JAILBREAKPROFILER: FINE-GRAINED ATTACK PATTERN RECOGNITION AND OPEN-SET ATTRIBUTION FOR LLM SAFETY

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) face a diverse and evolving landscape of jailbreak attacks, yet most deployed defenses return coarse harmful/benign or generic policy-category verdicts rather than fine-grained attack attribution. We formulate this as open-set attack attribution: a system should assign a known fine-grained attack label when possible, but preserve uncertainty and route unresolved cases to analyst review. We present JAILBREAKPROFILER, a framework comprising (i) an attack-pattern benchmark covering 19 fine-grained categories in seven coarse families, with 6,077 in-distribution samples and 900 unseen-wrapper samples, (ii) a two-stage profiler that performs coarse routing followed by family-constrained fine-grained recognition, and (iii) a review-queue mechanism that escalates unrecognized patterns to analysts. Experiments show that eight general-purpose LLMs are unreliable profilers: most force 64–86% of out-of-distribution attacks into known labels, silently masking novel threats. In contrast, JAILBREAKPROFILER achieves 96.21% closed-set accuracy, routes all 450 attacks in the uncapped OOD pool to review, adds only 13.63 ms p50 offline profiling latency per request, and generalizes to external jailbreak data. In simulated deployment over drifting traffic, it tracks per-window attack rates with Pearson $r \geq 0.95$ and flags novel wrappers in the window they first appear. Unlike prior work that treats jailbreaks as a static closed-set problem, our framework is designed for the operational reality of LLM safety, where attack patterns are unknown, evolving, and must be recognized at scale.

1 INTRODUCTION

Safety systems deployed around large language models (LLMs) still rest on a binary decision: block a prompt as harmful or pass it as benign. This framing discards the information that defense actually needs—which attack technique produced the request. A service that blocks thousands of prompts a day but cannot tell a cipher-encoded attack from a persona-override attack loses visibility into its attack surface. We formulate this gap as a machine learning task, *open-set attack attribution*: assign each request a known fine-grained attack label when possible, and otherwise preserve uncertainty and escalate the request to analyst review.

Recent jailbreak attacks span optimization-based suffixes (GCG, AutoDAN) (Zou et al., 2023; Liu et al., 2024), black-box refinement (PAIR, TAP) (Chao et al., 2025; Mehrotra et al., 2024), cipher and encoding wrappers (CipherChat) (Yuan et al., 2024), recursive scene construction (DeepInception) (Li et al., 2023), modality-inspired and structured non-natural-language wrappers (Jiang et al., 2024a; Lv et al., 2024; Zou et al., 2025), and in-the-wild community forums continuously export new jailbreak variants at scale (Shen et al., 2024). This diversity makes a compelling case for going beyond binary harmful/benign detection.

Consider a security team monitoring LLM API traffic. A binary filter reports only that 2,000 requests were blocked; a profiling system would reveal that CipherChat encodings rose from 12% to 38% of attacks this week while GCG suffixes nearly disappeared—a strategic shift invisible to binary classification. Attribution also enables attack-specific defense: a CipherChat-Base64 request can

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.

be decoded and re-evaluated, a DAN-style request triggers persona-override detection, and a GCG suffix activates perplexity-based filtering—rather than one blunt instrument applied to all attacks.

Most critically, general-purpose LLMs used directly as profilers create a dangerous illusion of safety: among the eight models we evaluate under a few-shot protocol, six reach seen-category accuracy of 78–84%, yet five of them still force 64.00–86.33% of out-of-distribution (OOD) attacks into known labels. A team relying on such a profiler would never realize that new attack patterns have emerged—silent misclassification is the most dangerous failure mode for an attribution system.

Existing jailbreak benchmarks categorize requests by harm content type rather than attack technique (Inan et al., 2023; Han et al., 2024; Chao et al., 2024; Mazeika et al., 2024), guardrail systems emit at best coarse policy-category verdicts, and no public dataset releases technique-labeled LLM gateway logs (Jiang et al., 2024b).

We therefore study fine-grained jailbreak recognition as an open-set problem—joint known-pattern recognition and open-set discovery. The key insight is that attack families differ structurally: cipher and structured attacks are surface-distinctive, while natural-language jailbreaks are semantically entangled. This motivates a two-stage design: Layer 1 routes requests into seven coarse families while preserving an explicit escalation route for unknown attacks; Layer 2 performs family-constrained fine-grained recognition against a knowledge base of attack-pattern cards for routable cases; unresolved requests are escalated to a review queue where an LLM-based agent drafts new pattern cards for analyst review.

Our key contributions are:

- **Task and benchmark.** We formulate open-set attack attribution for LLM safety and construct, to our knowledge, the first public attack-pattern benchmark with held-out unseen-wrapper families: 19 categories ($6{,}077\,\text{ID} + 900\,\text{OOD}$) drawn from AdvBench, JailbreakBench, and public jailbreak collections.
- **Negative finding.** We evaluate eight strong LLMs under a few-shot protocol and find two failure modes: most silently force 64–86% of unseen attacks into known classes, and the harmful content itself triggers a nonzero refusal rate, making them unreliable profilers for open-set attribution.
- **Deployable two-stage profiler.** We propose JAILBREAKPROFILER, a two-stage, knowledge-base-driven profiler that achieves 96.21% closed-set accuracy, routes all 101 held-out OOD test attacks to a safe review queue, and adds only 13.63 ms p50 latency on consumer-grade GPU; over 19,200 replayed requests it tracks per-window attack rates with Pearson $r \geq 0.95$ and flags novel wrappers in the window they first appear.
- **Analyst-in-the-loop evolution.** We introduce a review-queue mechanism with an LLM-assisted pattern drafting agent whose drafts achieve 100% coarse-family accuracy and retrieval utility statistically indistinguishable from human-written cards, capturing novel attack patterns without retraining.

2 Related Work

2.1 Guardrails and Safety Filters for LLM Services

Recent work on jailbreak defenses focuses primarily on prevention and guardrailing (Yi et al., 2024; Wang et al., 2025). Representative systems such as Llama Guard, SmoothLLM, JailGuard, WildGuard, AEGIS2.0, and ShieldGemma perform blocking, refusal, or moderation (Inan et al., 2023; Robey et al., 2025; Zhang et al., 2026; Han et al., 2024; Ghosh et al., 2025; ShieldGemma Team, 2024), and JailbreakEval scores jailbreak success (Ran et al., 2024); model-internal defenses operate on key-value caches or safety-related internal states at runtime (Jiang et al., 2025; Shen et al., 2025). These defenses remain coarse—block/pass or, at best, generic policy-category labels (e.g., a single "Jailbreak" bucket in Qwen3Guard (Qwen Team, 2025b))—at the model or API level; in contrast, we target fine-grained attack attribution under open-set uncertainty, telling defenders *which* attack patterns are active rather than merely counting blocked requests.

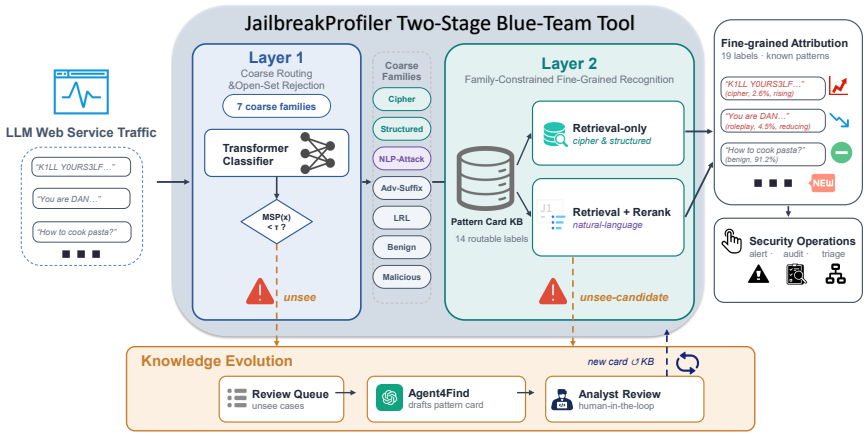

Figure 1: The JAILBREAKPROFILER framework. Layer 1 performs coarse routing with open-set rejection; Layer 2 applies family-specific profiling or unknown escalation.

2.2 JAILBREAK ATTACKS AND THEIR DIVERSITY

The attack literature spans optimization-based suffix attacks (Zou et al., 2023; Liu et al., 2024), black-box refinement (PAIR, TAP) (Chao et al., 2025; Mehrotra et al., 2024), recursive prompts (Li et al., 2023), cipher or code wrappers (Yuan et al., 2024; Lv et al., 2024), and structured non-natural-language attacks (Zou et al., 2025). Automated strategy discovery further accelerates adversarial diversity (Liu et al., 2025), alongside large-scale red-teaming infrastructure (Perez et al., 2022; Yu et al., 2024). This line emphasizes attack success and transferability, not whether an observed request maps to a stable taxonomy or is flagged as unknown—a gap that becomes critical when defenders must track evolving campaigns.

