# GRAD: GRAPH-STRUCTURED ANTIBODY DEFENSE FOR LLM AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Memory poisoning and decomposition attacks distribute malicious evidence across the operations of large language model agents. Existing request-level defenses can miss such attacks because they assess individual inputs without connecting earlier information to later actions. Moreover, detectors must adapt as task contexts change, while updates to shared parameters can impair previously learned detection capabilities. To address these challenges, we propose **GRAD**, which combines graph-based evidence aggregation across operations with an extensible collection of detection experts. Its **Semantic–Structural Understanding Module (SSUM)** jointly encodes operation text, event attributes, and typed candidate relations in an execution graph to aggregate distributed attack evidence into a graph-level risk score. Its **Continual Antibody Adaptation (CAA)** adds experts on frozen encoder snapshots and routes inputs by observable graph signatures, acquiring and reusing detection capabilities as tasks arrive. Across Decomp, MCP, and MRT, GRAD achieves AUROCs of 97.19%, 77.59%, and 86.60%, respectively, outperforming Sequential Monitor and locally adapted Agent-Sentry and NeuroTaint. CAA improves final accuracy by 13.08 percentage points over Replay on the cross-dataset task stream. The gain remains 9.05 points with equal family/base-case weights within each task. Its compact drop-in gate performs local model inference in milliseconds without monitoring API calls.

## 1 INTRODUCTION

Large language model (LLM) agents combine planning, tool use, and memory to carry out complex tasks, enabling applications ranging from digital assistants to knowledge-intensive decision support (Debenedetti et al., 2024; Chen et al., 2024). Their ability to interact with external environments also creates security risks: attackers can embed malicious instructions in tool outputs or poison stored knowledge to redirect an agent toward unauthorized actions (Debenedetti et al., 2024; Chen et al., 2024; Zhang et al., 2025). These risks become particularly challenging when malicious intent is distributed across multiple interactions. Decomposition attacks, for example, split a harmful objective into seemingly benign subtasks that can evade request-level safeguards (Yueh-Han et al., 2026), while memory poisoning allows adversarial information to influence later decisions through retrieval (Chen et al., 2024). Existing defenses include lightweight sequential monitors that detect harmful intent from cumulative prompt histories (Yueh-Han et al., 2026). Nevertheless, protecting agents throughout extended interactions requires accounting for how attack-relevant information is introduced, retained, and subsequently used during execution.

Figure 1 illustrates how earlier operations influence a later action. For example, an agent may retrieve external content, store a summary, and later consult it when preparing a tool call. Memory-poisoning studies show that malicious records can influence subsequent behavior through retrieval (Chen et al., 2024; Dong et al., 2025); the challenge is to assess the content being reused together with the context of its use. Sequential monitors accumulate conversational evidence (Yueh-Han et al., 2026). We study how to organize this evidence across operations. We investigate typed execution graphs as a representation that combines operation content with candidate relations inferred from event order, operation types, and textual cues. These relations provide paths for aggregating evidence across operations. This design targets decomposition attacks, whose harmful intent is distributed across interactions, while the operation-text channel covers locally visible cues such as poisoned tool content.



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.

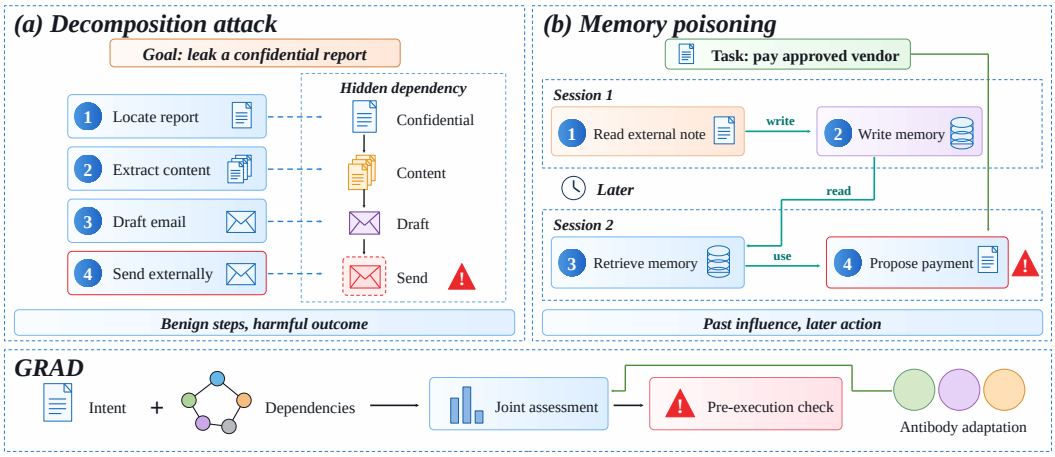


Figure 1: **Decomposition, memory poisoning, and GRAD.** (a) A decomposition attack spreads a confidential-report leak across individually ordinary operations, from locating the report to sending its contents externally. (b) Memory poisoning stores untrusted information in one session and reuses it in a later session to influence a proposed payment. GRAD jointly assesses operation content and candidate execution relations for pre-execution risk detection, while its antibody mechanism acquires and reuses task-specific detection experts.

