigram repetition, entropy, digit and punctuation rates. It also uses cross-fragment terms (pooled type–token ratio, shared-vocabulary size and mass, pairwise Jaccard, cosine and token-profile distances, contiguous shared-token runs, length dispersion) and two ordering-sensitive seam terms. Agent role identity is not among the features used for this table. The full-text comparator is a logistic regression over tf-idf unigrams and bigrams of the concatenated fragments. A local comparator scores single fragments with logistic regression on the per-fragment features alone.

Protocol. Groups are split 50 train / 20 calibration / 30 test, with five seeds. In the *in-family* setting a detector trains on the evaluated codec; in the *held-out* setting it trains on the other five codecs only. In both settings, benign calibration episodes from the evaluated codec are available, and the operating threshold is the 95th percentile of those calibration scores. Test episodes are used only for the reported AUROC and the realized TPR and FPR. The local comparator is scored per fragment. The joint and full-text methods are scored per episode. They are also different classifiers, so a gap between their numbers does not measure the effect of seeing more fragments alone.

Results. Table 16 reports every codec and method. The spread is the standard deviation across the five seeds, not a standard error or a confidence interval. Realized test false-positive rates differ from the nominal 5% calibration target, and are reported so that TPR values can be read against the operating point actually achieved. This experiment measures detection scores on a synthetic construction; it does not measure end-to-end attack success and does not evaluate a monitor that runs online and keeps a history.

H ADDITIONAL STRESS TESTS

Table 18 gives the full sweep behind the 5% FPR operating point, and Table 19 the natural-language payload transfer. The benign-code stress test (Table 6) and the K-sweep (Table 7) are in Section 5. Perturbations that preserve execution, including wrappers, import aliasing, dead code, and string construction, did not change the observed blocking pattern: the codec-aware gate blocked every case, and the common-decoder ensemble blocked none. The marker-free score only ranks the two classes; it does not block attacks. In the adaptive truncation sanity check, the rate of attacks that both evade the monitor and still execute is 0.000 at every retained-code fraction f. This outcome is partly definitional. The same retained code content drives both the code-structure score and executability.

Table 16: Codec transfer, per codec and method. AUROC and TPR are means over five group splits; SD is the spread across those splits. "Test FPR" is the realized benign false-positive rate on the test split at the threshold calibrated on held-out benign episodes.

Codec	Method	AUROC	SD	TPR	Test FPR
cleanword16	Joint RF, in	0.972	0.024	0.820	0.047
	Joint RF, held out	0.966	0.022	0.793	0.080
	Full-text LR, held out	0.495	0.020	0.107	0.127
	Local pooled, held out	0.593	0.038	0.089	0.053
nato	Joint RF, in	0.971	0.017	0.820	0.053
	Joint RF, held out	0.972	0.028	0.740	0.027
	Full-text LR, held out	0.443	0.064	0.060	0.073
	Local pooled, held out	0.588	0.040	0.084	0.053
syllable	Joint RF, in	0.970	0.018	0.740	0.033
	Joint RF, held out	0.968	0.022	0.753	0.073
	Full-text LR, held out	0.500	0.000	0.153	0.153
	Local pooled, held out	0.582	0.041	0.098	0.062
wordpair32	Joint RF, in	0.883	0.064	0.633	0.140
	Joint RF, held out	0.501	0.051	0.107	0.093
	Full-text LR, held out	0.500	0.000	0.160	0.160
	Local pooled, held out	0.500	0.030	0.073	0.060
base64	Joint RF, in	0.643	0.061	0.187	0.060
	Joint RF, held out	0.500	0.001	0.073	0.067
	Full-text LR, held out	0.500	0.000	0.127	0.127
	Local pooled, held out	0.498	0.006	0.082	0.069
decimal	Joint RF, in	0.901	0.036	0.780	0.113
	Joint RF, held out	0.501	0.044	0.133	0.140
	Full-text LR, held out	0.276	0.209	0.080	0.167
	Local pooled, held out	0.412	0.046	0.042	0.100

Table 17: **Code-structure scores under benign code traffic.** Full version of Table 6, including AP and FPR at TPR 0.95.

Monitor	**benign mean**	**attack mean**	**AUROC**	**AP**	**FPR @TPR**0.95
Codec-blind code structure	0.897	0.600	0.090	0.510	0.960
Codec-aware gate	0.897	1.240	1.000	1.000	0.000

Table 18: Codec-aware gate operating points on the synthetic testbed (Collaborative Shadows carrier construction, 50 groups). Benign scores stay at most 0.120 and attack scores reach 1.240 on this construction. "Benign FPR" is the realized benign false-positive rate at that operating point.

operating point	**benign FPR**	**TPR**	**residual ASR**
nominal 5%-FPR[†]	0.06	1.000	0.000
any thr $\in (0.12, 1.24)$	**0.000**	**1.000**	**0.000**

[†]The nominal 5% target places the threshold inside the benign score range, so $3/50$ benign trajectories cross it and the realized FPR is 0.06. Nominal targets and realized rates differ throughout this paper.

Table 19: **Natural-language payload transfer.** Code-structure monitors do not transfer cleanly to a non-code harmful instruction. The one-class scores cover this single code/NL pair and include formatting artifacts. The code one-class (raw) value here (0.878) is for this pair; the 0.874 elsewhere is the one-class mean across the four wrapper variants.

Payload	**assembles-to-code**	**one-class (raw)**	**one-class (strict)**	**marker-based**
code	0.56 to 0.66	0.878	0.956	1.000
natural lang.	0.000	0.753[*]	**0.406**	1.000

[*]artifact: drops to 0.406 once formatting tokens are removed.

Adversarial assembly order and decoys. A distributed attacker need not present the fragments in order or in isolation. We permute the K fragments before assembly and interleave benign decoy fragments ($D \in \{1, 3, 5\}$). Neither change breaks the attack. A per-fragment index lets the codec-aware gate re-sort the pieces and ignore the decoys. Reconstruction and execution stay at 1.000. Detection does not degrade either. The marker-free one-class monitor holds at AUROC 0.873 under reordering, and its AUROC rises to 0.97 to 0.99 under decoys. The supervised monitor and the codec-aware gate stay at 1.000. Within our threat model, reordering the fragments and adding decoys therefore do not defeat the gate.