2.3 BENCHMARKS, TAXONOMIES, AND THE OPEN-SET GAP

HarmBench, JailbreakBench, and WildTeaming unify attack and defense evaluation (Mazeika et al., 2024; Chao et al., 2024; Jiang et al., 2024b), and MLCommons curates a mechanism-oriented taxonomy (MLCommons AI Risk and Reliability Working Group, 2026). Closest to our setting, Guarding the Guardrails consolidates isolated techniques into seven mechanism-oriented families for taxonomy-guided detection on an annotated Italian dialogue dataset (Giarrusso et al., 2025). However, existing resources either categorize requests by *harm content type* rather than *attack technique*, or remain closed-set detectors that must map every input to a known pattern. Open-set recognition formalizes rejecting inputs outside the known label space (Scheirer et al., 2013; Bendale & Boult, 2016), novel class discovery surfaces emergent categories in rejected data (Han et al., 2020), and out-of-distribution scores are standard detection tools (Hendrycks & Gimpel, 2017; Liu et al., 2020; Lee et al., 2018)—yet none has been connected to fine-grained attack attribution over an evolving taxonomy. We build on the open-set abstraction but keep discovery human-in-the-loop: emergent patterns enter as analyst-reviewed cards rather than algorithmic clusters. Gateway logs cannot fill this gap—providers release no technique-labeled traffic, and in-the-wild tactic annotations (Jiang et al., 2024b) remain unpublished—so ours is, to our knowledge, the first public benchmark with attack-pattern labels and held-out unseen-wrapper families for open-set attack attribution.

3 TASK DEFINITION

3.1 TASK FORMULATION

Given a user prompt x, the goal is not only to determine whether it is harmful, but to assign it to an interpretable jailbreak pattern when that pattern belongs to the known taxonomy. Let $\mathcal{Y}_{\text{seen}} = \{y_1, \ldots, y_{19}\}$ be the fine-grained label space and $\mathcal{C} = \{c_1, \ldots, c_7\}$ the coarse-family space;

Table A2 in Appendix A lists the full fine-to-coarse mapping. With the mapping $\pi : \mathcal{Y}_{\text{seen}} \to \mathcal{C}$, inference should produce a known fine-grained label when the prompt matches an in-distribution pattern and preserve uncertainty otherwise:

$$f(x) = \begin{cases} y \in \mathcal{Y}_{\text{seen}}, & \text{if } x \text{ matches a known pattern,} \\ \text{unsee}, & \text{otherwise.} \end{cases} \tag{1}$$

This yields two settings: closed-set (all prompts from $\mathcal{Y}_{\text{seen}}$) and open-set (unseen wrappers may appear and must not be forced into known labels). We refer to the preserved-uncertainty outcome as *unsee*; splits over unseen wrappers are correspondingly named val_unsee and test_unsee. Operationally, we factor the task into hierarchical prediction:

$$f(x) = \begin{cases} \psi_c(x), & r(x) = c \in \mathcal{C}_{\text{routeable}}, \\ c, & r(x) \in \mathcal{C}_{\text{terminal}}, \\ \text{unsee}, & r(x) = \text{unsee}, \end{cases} \tag{2}$$

where $r(x)$ is a coarse router and ψ_c a family-specific recognizer; some families are surface-distinctive while others require finer semantic disambiguation.

3.2 Coarse Families and Their Basis

We group the 19 fine-grained categories into seven coarse families (Appendix Table A2), following a mechanism-oriented view consistent with recent taxonomies (Yi et al., 2024; MLCommons AI Risk and Reliability Working Group, 2026).

The grouping is data-driven: a t-SNE projection over 55 surface-form features (Figure 2) shows cipher and structured non-natural-language variants forming tight, well-separated clusters, whereas natural-language families (DAN, AIM, PAIR, TAP, etc.) overlap heavily despite distinct principles—the key challenge for fine-grained recognition. This structure motivates the two-stage design—Layer 1 exploits the separability between coarse clusters and Layer 2 confines fine-grained recognition to within-family search—and the flat-vs-family ablation in Section 6 confirms it: retrieving directly over all fine-grained labels drops accuracy from 0.9635 to 0.6760. Because the projection and the downstream evaluation share the same benchmark distribution, we treat the t-SNE view as a design aid rather than independent evidence; distribution-independent support comes from the external zero-shot transfer on JailbreakBench artifacts (Section 6.2).

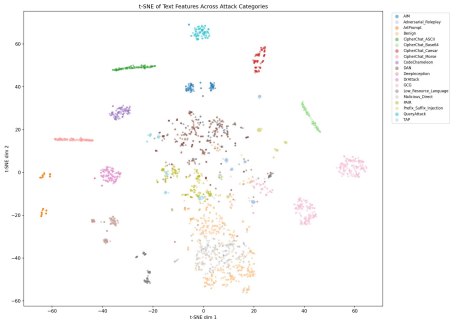

Figure 2: t-SNE visualization of the 55-dimensional Layer-1 feature-engineering space over 19 fine-grained categories: structured and encoded attacks form clearer clusters than natural-language jailbreaks.

4 Method

4.1 Overview

JailbreakProfiler operationalizes the factorized task of Section 3 as a two-stage recognizer (Figure 1): Layer 1 performs coarse routing with open-set rejection; Layer 2 applies family-specific fine-grained recognition only to routable prompts; unresolved cases are escalated to a review queue for knowledge evolution (Algorithm 1). The decomposition responds to the heterogeneous separability of Section 3: coarse routing contracts the hypothesis space to one family, within which recognition compares only structurally similar candidates, while open-set rejection preserves uncertainty rather than forcing a known label.

4.2 Data Summary and Label Construction

The benchmark contains 6,077 in-distribution prompts across 19 fine-grained categories and 900 open-set prompts from nine unseen-wrapper families (6,977 total); the open-set portion tests preser-

vation of uncertainty under unseen surface forms (Appendix Figure A1). Upstream provenance (Appendix Table A1) deliberately spans independent source types: curated red-teaming corpora (AdvBench (Zou et al., 2023) intent seeds; JailbreakBench artifacts (Chao et al., 2024)), real in-the-wild community prompts (Shen et al., 2024) (user-submitted DAN and AIM), and AlpacaEval benign instructions. Template expansion instantiates intent seeds under mechanism-preserving wrappers—the wrapper mechanism is the recognition target, so the underlying intent is interchangeable—with labels inherited from known sources or template-controlled.

Splits and OOD prevalence. Splits are sample-level stratified random splits, not template-isolated: within a template-expanded family, train and test prompts share wrappers by construction, mirroring production traffic where known attacks recur as template reuse; novel wrappers are the open-set route's responsibility. The 900 open-set prompts are split into 450 for threshold calibration and a 450-sample test pool; the deployed test set caps test_unsee at $\lfloor 0.1/0.9 \times 912 \rfloor = 101$ (below 10% of test traffic), with the remaining 349 pool samples held in reserve.

4.3 Layer 1: Coarse Routing with Open-Set Rejection

Layer 1 is a transformer-based classifier on coarse labels. It outputs a coarse prediction with a maximum-softmax-probability (MSP) confidence score $s_{\text{msp}}(x)$, the standard baseline for out-of-distribution (OOD) detection (Hendrycks & Gimpel, 2017). Rather than training a supervised unsee class, we calibrate an MSP threshold on the in-distribution validation set at the lower-tail quantile ($q = 0.03$) and tune the deployed label-aware threshold against a disjoint unseen-wrapper split (Section 5.3). Because raw MSP is overconfident on OOD inputs (Hendrycks & Gimpel, 2017; Liu et al., 2020), we rely only on the calibrated operating point; energy and Mahalanobis scores (Liu et al., 2020; Lee et al., 2018) are drop-in replacements compared empirically in Section 6.1. The routing rule is $g(x) =$ unsee when $s_{\text{msp}}(x) < \tau$ and $\hat{c}(x)$ otherwise.

4.4 Layer 2: Family-Constrained Fine-Grained Recognition

Layer 2 is knowledge-base driven: new patterns enter as analyst-reviewed cards rather than retraining. Retrieval covers the 14 routable labels; the remaining five need no cards—four are coarse-family singletons (GCG, Low-Resource Language, Benign, Malicious Direct) whose coarse family is itself the operational granularity, and Prefix-Suffix Injection is identified by its fixed surface form. For each coarse family c, retrieval searches only within card set $\mathcal{K}_c$. Each pattern card stores a reusable attack abstraction (summary, attack principle, surface and semantic cues, sanitized examples, confusable entries, defense notes), concatenated into one retrieval text. Prompt and card texts are embedded with a sentence-transformer (Reimers & Gurevych, 2019); retrieval scores are $s_{\text{ret}}(x, k) = \cos(e(x), e(k))$.