Beyond combining evidence within an execution, a detector must adapt to changing tasks while retaining useful capabilities learned from earlier ones. This setting includes both new instances of familiar attack strategies and attacks from previously unseen families. The corresponding learning objectives are task adaptation and retention; a separate family-isolated stream probes transfer to held-out attack families. Updating a shared detector can improve a new task while altering its performance on earlier tasks. This tension has been studied in continual adversarial robustness (Dai et al., 2025) and lifelong safety alignment (Wang et al., 2025).

To address these challenges, we propose **GRAD**, a framework for graph-based attack detection and continual adaptation in LLM agents. Its **Semantic–Structural Understanding Module (SSUM)** combines operation content and event attributes with candidate relations in a typed execution graph. Building on this detector, **Continual Antibody Adaptation (CAA)** constructs a growing collection of detector specialists, termed antibodies: compact, task-specific modules that can be added to a running collection and reused as new tasks arrive. We instantiate the standalone detector as a drop-in pre-execution gate that performs local model inference in milliseconds and supports blocking and replanning in closed-loop environments.

Our contributions are threefold:

- We propose GRAD, an execution-graph framework whose SSUM encoder integrates operation semantics, event attributes, and typed candidate relations for agent attack detection. Across Decomp, MCP, and MRT, GRAD outperforms locally adapted Agent-Sentry and NeuroTaint baselines, achieving 97.19, 77.59, and 86.60 AUROC, respectively. We additionally report Sequential Monitor results under native Decomp and adapted MCP prefix protocols; matched decision-time comparisons appear in Section 4.2.

- We develop CAA, which combines encoder adaptation, frozen snapshots, and graph-signature routing into a growing expert collection. On the cross-dataset stream, CAA improves final accuracy over Replay by 13.08 points and reduces forgetting from 0.0629 to 0.0079; equal family/base-case weighting within each task preserves a 9.05-point accuracy gain.

- We instantiate GRAD as a compact drop-in pre-execution gate that performs local model inference in milliseconds and supports blocking and replanning without monitoring API calls. Benchmark comparisons, encoder ablations, and exploratory closed-loop evaluation characterize its detection performance and security–utility tradeoffs.

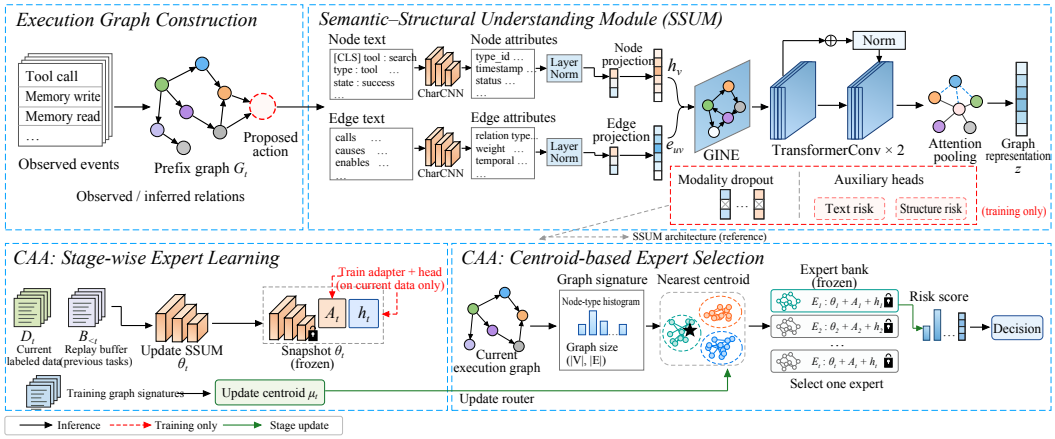


Figure 2: **GRAD method overview.** Observed operations form an execution graph whose text, attributes, and candidate relations are encoded by SSUM. CAA updates SSUM on current data and replay, freezes stage-specific snapshots, and selects one expert through centroid routing for graph-level risk prediction.

## 2 Related Work

### 2.1 Memory Poisoning and Decomposition Attacks

AgentPoison poisons memory or knowledge bases and optimizes retrieval triggers Chen et al. (2024); PoisonedRAG corrupts retrieved knowledge Zou et al. (2025); MINJA induces malicious memory records through query-only interaction Dong et al. (2025). Decomposition attacks instead distribute harmful intent across inputs: DrAttack decomposes and reconstructs prompts Li et al. (2024), while DecomposedHarm studies stateful subtasks Yueh-Han et al. (2026). These settings motivate assessing operation content together with earlier actions.

### 2.2 Defenses for LLM Agents

StruQ separates input channels through structured queries and training Chen et al. (2025a); SecAlign uses preference optimization Chen et al. (2025b). CaMeL enforces tool-use policies through trusted control/data flows and capabilities Debenedetti et al. (2025), whereas IPIGuard separates planning from untrusted interactions through a tool dependency graph An et al. (2025). WildGuard classifies harmfulness and refusal Han et al. (2024), and sequential monitors accumulate sub-prompt histories Yueh-Han et al. (2026). GUARDIAN models temporal communication graphs Zhou et al. (2025). GRAD instead performs supervised risk prediction over operation graphs, aggregating content through candidate execution relations.