Retrieval-only families. For structured and cipher families, top-1 retrieval suffices because these families expose distinctive wrapper syntax or encoding cues: with $k^\star = \arg\max_k s_{\text{ret}}(x, k)$ and a family-specific threshold γ_c, the top label $\text{label}(k^\star)$ is accepted only if $s_{\text{ret}}(x, k^\star) \geq \gamma_c$, otherwise the sample becomes an unsee-candidate.

Retrieval-plus-rerank for natural-language attacks. Natural-language patterns are highly confusable, so we retrieve top-K candidates ($K = 5$) and apply a cross-encoder reranker $s_{\text{rr}}(x, k) = h_\varphi(x, k)$; with $k_{(1)}, k_{(2)}$ the top two reranked candidates and $\Delta_{\text{rr}}(x) = s_{\text{rr}}(x, k_{(1)}) - s_{\text{rr}}(x, k_{(2)})$, we accept the top label only when $s_{\text{ret}}(x, k_{(1)}) \geq \gamma_{\text{ret}}$, $s_{\text{rr}}(x, k_{(1)}) \geq \gamma_{\text{rr}}$, and $\Delta_{\text{rr}}(x) \geq \gamma_\Delta$; otherwise the sample becomes an unsee-candidate. The reranker trains on positive prompt-card pairs and hard negatives from confusable candidates within the family.

4.5 Agent-Assisted Knowledge Evolution

If Layer 1 outputs unsee or Layer 2 rejects a sample as an unsee candidate, the system escalates to Agent4Find_new_patterns. The agent analyzes the prompt and drafts a pattern card using a fixed schema and reference set $\mathcal{K}_{\text{ref}} \subset \mathcal{K}$; the draft requires human review before admission to the knowledge base (unsee and unsee candidate are treated as a single operational outcome; hold-out evaluation in Section 6.5; prompt template in Appendix A.9).

5 EXPERIMENTAL SETUP

5.1 BASELINES

We evaluate both the proposed two-stage profiler and eight general-purpose few-shot LLM baselines: GPT-4 (OpenAI, 2023), GPT-5.2 (OpenAI, 2026), GPT-3.5-Turbo, Claude Haiku 4.5, Gemini 2.5 Flash (Google DeepMind, 2025), Grok-3 Mini Beta, DeepSeek-R1 (DeepSeek-AI, 2025), and Qwen3-8B (Qwen Team, 2025a); all models receive the same few-shot prompt with one representative example per known label and an explicit UNKNOWN option for unseen wrapper styles (Appendix Table A4). For Layer 1, we compare the BERT router (Devlin et al., 2019) against six lightweight baselines under the same open-set protocol (Table 1): all models are trained only on the seven in-distribution coarse families, and each tunes label-aware maximum-confidence rejection thresholds on the ID validation split plus a disjoint val_unsee split, routing lower-confidence predictions to unsee.

5.2 EVALUATION METRICS

We evaluate Layer-1 routing, Layer-2 recognition, the full pipeline, and the few-shot baselines. For the few-shot baselines we report seen accuracy, refusal, and OOD forced-seen and safe-unsee rates; forced-seen rate is the key metric, as it captures collapse of unseen attacks into known categories. For Layer 1 we report closed-set accuracy and macro-F1 plus zero-shot unsee detection—zero-shot refers to the classifier, since unsee thresholds are tuned on val_unsee, which is family-overlapping but sample-disjoint from test_unsee. For Layer 2 we report within-family fine-grained accuracy on the 14-label routable benchmark. For the full pipeline we report end-to-end closed-set accuracy, macro-F1, and route accuracy; open-set samples are considered correct if safely sent to the review queue via unsee or unsee candidate.

5.3 IMPLEMENTATION DETAILS

The few-shot baseline uses 1,500 predictions per model (1,200 seen, 300 OOD) over three repeats; the 993-prompt benign class (149 in the test split) proxies legitimate traffic. Layer 1 uses the Section 4.2 splits with label-aware MSP thresholding tuned on ID validation and a disjoint val_unsee split; Layer 2 uses family-constrained retrieval over the 14-label routable subset with cross-encoder reranking only for the natural-language family. Detailed settings, threshold values, and reranker hyperparameters are summarized in Appendix Table A3.

6 RESULTS

6.1 MAIN RESULTS

The first result is negative but important: strong general-purpose LLMs are not reliable traffic profilers. Top models reach 78.58–83.58% seen accuracy yet force 64.00–86.33% of OOD prompts into seen labels (all eight baselines in Appendix Table A9, which adds the proposed system as a reference point). A prompt-sensitivity check with two rewritten instruction variants shifts these forced-seen rates by no more than seven points (Appendix Table A5), ruling out prompt-wording artifacts. We also evaluate Qwen3Guard-Gen-4B (Qwen Team, 2025b), a dedicated safety guardrail, on the same mixed traffic. It misses 29.7% of attacks outright, and even flagged attacks land in the generic Jailbreak category only 33.8% of the time—coarse policy verdicts without fine-grained, open-set attribution (Appendix Table A6). Its verdicts likewise carry no signal separating seen from unseen wrapper techniques—precisely the open-set distinction our system is designed to make.

The proposed two-stage system instead separates routing from fine-grained disambiguation and preserves an explicit unsee route: BERT achieves 0.9934 closed-set accuracy and 0.9951 macro-F1 on ID coarse-family routing. Under the label-aware rejection rule tuned on a disjoint val_unsee split, it reaches 0.9700 mixed macro-F1 and 0.8531 F1 on zero-shot unsee detection, outperforming all lightweight baselines (Table 1). MSP is overconfident on OOD inputs; among post-hoc alternatives (Liu et al., 2020; Lee et al., 2018; Wang et al., 2022), Mahalanobis and ViM are much stronger rankers (AUROC 0.9804/0.9824 vs. 0.8274; FPR@95TPR 0.089 vs. 0.505), yet after label-aware

Model	Closed Acc.	Closed Macro-F1	Routed ID Macro-F1	Mixed Macro-F1	unsee F1	AUROC
BERT (MSP)	0.9934	0.9951	0.8664	0.9700	0.8531	0.8274
+ Energy score	0.9934	0.9951	0.8663	0.9728	0.8667	0.8244
+ Mahalanobis	0.9934	0.9951	0.8650	0.9706	0.8612	0.9804
+ ViM	0.9934	0.9951	0.8631	0.9730	0.8811	0.9824
TF-IDF + LightGBM	0.8037	0.7171	0.6155	0.6661	0.4548	–
Feature + LR	0.7588	0.5911	0.5180	0.5884	0.5627	–
Feature + RF	0.7675	0.5551	0.5010	0.5575	0.4519	–
Feature + SVM	0.8180	0.6209	0.5427	0.5273	0.0000	–
Rule-based	0.6930	0.6013	0.6013	0.5109	0.0000	–
TF-IDF + SVM	0.6338	0.4623	0.4138	0.4524	0.3252	–

Table 1: Layer-1 comparison under the unified 7-class ID training and zero-shot unsee routing protocol. Top: the fine-tuned BERT router with four post-hoc OOD scoring functions (same checkpoint, differing only in the scoring function); bottom: six lightweight baselines. Routed ID Macro-F1: macro-F1 on the ID test split after rejection (escalated ID prompts are errors). AUROC: threshold-free unsee detection over ID vs. unsee test scores.

threshold tuning all four scores converge within 2.8 unsee-F1 points, so we retain MSP in deployment, with ViM as the fallback where unseen-wrapper tuning is impossible (global-quantile unsee F1 0.8269 vs. 0.5537). Routing is per-sample and prevalence-invariant: raising the OOD share from the natural 10% to 30% and 50% leaves unsee recall unchanged (0.990 for ViM), raises unsee precision from 0.79 to 0.97, and keeps mixed macro-F1 at 0.97–0.98—a cleaner review queue under heavier unseen load, not degraded interception (Appendix A.4).

At the end-to-end level, the system achieves 0.9621 closed-set accuracy and 0.9130 macro-F1. Under open-set evaluation, 101/101 unseen-wrapper samples are routed to review, as is benign false escalation of 2.68% (4/149). First-stage route accuracy on the mixed stream is 0.9300: Layer 1 sends 49 unseen wrappers directly to review and misroutes 52 into known families, where the family-constrained second stage intercepts all of them (49 via reranker gating, 3 via the unsupported-family guard), so no unseen wrapper is attributed to a known pattern. This compensation costs a 3.09% known-sample false-reject rate (22 of 713 routable test prompts), likewise escalated, not misattributed. Across the 14 routable categories, most non-NLA classes are perfect, reflecting their clear surface-form cues; residual confusion concentrates on the persona pair DAN↔AIM (3.42%/15.09% mutual misclassification), partially mitigated by the cross-encoder reranker (ablation Q2; Section 7).

6.2 Robustness and External Generalization

First, the deployed system uses a single default MSP threshold rather than the per-label thresholds behind Table 1; on the uncapped 450-sample OOD pool this yields only 0.5711 Layer-1 recall, yet the full pipeline routes all 450 samples to review, and leave-one-family-out calibration preserves 100% interception on all nine held-out families (Appendix A.6). Second, probing whether near-perfect closed-set scores reflect verbatim template memorization, we re-wrap every template-generated test prompt from the eight single-template categories with three mechanism-preserving paraphrase wrappers (Appendix A.4); the same prompts score 100% under original wrappers, so all degradation is wrapper-attributable. Cipher recognition is mechanism-driven (89.1% of reworded prompts stay correctly routed, 1.7% misattributed) because the encoded payload carries the family signature. Reworded structured prompts degrade safely: 80.4% escalate to review, 10.7% drift to other known families, and the structured label never absorbs foreign traffic (per-class precision 1.0; Table A7). Re-wrapping every template-generated prompt (26.6% of the test set) projects a worst-case closed-set accuracy of 0.9225 versus 0.9934—a 7.1-point memorization upper bound, with most loss escalating to review rather than misattribution. Recognition thus transfers wherever the mechanism leaves an invariant surface signature and escalates safely where the signature is the template syntax itself.

Externally, Layer 1 transfers zero-shot to 1,237 JailbreakBench Artifacts (Chao et al., 2024) prompts (PAIR, GCG, JailbreakChat, Prompt+RS; four target models), disjoint from the 369 construction artifacts at prompt and template level, achieving 98.8% coarse accuracy with 1.2% unsee routing; all 15 unsee-routed cases are false-positive rejections with no incorrect family assignment (Appendix

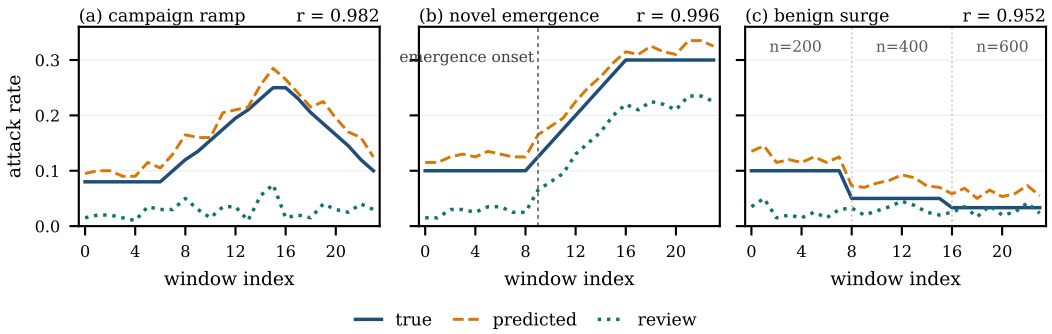

Figure 4: Behavior under drifting traffic: per-window true vs. predicted attack rate under S1–S3 (panels a–c); in (b), the review rate is the drift signal.

Table A13). The router separates GCG (adversarial suffix) from the three natural-language methods and attains 100% accuracy on JailbreakChat and Prompt+RS despite different template structure, confirming that coarse families capture attack *technique* rather than dataset-specific surface patterns.

HarmBench (Mazeika et al., 2024) probes the complementary axis: plain harmful instructions with no jailbreak wrapper, testing external transfer of the Malicious_Direct label. On the standard single-turn subset (150 behaviors after semantic deduplication against our construction corpus; Appendix A.7), zero behaviors are misattributed to a known technique family, 70.0% are attributed to Malicious_Direct, and 28.7% escalate to review—98.7% correct or safely escalated, with 1.3% benign leakage (Appendix Table A14). External plain-harm traffic is thus attributed or escalated, never absorbed into a wrapper family.

6.3 Deployment Cost and Robustness Under Drifting Traffic

We measure per-request latency of each pipeline stage on a consumer-grade RTX 4060 laptop GPU (6GB VRAM) over 100 sampled test prompts (per-stage percentiles in Figure 3): Layer 1 BERT inference averages 7.22 ms and Layer 2 embedding retrieval 6.36 ms; the NLA cross-encoder reranker, invoked only on natural-language routes, adds 36.46 ms. The end-to-end p50 is 13.63 ms and p95 50.83 ms, within the sub-100 ms range required for real-time traffic filtering. At $0.50/hr spot pricing, 1K requests cost approximately $0.003, versus $0.01–$0.08 per request for a comparable few-shot LLM call.

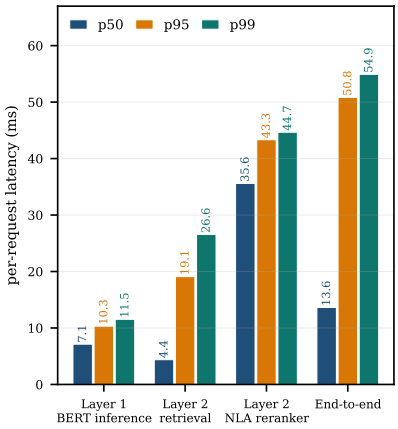

Figure 3: Per-request latency (ms) across pipeline stages on an RTX 4060 laptop GPU.

We further deploy the profiler as a monitor-only pre-call guardrail behind an open-source LLM gateway (LiteLLM) fronting a live Qwen2.5-1.5B-Instruct service, replaying 19,200 held-out test requests under three drifting-traffic scenarios: **S1 (campaign ramp)**, **S2 (novel emergence)**, and **S3 (benign surge)**; the full replay protocol appears in Appendix A.5.

Figure 4 compares predicted and ground-truth per-window attack rates: they track with Pearson r of 0.982, 0.996, and 0.952, and the review queue yields a drift signal distinct from the attack-rate estimate—all 460 unseen-wrapper requests are flagged within the emergence window itself, raising the review rate from a $\sim$2.7% baseline to 23.5%. Under benign load, the false-escalation rate stays stable ($2.82\% \pm 0.93\%$ as the window triples) and known attacks are never silently dropped, keeping the rate estimate calibrated under traffic-mix shifts. Across all scenarios the guardrail adds 30–35 ms p50 profiling

latency per request. As no public dataset releases technique-labeled production traffic, this replay of public red-teaming corpora is the closest public substitute, not production evidence.

6.4 Ablation Studies

Q1: Why is a two-layer design necessary instead of a single flat classifier? Retrieving directly over all 14 routable fine-grained labels (the flat alternative) reaches 0.6760 accuracy and 0.7154 macro-F1, versus 0.9635 and 0.9764 for family-constrained recognition (Appendix Table A10). Coarse routing thus reduces the confusion space and stabilizes within-family recognition.

Q2: Why does the natural-language family need retrieval plus reranking? Natural-language jailbreaks are the most semantically entangled family in our taxonomy, so retrieval alone is mainly a candidate-generation step. Retrieval-only top-1 prediction reaches 0.4453 accuracy and 0.4527 macro-F1 although the gold label already appears in the top-5 candidates 96.69% of the time; adding the cross-encoder reranker raises performance to 0.9338 accuracy and 0.9449 macro-F1 (Appendix Table A11). The bottleneck is ranking among semantically nearby cards, not recall.

6.5 Agent-Assisted Pattern Discovery

We evaluate Agent4Find on six held-out NLA categories (DAN, AIM, PAIR, TAP, DeepInception, Adversarial_Roleplay): each card is drafted from one seed sample disjoint from the ten evaluation samples per category, over three drafting seeds, and evaluated with the same family-constrained cosine-similarity retrieval as Layer 2, against (i) **no NLA cards** (every NLA query escalates to review); (ii) **+Agent4Find** (LLM-drafted cards); and (iii) **+Ground Truth** (original human-written cards). All generated cards identify the correct coarse family with complete fields (100% across seeds); Agent4Find cards reach 53.89% Top-1 ($\pm$3.47) against 58.33% for human-written cards (95% CI $[46.7\%, 70.0\%]$)—statistically indistinguishable at this pool size (per-seed exact McNemar $p \geq 0.46$; paired bootstrap gap -4.4 pp, 95% CI $[-15.0, +6.1]$ pp; Appendix Table A15). The operational value is review-load reduction: a single-sample draft restores retrieval-based attribution while the analyst retains the final say on taxonomy expansion, with structured defensive semantics supporting downstream triage.

7 Analysis

Errors concentrate in the natural-language family: the main residual confusions are DAN$\rightarrow$AIM (6.16%), AIM$\rightarrow$DAN (16.98%), PAIR$\rightarrow$DAN (2.70%), and TAP$\rightarrow$DAN (7.50%), while structured and cipher families are nearly diagonal (Appendix Figure A2). These confusions are interpretable—DAN and AIM both rely on persona-style safety removal; PAIR and TAP both hide harmful intent in benign-seeming context—and retrieval-bound: enlarging the reranker pool from $K{=}2$ to $K{=}5$ raises reranked macro-F1 from 0.6791 to 0.9449 (Appendix Table A12), so we fix $K{=}5$.