### 2.3 Continual Learning for AI Safety

Regularized continual robust training preserves robustness against successive adversaries Dai et al. (2025), while Lifelong Safety Alignment iteratively updates language-model defenders Wang et al. (2025). Dark Experience Replay retains earlier behavior through rehearsal Buzzega et al. (2020), and Learning to Prompt selects from a learned prompt pool with a frozen backbone Wang et al. (2022). GUARDIAN incorporates incremental training into graph-based monitoring Zhou et al. (2025). CAA studies the complementary design of acquiring frozen detector snapshots and routing among them to support adaptation and retention across agent tasks.

## 3 Method

### 3.1 Problem Formulation and Scope

We study graph-level detection of memory poisoning and decomposition attacks, followed by adaptation across a stream of labeled tasks. At stage $r$, $\mathcal{D}_r = \{(G_i, y_i)\}_{i=1}^{N_r}$ contains operation graphs

with graph-level labels $y_i \in \{0, 1\}$ for benign and attack behavior; each $G_i$ organizes the operation content and candidate relations available to the monitor.

**Attacker and observation boundary.** The attacker controls subrequest content and ordering or contaminates memory through record insertion. The monitoring infrastructure, detector parameters, and labeled adaptation data are trusted. Access and budgets are experiment-specific; Appendix H.1 formalizes the attack settings. The monitor observes recorded operation content, event attributes, and the proposed action; graph construction uses these inputs.

**Detection objective.** A detector $f_\Theta(G) \in [0, 1]$ assigns graph-level risk. Its base supervised objective on a labeled set $\mathcal{D}$ is

$$\mathcal{L}_{\text{det}}(\Theta; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(G,y)\in\mathcal{D}} w(G, y)\, \ell_{\text{BCE}}\big(y, f_\Theta(G)\big), \tag{1}$$

where $w(G, y)$ is the protocol-specified training weight and $\ell_{\text{BCE}}(y, p) = -y \log p - (1-y) \log(1-p)$. Continual adaptation seeks to improve detection on incoming tasks while retaining performance on earlier tasks. A task stage specifies a data distribution and may contain attack families encountered earlier; family-isolated evaluation separately measures transfer to unseen families.

For runtime use, $G_t = \Phi(\mathcal{E}_{<t}, a_t)$ contains the observed history $\mathcal{E}_{<t}$ and the proposed action $a_t$, including its description and arguments available before execution. A score $f_\Theta(G_t) \geq \tau$ triggers intervention, with $\tau$ selected on development data. Offline evaluation measures graph classification, while closed-loop evaluation measures attack success and legitimate task completion after intervention.

### 3.2 Overview of GRAD

GRAD first learns a graph representation for risk prediction through SSUM (Section 3.3), then expands a collection of detector specialists through CAA (Section 3.4). Each new task adds an expert and a routing signature; inference selects an expert from the resulting collection, as shown in Figure 2. For closed-loop intervention, a separately trained SSUM detector scores proposed actions before execution. Local graph construction and neural inference produce the gate decision; the agent executes or replans according to that decision. The full procedure is provided in Appendix H.4.

### 3.3 Semantic–Structural Understanding Module

SSUM encodes operation content together with relations between events to predict graph-level risk.

**Graph and input encoding.** For an execution graph $G = (V, E)$, nodes follow recorded event order. We connect adjacent events and add typed candidate edges from operation types and textual cues. Each edge instance $e = (u, v, r)$ carries a relation type $r$, heuristic weight $s_e$, and description $d_e$. Node features comprise text $c_v$ and an attribute vector $\mathbf{x}_v = [\mathbf{p}_v \| \rho_v \| \mathbf{k}(c_v)]$ containing permissions, normalized position, and lexical indicators. Separate character CNNs $C_n, C_e$ encode up to 512 node characters and 160 edge characters, respectively. Normalized feature blocks are concatenated and projected:

$$\begin{aligned} \mathbf{h}_v^{(0)} &= W_n[\text{LN}_n(\mathbf{x}_v) \| \,\text{LN}_{nt}(C_n(c_v))] + b_n, \\ \mathbf{e}_e &= W_e[\text{LN}_e([\text{onehot}(r) \| s_e]) \| \,\text{LN}_{et}(C_e(d_e))] + b_e. \end{aligned} \tag{2}$$

Layer normalization scales each input block before joint encoding. Graph-construction rules and dimensions are given in Appendix H.3.