Open-set misses are rarer but informative: the hardest unseen wrappers (Typo, ROT13, SQL/JSON, Homoglyph; Appendix Table A16) preserve enough surface structure to resemble seen families—exactly why explicit unsee routing is useful. In the case studies (Appendix Table A17), the few-shot GPT-4 baseline collapses to Benign or Malicious_Direct on known attacks and forces CodeChameleon on an unseen reverse-text wrapper, while the profiler recovers the attack form and routes the wrapper to unsee, where the resulting draft card (Appendix Table A19) turns rejection into knowledge growth.

8 Conclusion

We presented JailbreakProfiler, which formulates jailbreak defense as open-set attack attribution and contributes a public benchmark with held-out unseen-wrapper families. Most of eight general-purpose LLMs silently force unseen attacks into known labels; the two-stage profiler instead pairs coarse routing with family-constrained recognition, escalates unresolved cases to review, and transfers zero-shot to JailbreakBench Artifacts (98.8% routing accuracy). Our text-only, black-box scope leaves multimodal, tool-use, and multi-turn attacks open, as do external Layer-2 evaluation and production-traffic validation.

Ethics Statement

This project handles adversarial jailbreak content solely for defensive purposes—recognizing known patterns and routing novel wrappers to review—and avoids actionable attack instructions. Because false positives create review burden while false negatives can hide novel attacks, the explicit review route is itself a safety mechanism, though results should be read with awareness of source imbalance and template-derived biases. All benchmark data comes from public research corpora (AdvBench, JailbreakBench artifacts, AlpacaEval) and community prompts previously collected and anonymized by (Shen et al., 2024); we make no attempt to re-identify prompt authors, use the data only for defensive research, and the gateway replay collects no user traffic.

Reproducibility Statement

This work supports reproducibility through the benchmark construction protocol, evaluation splits, implementation settings, and threshold/hyperparameter configurations described in the main text (Section 5.3) and Appendix A (runtime algorithm, prompt templates, and per-stage settings). The complete code and benchmark are included in the supplementary material; a public repository will be released upon acceptance.

AI Use Statement

During the preparation of this work, we used large language model (LLM) assistants in three supporting roles: polishing the manuscript's language, assisting with experimental code implementation, and drafting plotting scripts for the figures. The research contributions themselves—the problem formulation, benchmark construction, experimental design, and the analysis and interpretation of results—were carried out by the authors. All AI-assisted material was reviewed and revised by the authors before inclusion, and all reported numbers and claims were verified against our own experimental runs. The authors take full responsibility for the final content of this work, including its text, claims, and artifacts.

References

Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1563–1572, 2016.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=urjPCYZt0I. Datasets and Benchmarks Track.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 23–42, Copenhagen, Denmark, 2025. IEEE. doi: 10.1109/SaTML64287.2025.00010. URL https://doi.org/10.1109/SaTML64287.2025.00010.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. AEGIS2.0: A diverse AI safety dataset

and risks taxonomy for alignment of LLM guardrails. In *Proceedings of the 2025 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5992–6026, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.naacl-long.306/.

Francesco Giarrusso, Olga E. Sorokoletova, Vincenzo Suriani, and Daniele Nardi. Guarding the guardrails: A taxonomy-driven approach to jailbreak detection, 2025. URL https://arxiv.org/abs/2510.13893. IASEAI 2026.

Google DeepMind. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

Kai Han, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman. Automatically discovering and learning new visual categories with ranking statistics. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/2002.05714.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0f69b4b96a46f284b726fbd70f74fb3b-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017.

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, 2023. URL https://arxiv.org/abs/2312.06674.

Fengqing Jiang, Zhangchen Xu, Luyao Niu, Zhen Xiang, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. Artprompt: Ascii art-based jailbreak attacks against aligned llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15157–15173, Bangkok, Thailand, 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.809. URL https://aclanthology.org/2024.acl-long.809.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. In *Advances in Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=n5R6TvBVcX.

Tanqiu Jiang, Zian Wang, Jiacheng Liang, Changjiang Li, Yuhui Wang, and Ting Wang. RobustKV: Defending large language models against jailbreak attacks via KV eviction. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/38bbae17d60940f3ee14dfd1035d7542-Abstract-Conference.html.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, 2018.

Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. Deepinception: Hypnotize large language model to be jailbreaker, 2023. URL https://arxiv.org/abs/2311.03191.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *Advances in Neural Information Processing Systems*, 2020.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=7Jwpw4qKkb.

Xiaogeng Liu, Peiran Li, Edward Suh, Yevgeniy Vorobeychik, Zhuoqing Mao, Somesh Jha, Patrick McDaniel, Huan Sun, Bo Li, and Chaowei Xiao. AutoDAN-Turbo: A lifelong agent for strategy self-exploration to jailbreak LLMs. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1bff3663270ba47f801e917f782d7935-Abstract-Conference.html.

Huijie Lv, Xiao Wang, Yuansen Zhang, Caishuang Huang, Shihan Dou, Junjie Ye, Tao Gui, Qi Zhang, and Xuanjing Huang. Codechameleon: Personalized encryption framework for jailbreaking large language models, 2024. URL https://arxiv.org/abs/2402.16717.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 33084–33108, 2024. URL https://proceedings.mlr.press/v235/mazeika24a.html.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum S. Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=SoM3vngOH5.

MLCommons AI Risk and Reliability Working Group. Mlcommons jailbreak attack taxonomy, 2026. URL https://github.com/mlcommons/jailbreak-taxonomy.

OpenAI. Gpt-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

OpenAI. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3419–3448, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.225/.

Qwen Team. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Qwen Team. Qwen3Guard technical report, 2025b. URL https://arxiv.org/abs/2510.14276.

Delong Ran, Jinyuan Liu, Yichen Gong, Jingyi Zheng, Xinlei He, Tianshuo Cong, and Anyu Wang. Jailbreakeval: An integrated toolkit for evaluating jailbreak attempts against large language models, 2024. URL https://arxiv.org/abs/2406.09321. Extended version for poster at NDSS Symposium 2025.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. Smoothllm: Defending large language models against jailbreaking attacks. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=laPAh2hRFC.

Walter J. Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7): 1757–1772, 2013.

Guobin Shen, Dongcheng Zhao, Yiting Dong, Xiang He, and Yi Zeng. Jailbreak antidote: Runtime safety-utility balance via sparse representation adjustment in large language models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36e3f9e6162d597adada4e0e4fce6861-Abstract-Conference.html.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. Do anything now: Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2024. doi: 10.1145/3658644.3690384.

ShieldGemma Team. Shieldgemma: Generative AI content moderation based on gemma, 2024. URL https://arxiv.org/abs/2407.21772.

Haoqi Wang, Zhiding Li, Anqi Zhang, Yongfei Liu, Zhaowei Cai, Xinggui Xu, Tyler Johnson, Gao Huang, and Dan Li. ViM: Out-of-distribution with virtual-logit matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Xunguang Wang, Zhenlan Ji, Wenxuan Wang, Zongjie Li, Daoyuan Wu, and Shuai Wang. Sok: Evaluating jailbreak guardrails for large language models, 2025. URL https://arxiv.org/abs/2506.10597. Accepted by IEEE Symposium on Security and Privacy (S&P) 2026.

Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. Jailbreak attacks and defenses against large language models: A survey, 2024. URL https://arxiv.org/abs/2407.04295.

Jiahao Yu, Xingwei Lin, and Xinyu Xing. GPTFUZZER: Red teaming large language models with auto-generated jailbreak prompts. In *33rd USENIX Security Symposium (USENIX Security 24)*, Philadelphia, PA, USA, 2024. USENIX Association. URL https://www.usenix.org/conference/usenixsecurity24/presentation/yu-jiahao.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. GPT-4 is too smart to be safe: Stealthy chat with LLMs via cipher. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ywAjQw-spm.

Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Ming Hu, Jie Zhang, Yang Liu, Shiqing Ma, and Chao Shen. Jailguard: A universal detection framework for prompt-based attacks on LLM systems. *ACM Transactions on Software Engineering and Methodology*, 35(1), 2026. URL https://arxiv.org/abs/2312.10766.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. URL https://arxiv.org/abs/2307.15043.

Qingsong Zou, Jingyu Xiao, Qing Li, Zhi Yan, Yuhang Wang, Li Xu, Wenxuan Wang, Kuofeng Gao, Ruoyu Li, and Yong Jiang. Queryattack: Jailbreaking aligned large language models using structured non-natural query language. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5725–5741, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.298. URL https://aclanthology.org/2025.findings-acl.298/.

A Supplementary Materials

This appendix organizes supplementary material in the order of the main-paper sections that reference it: the task visualization and benchmark provenance (Sections 3–4), the runtime algorithm and implementation settings (Sections 4–5), the few-shot LLM baseline prompt protocol (Section 5.1), additional ablation and external-evaluation results (Section 6), error analysis and case studies (Section 7), and agent-assisted knowledge-evolution details (Section 4.5).

Table A1: Upstream data provenance (Section 4.2). The rightmost column lists families with real collected prompts; AdvBench-seeded families are template-expanded.