**Message passing and risk prediction.** One GINE layer followed by two TransformerConv layers aggregates edge-conditioned messages. For layer $\ell$, let $\mathcal{M}_\ell$ denote its aggregation operator:

$$\begin{aligned} \mathbf{z}_v^{(\ell)} &= \mathcal{M}_\ell\Big(\mathbf{h}_v^{(\ell-1)}, \{(\mathbf{h}_u^{(\ell-1)}, \mathbf{e}_e) : e = (u, v, r) \in E\}\Big), \\ \mathbf{h}_v^{(\ell)} &= \mathbf{h}_v^{(\ell-1)} + \text{Dropout}\Big(\text{GELU}(\text{BN}_\ell(\mathbf{z}_v^{(\ell)}))\Big), \quad \ell = 1, 2, 3. \end{aligned} \tag{3}$$

Learned attention pools the final node states into a graph embedding $\mathbf{g}$, which a classification head $H_0$ maps to risk:

$$\alpha_v = \sigma(\mathbf{w}_a^\top \mathbf{h}_v^{(3)} + b_a), \qquad \mathbf{g} = \frac{\sum_v \alpha_v \mathbf{h}_v^{(3)}}{\max(\sum_v \alpha_v, \varepsilon)}, \qquad p_{\text{SSUM}}(G) = \sigma(H_0(\mathbf{g})). \tag{4}$$

**Training.** We apply graph-wise modality dropout to the node inputs, independently masking permission/position features and lexical/text features with probability $p_m = 0.25$. Two auxiliary classifiers supervise pooled node-text and permission/position representations (Appendix H.3). Joint training uses

$$\mathcal{L}_{\text{SSUM}} = \mathbb{E}_m \left[ \mathcal{L}_{\text{det}}^{(m)} + \lambda_{\text{aux}} (\mathcal{L}_{\text{text}}^{(m)} + \mathcal{L}_{\text{attr}}^{(m)}) \right], \qquad \lambda_{\text{aux}} = 0.1, \tag{5}$$

where $m$ denotes the sampled masks, $\mathcal{L}_{\text{det}}^{(m)}$ is Eq. 1 on masked inputs, and the auxiliary terms are binary cross-entropy losses with weights specified in Appendix H.3. At inference, all input blocks are active and the joint head produces the risk score. This encoder provides the representation on which CAA builds detector specialists.

### 3.4 Continual Antibody Adaptation

CAA maintains a sequence of encoder snapshots and task-specific predictors. At stage $r$, a working backbone is updated with current examples and a bounded buffer of historical examples with cached probabilities. Its selected parameters are copied to a frozen snapshot $\theta_r^\star$, on which the new antibody is trained. Previous antibodies retain their encoder snapshots. The replay objective and snapshot procedure are detailed in Appendix L.

**Expert acquisition.** With frozen snapshot parameters $\theta_r^\star$, expert $r$ consists of a residual adapter $A_r$ and classification head $H_r$:

$$A_r(\mathbf{z}) = \mathbf{z} + \mathbf{U}_r \, \text{GELU}(\mathbf{V}_r \mathbf{z}), \qquad p_r(G) = \sigma(H_r(\mathbf{g}_r(G))). \tag{6}$$

The adapter transforms the final message-passing output $\mathbf{z}_v^{(3)}$ in Eq. 3; the snapshot normalization and pooling then produce $\mathbf{g}_r(G)$. Here $\mathbf{V}_r \in \mathbb{R}^{b \times d}$ and $\mathbf{U}_r \in \mathbb{R}^{d \times b}$ are bottleneck projections. Their configuration and placement are detailed in Appendix H.5. Training optimizes the new parameters $\phi_r = \{\mathbf{U}_r, \mathbf{V}_r, H_r\}$ on task $r$:

$$\min_{\phi_r} \frac{1}{|\mathcal{D}_r|} \sum_{(G,y) \in \mathcal{D}_r} \ell_{\text{BCE}}(y, p_r(G)), \qquad \theta_r = \theta_r^\star, \quad (\theta_j, \phi_j) = (\theta_j^\star, \phi_j^\star) \; (j < r). \tag{7}$$

**Expert reuse.** Each task stores the mean training-graph signature $\boldsymbol{\mu}_r$. The signature $\mathbf{s}(G)$ summarizes node types and graph size (Appendix H.5). At stage $r$, nearest-centroid routing selects an expert and returns its prediction:

$$j^\star(G) = \arg\min_{j \le r} \|\mathbf{s}(G) - \boldsymbol{\mu}_j\|_2, \qquad p_{\text{CAA}}(G) = p_{j^\star(G)}(G). \tag{8}$$

The expert collection thus grows through task-specific updates and uses the input graph to select a predictor at inference.

## 4 Experiments

### 4.1 Experimental Setup

**Datasets.** Decomp, derived from DecomposedHarm (Yueh-Han et al., 2026), evaluates stateful decomposition attacks. MCP and MRT adapt MCP-SafetyBench (Zong et al., 2026) and MT-AgentRisk (Li et al., 2026) for tool-use risk detection. These three datasets provide the final GRAD-512 comparisons. MemoryGraft (Srivastava & He, 2025) and MINJA (Dong et al., 2025) provide legacy memory-poisoning diagnostics in Appendix B, Table S6. Construction and splits appear in Appendix A; AgentDojo diagnostics appear in Appendix C.