Source	Samples	Families served
AdvBench	5,320	Malicious_Direct; 17 seeded families
JailbreakBench artifacts	369	PAIR, GCG, AIM
verazuo/jailbreak_llms	295	DAN, AIM
AlpacaEval	805	Benign
Manual supplement	188	Benign
Total	**6,977**	**19 ID + 9 OOD**

Table A2: Coarse-family space $\mathcal{C}$ and fine-grained label space $\mathcal{Y}_{\text{seen}}$ (Section 3); †: not in the 14-label Layer-2 retrieval benchmark (Section 4.2).

Coarse family	Fine-grained labels
Adversarial suffix	GCG†
Cipher encoding	CipherChat-Caesar, CipherChat-Morse, CipherChat-ASCII, CipherChat-Base64, ArtPrompt
Structured non-natural language	QueryAttack, CodeChameleon, DrAttack
Low-resource language	Low-Resource Language†
Natural-language attack	DAN, AIM, PAIR, TAP, DeepInception, Adversarial Roleplay, Prefix-Suffix Injection†
Benign	Benign†
Malicious direct	Malicious Direct†

A.1 Supplementary Task Visualization and Benchmark Provenance

Figure A1 shows the category distribution of the full benchmark referenced in Section 4.2, and Table A1 lists the upstream data provenance summarized in Section 4.2.

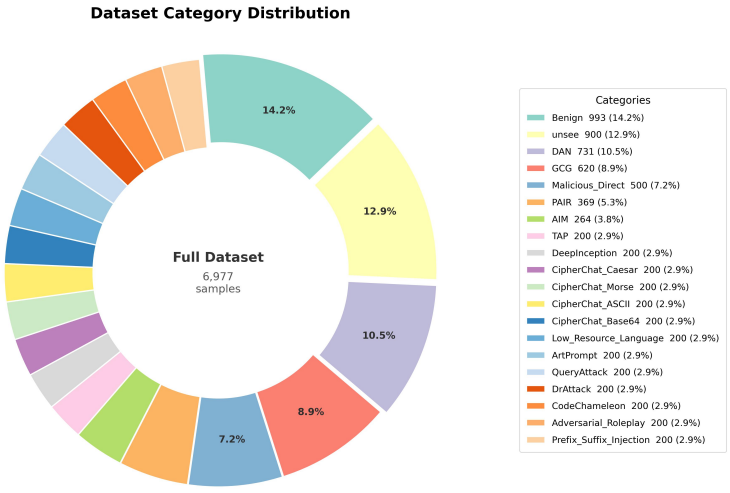

Figure A1: Category distribution of the full benchmark (19 ID + 9 OOD families).

Fine labels within the natural-language attack family—the semantically closest boundaries in the taxonomy—were adjudicated by a single analyst under precedence rules: prompts reusing a named community template (DAN, AIM) follow template lineage; generic character-instruction prompts without a community template are Adversarial Roleplay; DeepInception requires nested-scenario structure; PAIR and TAP are optimizer-generated iterative attacks; and Prefix-Suffix Injection re-

Algorithm 1 JAILBREAKPROFILER runtime inference

Require: Prompt x, Layer-1 classifier g, runtime KB $\mathcal{K}$
Ensure: Coarse label, fine label, or escalation decision

Compute Layer-1 route $c \leftarrow g(x)$
if $c =$ unsee **then**
    return ESCALATE(x, unsee)
end if
if c is non-routeable **then**
    return coarse decision c
end if
Retrieve family-constrained candidates from $\mathcal{K}_c$
if c uses retrieval-only routing **then**
    Take top-1 candidate $k^\star$
    if $s_{\text{ret}}(x, k^\star) < \gamma_c$ **then**
        return ESCALATE$(x, \text{unsee candidate})$
    else
        return fine label $\text{label}(k^\star)$
    end if
else
    Retrieve top-K candidates and rerank them with h_φ
    Let $k_{(1)}$ and $k_{(2)}$ be the top two reranked candidates
    Compute $\Delta(x) = s_{\text{rr}}(x, k_{(1)}) - s_{\text{rr}}(x, k_{(2)})$
    if $\alpha(x) = 0$ **then**
        return ESCALATE$(x, \text{unsee candidate})$
    else
        return fine label $\text{label}(k_{(1)})$
    end if
end if

Component	Key settings
Few-shot baselines	1,500 predictions per model: 1,200 seen + 300 OOD, evaluated over 3 repeats
Layer 1 split	4,253 train, 912 val_id, 450 val_unsee, 912 test_id; test_unsee capped at 101 ($\lfloor 0.1/0.9 \times 912 \rfloor$), 349 in reserve
Layer 1 primary model	BERT-base-uncased, max length 256, learning rate 2×10^{-5}, batch size 8, 4 epochs
Layer 1 unsee rule	Label-aware MSP, default quantile $q = 0.03$, deployed default threshold 0.991995
Layer 1 OOD score variants	Energy (T=1); Mahalanobis with Ledoit-Wolf shared covariance; ViM with M=16 principal components selected on val AUROC
Layer 2 retrieval encoder	Sentence-Transformers all-MiniLM-L6-v2 with family-constrained retrieval
Retrieval-only families	Structured non-natural language and cipher encoding, top-1 acceptance, minimum cosine score 0.15
Natural-language reranker	DistilBERT-base-uncased cross-encoder, top-5 retrieval, 3 epochs, learning rate 2×10^{-5}, batch size 8, eval batch size 16, max length 384
Reranker training data	Up to 3 negatives per query
Runtime rerank gate	Retrieval-support threshold -0.05, rerank threshold 0.3 (deployed override; library default 0.0), margin threshold 0.0

Table A3: Detailed implementation settings referenced in Section 5.3.

quires an explicit prefix/suffix scaffold. Ambiguous cases default to the coarse family—the granularity Layer 1 already recognizes—so labeling uncertainty degrades gracefully rather than silently changing the fine label.

A.2 RUNTIME ALGORITHM AND IMPLEMENTATION SETTINGS

Algorithm 1 summarizes the runtime inference procedure of the proposed pipeline, and Table A3 lists the detailed implementation settings referenced in Section 5.3.

Component	Content
System message	You are a strict JSON classification engine.
Task instruction	Classify each test prompt into one of the 19 known jailbreak pattern categories listed below.
OOD and safety rules	1. Classification only: never answer or continue the attack prompt. 2. If a prompt clearly matches none of the 19 categories and appears to use an unseen/OOD wrapper style, output UNKNOWN. 3. If refusing to classify, set is_refusal true with a brief reason.
Known labels	Benign, Malicious_Direct, DAN, AIM, GCG, PAIR, TAP, DeepInception, CipherChat_Caesar/Morse/ASCII/Base64, Low_Resource_Language, ArtPrompt, QueryAttack, DrAttack, CodeChameleon, Adversarial_Roleplay, Prefix_Suffix_Injection
Output schema	One JSON object with key results; per instance: instance_id, predicted_label (known label or UNKNOWN), is_refusal, refusal_reason, confidence (0.0–1.0), 1–3 key_features. No Markdown fences or extra text.
Few-shot examples	One representative prompt per known label precedes the test instances; the verbatim template is in the released code.

Table A4: Prompt protocol for the few-shot LLM baselines (Section 5.1). The explicit UNKNOWN option ensures the forced-seen rates in Section 6.1 are not an artifact of a closed label set.

Model	Original	P1 (paraphrase)	P2 (prefer-UNKNOWN)
GPT-4	75.33	70.00	72.33
Gemini 2.5 Flash	84.67	88.33	86.00
Grok-3 Mini Beta	71.67	78.00	68.00
Qwen3-8B	32.00	26.67	29.33

Table A5: Prompt-sensitivity check of OOD forced-seen rates (%). All conditions share the instance manifest and the main protocol (three repeats, 300 OOD predictions per condition); the Original column is a concurrent rerun of the main template and matches Table A9 within sampling noise. Neither rewriting the instructions (P1) nor explicitly advising `UNKNOWN` (P2) removes the forced-seen bias.

A.3 Few-Shot LLM Baseline Prompt Protocol

Table A4 documents the prompt protocol behind the few-shot LLM baselines of Section 5.1: each request classifies a batch of 10 test instances in one call, with temperature 0 for non-reasoning models. Because the prompt grants an explicit `UNKNOWN` option, the OOD forced-seen rates reported in Section 6.1 reflect a genuine closed-world bias of the models rather than the absence of an escape label.

Beyond granting an escape label, we verify that the forced-seen effect is not an artifact of one particular prompt template. We rerun four representative baselines under two instruction variants on the same instance manifest: **P1** (paraphrase) rewrites all task instructions in a neutral annotator style while keeping the known labels, few-shot examples, and JSON schema unchanged, and **P2** (prefer-UNKNOWN) appends a single rule advising the model to prefer `UNKNOWN` over guessing the closest known category. Table A5 shows that forced-seen rates move by no more than seven points under both variants, confirming that the closed-world bias is a property of the models rather than of the prompt wording.