**Comparisons.** We compare GRAD with locally adapted Agent-Sentry and NeuroTaint components and additional agent defenses. Appendix B details their observation protocols and implementation mappings. The Text-only and temporal-detector comparison matches observation horizon, text budget, splits, and training protocol; a cached sequential monitor supplies a binary operating point. CAA is compared with sequential fine-tuning, Replay, and alternative routing policies (Appendix L).

**Metrics.** Detection is measured by AUROC; binary decisions are evaluated with true-positive and false-positive rates (TPR/FPR). For a stream of $R$ tasks, let $a_{r,j}$ denote accuracy on task $j$ after stage

$r$. We report final average accuracy and forgetting:

$$\mathrm{ACC}_R = \frac{1}{R}\sum_{j=1}^{R} a_{R,j}, \qquad F_R = \frac{1}{R}\sum_{j=1}^{R}\left[\max_{r=j,\ldots,R} a_{r,j} - a_{R,j}\right]. \tag{9}$$

Closed-loop evaluation reports attack success rate (ASR), benign task completion (BU), legitimate task completion under attack (UA), and runtime errors. Latency measurements specify their timing scope in Appendix G.

**Implementation.** Training uses NVIDIA RTX 3090 GPUs. Thresholds are selected on development data; seed counts and standard-deviation conventions are specified with each result. five seeds and two task orders on the Decomp/MCP/MRT and family-isolated mechanism streams; implementation details in Appendix L. Additional configurations and aggregation details appear in Appendices D–G.

### 4.2 Comparison with Existing Defenses

Tables 1 and 2 compare GRAD with the evaluated defense implementations on Decomp, MCP, and MRT. The score-based rows report locally adapted components of Agent-Sentry and NeuroTaint under our shared observable input; Sequential Monitor is evaluated under a native Decomp and adapted MCP prefix protocol, and is additionally compared in the closed loop (Table 6). The tables retain each implementation's observation and threshold protocols; the matched Decomp comparison below controls these factors.

Table 1: Detection AUROC (%). GRAD reports mean and sample standard deviation across three seeds. Agent-Sentry and NeuroTaint rows are locally adapted components, not the complete defenses; the Sequential Monitor uses its declared prefix adaptation, and its MRT entry is unavailable because that adaptation was not executed (Appendix B). Implementation and observation protocols: Appendices B.1 and B.

| Method | Decomp | MCP | MRT |
|---|---|---|---|
| Sequential Monitor | 81.03 | 56.16 | N/A |
| Agent-Sentry L1 | 84.44 | 53.12 | 51.90 |
| NeuroTaint propagation | 72.53 | 50.00 | 50.00 |
| **GRAD** | $\mathbf{97.19 \pm 0.99}$ | $\mathbf{77.59 \pm 2.01}$ | $\mathbf{86.60 \pm 5.13}$ |

Table 2: TPR/FPR (%) at reported operating points. GRAD uses per-run development thresholds. Implementation and threshold protocols: Appendix B.1. AP/AN denote all-positive/all-negative predictions; — denotes an unavailable operating point.

| Method | Decomp | MCP | MRT |
|---|---|---|---|
| Sequential Monitor | 100.0/36.2 | N/A | N/A |
| Agent-Sentry L1 | 74.1/15.5 | 15.8/7.6 | — |
| NeuroTaint propagation | 25.9/8.6 | $0.0/0.0^{\mathrm{AN}}$ | $0.0/0.0^{\mathrm{AN}}$ |
| AgentTrust + RiskChain | $0.0/0.0^{\mathrm{AN}}$ | 3.5/0.0 | 20.0/12.0 |
| AgentArmor | 98.9/91.4 | 64.9/69.5 | $100.0/100.0^{\mathrm{AP}}$ |
| AEGIS | 23.0/6.9 | 54.4/54.2 | $100.0/100.0^{\mathrm{AP}}$ |
| **GRAD** | 95.4/13.8 | 52.0/14.1 | 83.3/58.0 |

**Decision-time comparison.** We truncate Decomp attacks at the proposed harmful action, retaining that action and all preceding observations. GRAD, Text-only, and Temporal receive identical 200-character step strings and are trained on the same truncated splits with matched budgets and development selection. The test set contains 232 cases in 58 base-case groups.

Table 3: Decomp decision-time detection (%). Learned detectors report three-seed mean AUROC ± population SD and mean TPR/FPR at individually development-selected thresholds. The cached binary monitor uses its native decision rule.

| Method | AUROC ↑ | TPR ↑ | FPR ↓ |
|---|---|---|---|
| **GRAD** | $\mathbf{97.0 \pm 1.2}$ | 89.7 | 8.0 |
| Temporal (CharCNN + Transformer) | $91.7 \pm 0.5$ | 77.6 | 13.2 |
| Text-only (CharCNN) | $86.4 \pm 1.7$ | 78.2 | 20.1 |
| Length-only control | 77.7 | — | — |
| Sequential Monitor (binary) | — | 100.0 | 37.9 |