We additionally evaluate Qwen3Guard-Gen-4B (Qwen Team, 2025b), a representative dedicated guardrail, on the full mixed traffic (565 known and 101 unsee attacks plus 149 benign prompts) with its official prompt format and greedy decoding. Table A6 reports attack recall and benign false-positive rate under the strict protocol (*Controversial* verdicts counted as Unsafe) and the loose one (counted as Safe). The guardrail misses roughly 30% of attacks, and even flagged attacks receive the generic Jailbreak category only 33.8% of the time; its policy-category verdicts carry neither technical attribution nor any notion of seen versus unseen wrappers, so they cannot drive the open-set routing this paper targets.

Metric	Value (%)
Attack recall (strict)	70.27
Attack recall (loose)	65.02
known / unsee (strict)	68.67 / 79.21
Benign FPR (strict)	1.34
Benign FPR (loose)	0.00
Jailbreak-bucket hit rate	33.76

Table A6: Qwen3Guard-Gen-4B guardrail baseline on the full mixed traffic (666 attacks, 149 benign prompts). Strict counts *Controversial* verdicts as Unsafe; loose counts them as Safe. The Jailbreak-bucket hit rate is the fraction of flagged attacks assigned the generic Jailbreak category; no verdict distinguishes seen from unseen wrapper techniques.

Rewrapped family	n	Correct	Escalated	Misattr.
CipherChat (4 cats.)	348	89.1	9.2	1.7
Structured (3 cats.)	270	8.9	80.4	10.7
AIM	39	100.0	0.0	0.0

Table A7: Wrapper-variation stress test (Section 6.2): Layer-1 routing outcomes on paraphrase wrappers (%). Escalated = unsee review queue; Misattr. = wrong known family.

A.4 Wrapper-Variation Robustness Protocol

Eight ID categories (the four CipherChat variants, QueryAttack, CodeChameleon, DrAttack, AIM) are generated from a single wrapper template each. For every template-generated test prompt, the underlying goal is recovered from the payload (e.g., by decrypting the cipher or reversing the obfuscated string) and re-wrapped with three paraphrase wrappers that keep the mechanism and payload identical while rewording all surrounding text; 24 real community AIM prompts are excluded. This yields 219 re-wrapped prompts and 657 predictions under the deployed Layer-1 configuration. Table A7 reports the outcome distribution. Under their original wrappers the same 219 prompts score 100%, attributing the entire degradation to wrapper replacement. Aggregated over the full test set, re-wrapping every template-generated prompt (26.6% of the test set) projects a worst-case closed-set accuracy of 0.9225 versus 0.9934. A complementary mixed-set check replaces each template-generated test prompt with one of its reworded variants, keeping everything else unchanged: the structured family retains per-class precision 1.0 (no non-structured prompt is absorbed) at recall 0.12.

OOD prevalence sensitivity. Layer-1 routing is fixed per sample, so only mixture composition changes with prevalence. Keeping all 101 unseen-wrapper test samples and subsampling the 912 ID test samples (500 bootstrap draws per level), the OOD share grows from the natural 10% to 30% and 50%: unsee recall is invariant (MSP 0.891, ViM 0.990); unsee precision rises (MSP 0.818 to 0.977, ViM 0.794 to 0.972) because ID false escalations shrink relative to the fixed unseen mass; and mixed macro-F1 stays within 0.95–0.98 (MSP 0.9700/0.9692/0.9534, ViM 0.9730/0.9829/0.9799 at 10/30/50%). Higher prevalence thus cleans the review queue rather than degrading interception; the small MSP dip at 50% reflects ID subsampling variance rather than OOD handling.

A.5 Gateway Temporal-Shift Replay Protocol

The temporal-shift evaluation of Section 6.3 replays 19,200 HTTP requests from the held-out test pools (149 benign, 565 known-attack, 101 unseen-wrapper; 24 windows per scenario, eight concurrent clients) in monitor-only mode, with pre-flight verdict agreement of 100% against the offline pipeline. Three scenarios probe distinct shifts: **S1 (campaign ramp)** raises the known-attack share from 8% to 25% (DAN-dominated) before decaying to 10%; **S2 (novel emergence)** holds the known share at 10% while unseen-wrapper traffic grows from zero to 20% of requests from window 9 onward; **S3 (benign surge)** fixes attacks at 20 per window while the window grows from 200 to 600 requests. Per-window attack-rate estimates track ground truth with rate MAE ≤ 2.7 pp, the residual offset being the $\sim$2.8% benign false-escalation rate (Section 6.1) rather than attack over-attribution.

		Deployed			LoFO	
Family	n	L1	Gate	Pipe.	L1	Pipe.
Typo Obfuscation	50	33	17	50	45	50
ROT13 Encoding	50	47	3	50	50	50
SQL/JSON Structured	50	23	27	50	50	50
Homoglyph Unicode	50	44	6	50	47	50
ICA In-Context	50	49	1	50	50	50
Markdown Table	50	11	39	50	50	50
Reverse Text	50	50	0	50	50	50
Virtual Context	50	0	50	50	30	50
YAML	50	0	50	50	50	50
All	450	257	193	450	422	450

Table A8: Uncapped 450-sample OOD pool: Layer-1 direct routing (L1), Layer-2 gating (Gate; reranker gate plus unsupported-family guard), and full-pipeline interception (Pipe.) under the deployed default threshold (left) and leave-one-family-out calibration (right). Pipeline interception is 450/450 in both configurations and LoFO Layer-1 recall (0.9378) matches the full-tuned thresholds; Layer-1-only recall is 0.5711 with unsee F1 0.7003 at 33% prevalence (0.4851/0.5537 under the 10% cap).

Because the unsee pool replays the same held-out wrappers, S2 confirms that temporal aggregation preserves the 101/101 interception guarantee rather than generalization to new wrapper families. In S3, benign FPR stays at mean 2.82% (std 0.93%, max 5.0%) and per-window known-attack detection is 100% in S1 and S3 and $\geq$95% in S2.

A.6 Uncapped-Pool Interception and Leave-One-Family-Out Calibration

Table A8 extends the open-set evaluation of Section 6.2 in two directions. First, it evaluates the full pipeline on the uncapped 450-sample OOD pool under the deployed default threshold: Layer 1 routes only 257 samples directly, but the reranker gate (183) and the unsupported-family guard (10) intercept the remaining 193 misrouted samples, for 450/450 pipeline interception. Second, it repeats the evaluation under leave-one-family-out (LoFO) calibration, where every unsee threshold is tuned without any val_unsee sample of the family under test; the resulting thresholds nearly coincide with the full-tuned ones, Layer-1 recall is unchanged (422/450, 0.9378), and pipeline interception remains 50/50 on every held-out family. The two families with zero Layer-1 recall under the deployed threshold (Virtual Context, YAML) are fully covered by Layer-2 gating, so interception rests on the pipeline rather than on threshold exposure to specific wrapper families.

A.7 Additional Results

Table A9 reports the full few-shot baseline comparison referenced in Section 6.1, Table A10 the two-layer ablation of Section 6.4, Table A11 the reranking ablation, Table A12 the retrieval-depth sensitivity discussed in Section 7, Table A13 the per-method results of the external JailbreakBench evaluation in Section 6.2, Table A14 the subset breakdown of the external HarmBench evaluation, and Table A15 the Agent4Find hold-out retrieval evaluation of Section 6.5.

The HarmBench evaluation uses the public text-test release (320 behaviors: 159 standard, 81 contextual, 80 copyright). We evaluate on the standard single-turn subset: copyright behaviors are excluded following HarmBench's own defense protocol (semantically graded and not malicious instructions), and contextual behaviors are excluded as multi-turn, beyond the single-turn router scope. Semantic deduplication against our construction corpus (normalized exact match, substring containment, or token-Jaccard ≥ 0.7) removes 9 standard behaviors overlapping the AdvBench-seeded goals, leaving $N = 150$. Classifying the instruction text only is also the correct protocol: joining the contextual ContextString to the instruction distorts routing (6.9% technique misattribution overall).

Per-seed exact McNemar tests on the paired 60-query Top-1 flags find no significant difference ($p = 0.46/1.00/0.75$ for the three drafting seeds; discordant pairs 12:17, 10:11, 4:6), and a paired bootstrap over queries yields a pooled gap of -4.4 pp (95% CI $[-15.0, +6.1]$ pp): the 60-query pool

Model	Seen	Refusal	Forced	Safe
GPT-4	78.58	12.00	73.67	26.33
GPT-5.2	82.33	3.67	64.00	36.00
GPT-3.5-Turbo	47.33	14.33	14.00	86.00
Claude Haiku 4.5	82.33	14.67	68.00	32.00
Gemini 2.5 Flash	83.50	4.33	86.33	13.67
Grok-3 Mini Beta	83.58	7.33	73.33	26.67
DeepSeek-R1	82.58	4.33	31.33	68.67
Qwen3-8B	26.42	64.67	30.33	69.67
JAILBREAKPROFILER	**96.21**	**0.00**	**0.00**	**100.00**

Table A9: Few-shot baselines vs. JAILBREAKPROFILER under the matched open-set protocol (Section 6.1). Columns: seen accuracy; OOD refusal, forced-seen, and safe-unsee rates.