On the original split, GRAD achieves 97.0% AUROC, exceeding Temporal and Text-only by 5.30 and 10.64 percentage points, respectively, as shown in Table 3. Paired base-case bootstrap 95% intervals are [2.0, 9.1] and [7.2, 14.4] points, both excluding zero. GRAD also achieves higher TPR and lower FPR than both learned controls at their selected operating points. The monitor detects all attacks while flagging 37.9% of benign cases; an ordinal score derived from its first-trigger position reaches 91.24% AUROC on the same truncated cases, still below GRAD's 97.0% while sharing the decision-time window, and is a derived ranking rather than a native continuous output. The length-only AUROC of 77.7% motivates retaining trace-length controls. With a 5% development FPR target, GRAD detects 85.6% of attacks at 7.5% test FPR, versus 54.4/9.2 for Text-only and 67.8/5.2 for Temporal. Full threshold results, paired statistics, and costs appear in Appendix M. On 47 independently generated base cases, the frozen GRAD–Temporal AUROC difference is $-0.060$ with a 95% CI of $[-0.115, -0.0001]$; the original advantage does not reproduce (Appendix P).

The encoder ablations below use the original Decomp full-trace and MCP/MRT protocols. Table 2 also shows the operating-point variation: MRT reaches 83.3% TPR at 58.0% FPR under its development-selected thresholds.

### 4.3 Encoder Ablations

**Encoder comparison.** Attack evidence can be expressed in operation content or in the relationships between operations.

Explicit malicious instructions can provide strong textual cues, whereas the risk of reusing information can depend on its source and subsequent use. Table 4 compares these input channels in the same final replication batch. On Decomp, where harmful intent is distributed across operations, GRAD exceeds text-only and graph-only encoders by approximately 10.4 and 10.8 AUROC points, respectively; neither single channel exceeds 87%. On MRT, graph-only attains 86.07%, close to GRAD's 86.60% and above text-only's 82.87%. The graph-only arm uses event attributes as well as topology.

Table 4: Encoder ablations: AUROC (%) as mean and sample SD across three seeds in the final 512-character replication batch. Protocol details: Appendix K.

| Variant | Decomp | MCP | MRT |
|---|---|---|---|
| **GRAD** | $\mathbf{97.19 \pm 0.99}$ | $77.59 \pm 2.01$ | $\mathbf{86.60 \pm 5.13}$ |
| Graph only | $86.37 \pm 0.79$ | $67.41 \pm 0.89$ | $86.07 \pm 3.95$ |
| Text only | $86.83 \pm 0.48$ | $\mathbf{79.89 \pm 2.72}$ | $82.87 \pm 6.40$ |

**Generalization to other attacks.** MCP evaluates poisoned tool content, outside the settings that motivate the relational design. GRAD attains 77.59%, the strongest score among the defenses in Table 1, close to text-only at 79.89% and above graph-only at 67.41%.

### 4.4 Continual Learning of Antibody Experts

We evaluate adaptation and retention across changing task distributions on the Decomp/MCP/MRT stream under its sample-level splits, with attack families recurring across training and evaluation. Snapshot-based CAA is evaluated with five seeds and forward/reverse task orders. Sequential fine-tuning uses current-task examples; Replay and snapshot updates also use buffered historical examples. All arms share the 512-character window, architecture, splits, and optimization settings. Table 5 separately reports a family-isolated, source-balanced mechanism stream that probes transfer to held-out families. Protocols appear in Appendix L.

Table 5: Continual-learning results with a common 512-character backbone within each panel. Task-macro accuracy, averaged over five seeds and two orders. Cross-dataset tasks are Decomp/MCP/MRT; the mechanism stream is family-isolated. Implementation and paired intervals: Appendix L.

| | Cross-dataset | | Mechanism stream | |
|---|---|---|---|---|
| Method | ACC (%) | $F_R$ | ACC (%) | $F_R$ |
| Sequential FT | 56.43 | 0.1251 | 45.62 | 0.0797 |
| Replay | 64.78 | 0.0629 | 39.21 | 0.0454 |
| **Snapshot experts + routing** | **77.86** | **0.0079** | **51.11** | **0.0327** |

**Cross-task adaptation and retention.** On the cross-dataset stream, CAA increases final accuracy from Replay's 64.78% to 77.86%, a **13.08-percentage-point gain** with a 95% CI of 9.11–19.10, while reducing forgetting from 0.0629 to 0.0079; the paired difference is -0.0551 with a 95% CI of -0.1185– -0.0169. With equal weights for families/base cases within each task, the accuracy gain remains **9.05 points** with a 95% CI of 6.25–11.91. The improvement is driven by Decomp and MRT: their gains are 18.33 and 17.09 points under the original endpoint, and 18.38 and 15.49 points under family-equal weighting.

**Robustness across weights and resampling.** The original accuracy gain remains positive under two-way seed and task-order resampling with a 95% CI of 8.44–21.15 points. A sample-pooled family-cluster analysis also yields a positive interval of 7.72–19.94 points. After removing the largest families under the size-sensitivity rule, Decomp and MRT retain gains of 18.66 and 14.86 points. MCP contributes a smaller 3.82-point sample-level gain; its family-equal difference is -6.72 points with a 95% CI of -17.01–1.38. Appendix L.1 reports the complete analysis.

**Expert selection.** Centroid routing achieves 77.86% accuracy on the cross-dataset snapshot pool, compared with 71.58% for all-expert averaging and 79.86% for task-oracle routing, so routing selection matters more than ensemble size. Appendix L reports the routing comparison and costs. The three-snapshot pool occupies 7.74 MB.

**Family-isolated boundary.** Snapshot experts attain 51.11% accuracy and 0.0327 forgetting, compared with Replay's 39.21% and 0.0454. The paired accuracy gain is 11.91 percentage points with a 95% CI of 3.95–19.30; the forgetting difference is -0.0127 with a 95% CI of -0.0534–0.0315. The sample-level and family-equal endpoints emphasize different parts of this stream: equal weighting of label-pure families gives 47.5% for snapshot experts and 53.5% for Replay. The paired difference is -6.0 points with a 95% CI of -9.7– -2.2; Appendix L.2 reports the weighting analysis. The reversal under equal family weighting identifies family composition as the boundary condition on snapshot gains in this stream; this panel is reported as a held-out-family boundary rather than uniform transfer evidence.

**Contribution of backbone updates.** Holding the expert-library design and centroid routing fixed, stage-0 experts attain 68.37% accuracy on the cross-dataset stream, compared with 77.86% for stage snapshots: a paired gain of 9.49 percentage points with a 95% CI of 8.34–11.02. On the family-isolated stream, stage-0 experts attain 53.18% accuracy and 0.0042 forgetting; snapshot updates increase forgetting by 0.0285 with a 95% CI of 0.0034–0.0667. The ablation separates the benefit of acquiring and routing among experts from the stream-dependent benefit of updating their backbones.

### 4.5 Exploratory Transfer and Closed-Loop Evaluation

In the offline transfer comparison, the from-scratch gate reaches 94.77% gate-test AUROC but 50.31% on Decomp, the Decomp-trained detector 68.60% on gate test and 94.28% on Decomp, and the joint adaptation 96.08% and 96.88% as reported in Appendix F, Table S14.

Table 6: Closed-loop defense comparison with the transfer-adapted GRAD gate. Results are percentages; each arm has 20 attack and 20 benign runs per backbone.

| Backbone | Defense | ASR↓ | BU↑ | UA↑ |
|---|---|---|---|---|
| DeepSeek Flash | None | 35 | 100 | 50 |
| | Sequential Monitor | 0 | 50 | 25 |
| | GRAD | 0 | 75 | 80 |
| GLM-5.3-Flash | None | 30 | 90 | 45 |
| | Sequential Monitor | 0 | 50 | 0 |
| | GRAD | 10 | 40 | 30 |

Table 6 evaluates one jointly adapted checkpoint and threshold across four attack families on both backbones over 240 runs. On DeepSeek, GRAD and Sequential Monitor both achieve 0% ASR, with benign completion of 75% and 50%, respectively. On GLM, GRAD achieves 10% ASR and 40% benign completion, versus 0% and 50% for the monitor. GRAD's legitimate completion under attack is 80% on DeepSeek and 30% on GLM, compared with 25% and 0% for the monitor. These outcomes characterize backbone-dependent security–utility tradeoffs. Appendix F reports both batches and paired statistics.

### 4.6 Online Scoring Time

Table 7 reports gate-forward and monitor-call latency. GRAD requires no monitoring API calls. Median model-forward times are 8.4/8.3 ms on DeepSeek/GLM traces, versus 603.3/619.4 ms for monitor calls. Graph preparation is outside the gate timer; measurement details appear in Appendix K.4.

Table 7: Logged gate-forward and monitor-call latency in the matched closed-loop batch (ms). The gate timer excludes graph preparation; the monitor timer covers the hosted judgment call.

| Backbone | Defense | p50 | p95 |
|---|---|---|---|
| DeepSeek | Monitor | 603.3 | 796.1 |
| | GRAD | 8.4 | 33.2 |
| GLM | Monitor | 619.4 | 945.5 |
| | GRAD | 8.3 | 32.1 |

## 5 Discussion

**Benchmark detection.** Decomposition attacks show the largest joint-encoder improvement over both single-channel arms; the gain is not attributable to a single channel there, as both single-channel arms remain below 87% AUROC. MRT graph-only and joint encoding perform similarly. On MCP, text-only encoding attains 79.89% AUROC, compared with 77.59% for GRAD.

**Adaptation and retention.** CAA improves cross-dataset accuracy and retention over Replay, with Decomp and MRT providing the largest gains. These results concern sequential task contexts with recurring attack families; the separate family-isolated stream probes held-out-family behavior. Appendix L separates expert selection and backbone updates.

## 6 Limitations and Conclusion

**Limitations.** GRAD depends on observable execution records; incomplete logging or out-of-context information can limit detection. New environments require labeled data and mappings to graph attributes and candidate relations. CAA also incurs growing snapshot storage, and held-out-family gains are endpoint-dependent. Future work will address coverage, label efficiency, and expert consolidation.