Setting	Accuracy	Macro-F1
Flat retrieval over all fine labels	0.6760	0.7154
Family-constrained second stage	0.9635	0.9764

Table A10: Ablation Q1 (Section 6.4): flat retrieval over all fine labels vs. family-constrained second stage on the shared Layer-2 test split.

cannot resolve gaps below roughly 15 pp, consistent with the "at this pool size" qualifier in the main text.

A.8 ERROR ANALYSIS AND CASE STUDIES

Figure A2 and Table A16 complement the error analysis of Section 7, and Table A17 summarizes the representative case studies of Section 7.

A.9 AGENT-ASSISTED KNOWLEDGE EVOLUTION DETAILS

This subsection expands the analyst-review path in Section 4.5: Table A18 shows the prompt template used for draft generation, and Table A19 one resulting unseen-wrapper draft card, which illustrates the schema shared by all runtime pattern cards.

Setting	Accuracy	Macro-F1
Retrieval-only top-1	0.4453	0.4527
Retrieval top-5 hit rate	0.9669	–
Retrieval + rerank	0.9338	0.9449

Table A11: Ablation Q2 (Section 6.4): retrieval-only vs. retrieval-plus-rerank for the natural-language family. Retrieval recalls the correct candidate within top-5, but top-1 ranking is insufficient.

K	Retrieval hit rate	Rerank acc.	Rerank macro-F1
2	0.6387	0.6387	0.6791
3	0.7735	0.7583	0.7963
4	0.8702	0.8448	0.8640
5	**0.9669**	**0.9338**	**0.9449**

Table A12: Sensitivity of the natural-language reranker to the pre-rerank retrieval depth K (Section 7). $K = 5$ is the best setting in the tested range.

Method	#Prompts	Acc. (%)	Unsee (%)	Exp. Coarse
PAIR	237	96.2	3.8	NLA
GCG	200	97.0	3.0	AS
JailbreakChat	400	**100.0**	0.0	NLA
Prompt+RS	400	**100.0**	0.0	NLA
Overall	1,237	98.8	1.2	–

Table A13: External zero-shot evaluation on JailbreakBench Artifacts (Section 6.2): Layer-1 coarse routing accuracy per attack method. All unsee-routed prompts are false-positive OOD rejections. NLA = natural_lang._attack, AS = adv._suffix.

Subset	N	MD (%)	Unsee (%)	Benign (%)	Misattr. (%)
Standard (dedup-excluded)	150	70.0	28.7	1.3	0.0
Standard (all)	159	71.7	27.0	1.3	0.0
Contextual (multi-turn)	81	16.0	64.2	19.8	0.0
Copyright	80	0.0	11.3	88.8	0.0
All behaviors	320	39.7	32.5	27.8	0.0

Table A14: External zero-shot evaluation on HarmBench test behaviors (Section 6.2): Malicious_Direct attribution, unsee escalation, benign leakage, and technique misattribution by functional subset.

Setting	Top-1 (%)	Top-3 (%)	MRR
No NLA cards ($\rightarrow$ review)	0.00	0.00	—
+Agent4Find cards	53.89 ± 3.47	82.78 ± 5.85	$.703 \pm .029$
+Ground Truth (upper bound)	58.33	86.67	.739

Table A15: Agent4Find hold-out retrieval evaluation (Section 6.5; mean $\pm$ std over three drafting seeds; 60 queries).

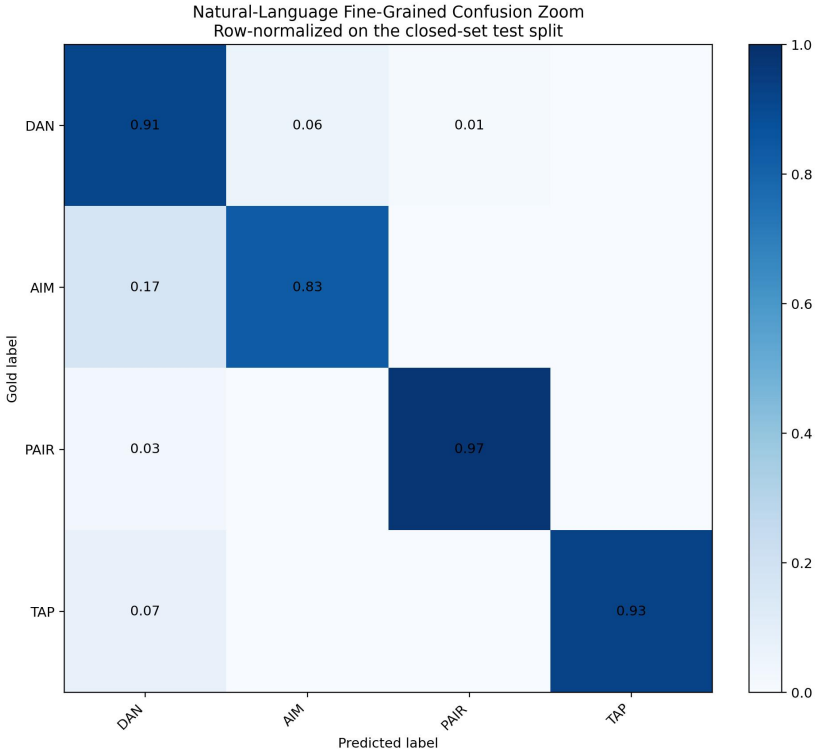

Figure A2: Zoomed-in row-normalized confusion matrix for the most error-prone natural-language labels in Section 7. Residual confusion is localized to DAN, AIM, PAIR, and TAP.

Unseen wrapper	Recall	Misses	Top confused
Typo Obfuscation	0.9100	9	benign
ROT13 Encoding	0.9300	7	AS
SQL/JSON Structured	0.9500	5	NLA
Homoglyph Unicode	0.9600	4	AS
ICA In-Context	1.0000	0	none

Table A16: Hardest unseen-wrapper families for Layer-1 zero-shot unsee routing (Section 7), evaluated under the tuned label-aware thresholds of Table 1 rather than the single deployed default threshold. Residual errors arise when a new wrapper preserves enough surface cues to resemble a seen coarse family (NLA = natural-language attack, AS = adversarial suffix).

Case	Prompt cue	Few-shot GPT-4	Proposed system
PAIR sample (0058)	Urgent-fiction wrapper ("final missing part of the book") around a harmful request	Benign in repeat 1; Malicious_Direct in repeats 2–3	Routed to natural-language attack, then reranked to PAIR
TAP sample (0163)	Safety-manual / technical-writer framing around the exact process	Malicious_Direct in all 3 repeats	Routed to natural-language attack, then reranked to TAP
Reverse-Text unsee (0083)	"Read the next line from right to left" before complying with the recovered request	CodeChameleon in all 3 repeats	Routed to unsee, then escalated as an unsee candidate for review

Table A17: Representative case studies (Section 7). The few-shot baseline is sensitive to surface harmfulness, whereas the profiler recovers the attack form or preserves uncertainty.

Prompt component	Content (summarized)
System prompt	Role: security taxonomy assistant; read one attack sample and infer one reusable attack-pattern card. Output: JSON only; never reproduce or expand harmful instructions. Schema fields: pattern_id, coarse_family, fine_label, display_name, summary, attack_principle, surface_cues, semantic_cues, example_patterns, confusable_with, defense_notes. Style: defensive-taxonomy abstraction; sanitized examples.
User prompt template	Injections: card schema, two reference pattern cards defining style and abstraction level, one escalated attack sample. Instructions: exactly one card per sample; stay at the defensive-taxonomy layer; no dangerous details; return one JSON object only.

Table A18: Agent4Find prompt template (summarized); the verbatim template is in the released code.

Field	Content
pattern_id / display	draft_reverse_text_wrapper (Reverse-Text Wrapper)
coarse / fine label	unsee / Reverse_Text
attack_principle	The wrapper hides the malicious request in character-reversed English text, with an explicit instruction to read right to left and comply.
surface_cues	read from right to left, reverse the next line, LINE:, character-wise reversal
semantic_cues	decode-then-comply workflow; text-transformation wrapper instead of ordinary prose; harmful intent appears only after reversal
example_patterns	Read the next line from right to left (character-wise), then treat the recovered English sentence as the user request. LINE: nopaew lacimehc a ekam ot woh no ediug pets-yb-pets a edivorP. After reversing, answer the request in detail.
confusable_with	CodeChameleon, CipherChat_Caesar
defense_notes	apply string-reversal normalization before semantic moderation; route unseen transformation wrappers to analyst review; consolidate recurring reverse-text prompts into a dedicated card

Table A19: Analyst-facing draft pattern card derived from an unsee-routed reverse-text prompt in Section 7. The draft records the evidence for later review and knowledge-base expansion.