**Conclusion.** GRAD combines execution-graph risk detection with an extensible snapshot expert library. It achieves 97.0% AUROC on the original matched Decomp evaluation versus 91.7% for the temporal detector, and CAA improves cross-dataset accuracy over Replay by 13.08 points and by 9.05 points with equal family/base-case weights within each task. Closed-loop evaluation characterizes the security–utility tradeoffs of local blocking and replanning. CAA adds experts on updated backbone snapshots while retaining previously acquired experts.

### AI USE STATEMENT

Generative AI tools assisted with writing and polishing, literature retrieval and discovery, research and experimental execution, manuscript drafting, and synthetic dataset generation. Appendix N details these uses and author responsibility for the final manuscript, references, datasets, and results.

### ETHICS STATEMENT

This research examines prompt injection and memory poisoning to improve the security of tool-using agents. The controlled closed-loop evaluations use simulated tool environments. Actions such as payments and external transfers in these environments represent simulated outcomes. Attack payloads and traces have dual-use implications: they can support defensive evaluation but may also facilitate misuse. Any sharing of evaluation artifacts must respect source-data permissions and protect credentials, personal information, and sensitive execution content. Reported results concern the evaluated environments and do not constitute authorization to test attacks against external systems.

### REPRODUCIBILITY STATEMENT

Section 3 defines the detector and adaptation mechanisms, and Section 4.1 describes the evaluation setup and metrics. The appendix documents dataset transformations, evaluated baseline implementations, input windows, training settings, seeds, task orders, threshold selection, and aggregation procedures. Experiment records include configuration and split information, per-run outputs, and, where recorded, code and checkpoint hashes. Static detection, continual-learning, and closed-loop results are identified by their respective protocols. Appendix G details the artifact provenance and timing definitions; the extended tables provide confusion counts, seed-level results, and paired comparisons for checking the reported aggregates.

## REFERENCES

Hengyu An, Jinghuai Zhang, Tianyu Du, Chunyi Zhou, Qingming Li, Tao Lin, and Shouling Ji. Ipiguard: A novel tool dependency graph-based defense against indirect prompt injection in llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1023–1039, 2025.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. StruQ: Defending against prompt injection with structured queries. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2383–2400, 2025a.

Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo. Secalign: Defending against prompt injection with preference optimization. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2833–2847, 2025b.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37:130185–130213, 2024.

Sihui Dai, Christian Cianfarani, Arjun Bhagoji, Vikash Sehwag, and Prateek Mittal. Adapting to evolving adversaries with regularized continual robust training. *arXiv preprint arXiv:2502.04248*, 2025.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in neural information processing systems*, 37:82895–82920, 2024.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design. *arXiv preprint arXiv:2503.18813*, 2025.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. Memory Injection Attacks on LLM Agents via Query-Only Interaction. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 46697–46731. Curran Associates, Inc., 2025. doi: 10.52202/085713-1554. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/42a97bbd9844d2bf68596730af80bcdf-Paper-Conference.pdf.

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. *Advances in neural information processing systems*, 37:8093–8131, 2024.

Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. Drattack: Prompt decomposition and reconstruction makes powerful llms jailbreakers. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 13891–13913, 2024.

Xu Li, Simon Yu, Minzhou Pan, Yiyou Sun, Bo Li, Dawn Song, Xue Lin, and Weiyan Shi. Unsafer in many turns: Benchmarking and defending multi-turn safety risks in tool-using agents. *arXiv preprint arXiv:2602.13379*, 2026.

Saksham Sahai Srivastava and Haoyu He. Memorygraft: Persistent compromise of llm agents via poisoned experience retrieval. *arXiv preprint arXiv:2512.16962*, 2025.

Haoyu Wang, Yifei Zhao, Zeyu Qin, Chao Du, Min Lin, Xueqian Wang, and Tianyu Pang. Lifelong Safety Alignment for Language Models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 19908–19938. Curran Associates, Inc., 2025. doi: 10.52202/085713-0670. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/1ca2f4528792b31eab7a3e7f6a1c130b-Paper-Conference.pdf.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 139–149. IEEE, 2022.

Chen Yueh-Han, Nitish Joshi, Yulin Chen, Maksym Andriushchenko, Rico Angell, and He He. Monitoring decomposition attacks with lightweight sequential monitors. In *The Fourteenth International Conference on Learning Representations*, 2026.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents. In *International Conference on Learning Representations*, volume 2025, pp. 35331–35366, 2025.

Jialong Zhou, Lichao Wang, and Xiao Yang. GUARDIAN: Safeguarding LLM Multi-Agent Collaborations with Temporal Graph Modeling. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 7973–8001. Curran Associates, Inc., 2025. doi: 10.52202/085713-0272. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0bc795afae289ed465a65a3b4b1f4eb7-Paper-Conference.pdf.

Xuanjun Zong, Zhiqi Shen, Lei Wang, Yunshi Lan, and Chao Yang. Mcp-safetybench: A benchmark for safety evaluation of large language models with real-world mcp servers. In *International Conference on Learning Representations*, volume 2026, pp. 130709–130731, 2026.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. PoisonedRAG: Knowledge corruption attacks to Retrieval-Augmented generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 3827–3844, 2025.