Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 3, 2026

AGENTIC AI FOR CONTINUOUS CONCEPT-DRIFT MONITORING AND AUTONOMOUS RETRAINING IN MACHINE-LEARNING-BASED INTRUSION DETECTION SYSTEMS: A PROPOSED FRAMEWORK FOR EXPLAINABLE, LEAKAGE-SAFE MODEL LIFECYCLE MANAGEMENT IN NETWORK SECURITY

[1]Muqaddas Abbas, [2]Muhammad Fasihuddin, [*3]Muhammad Bilal Akbar
[1]University of Wolverhampton
[1]Muqadasbbas476@gmail.com, [2]Fasihudin78@gmail.com, [*3]Bilalakbar1094@gmail.com

DOI: https://doi.org/10.5281/zenodo.22200496

Article History
Received on: 21 Feb, 2026
Accepted on: 15 March, 2026
Published on 17 March, 2026

Corresponding Author
Muhammad Bilal Akbar

Abstract

IDSs based on machine learning usually perform very well on the benchmark against which they were trained, but then start to get worse when deployed in production. Attacker behavior evolves, legitimate traffic patterns shift with new applications and user habits, and the statistical assumptions baked into a classifier at training time slowly stop holding — a problem known as concept drift. The usual fixes are blunt: retrain on a fixed schedule whether or not anything has actually changed, or bolt on a statistical drift detector that can say something shifted but not what shifted, why, or whether it is safe to retrain on the data that triggered the alarm. This paper proposes an agentic AI framework that closes that gap. A small team of cooperating agents continuously watches statistical drift signals from a deployed intrusion detection model, reasons over that evidence using an LLM to distinguish ordinary traffic evolution from a possible adversarial probing campaign, curates a retraining batch under the same leakage-safe preprocessing discipline the model was originally trained with, retrains only when justified, and validates the candidate model against held-out and canary traffic before it is ever promoted to production — all of it logged into an auditable trail rather than acting as an unaccountable black box. It explains the architecture, provides a description of how each agent's choice would be graded, and explicitly spells out failures this design will be protected from, such as drift created by an attacker specifically for the purpose of poisoning the retraining loop.

1. Introduction

There is a known disparity between accuracy and reliability in intrusion detection research. A classifier that is trained and cross validated with an idosyncratic benchmark like CIC-IDS2018 or NSL-KDD may get very close to perfect scores before getting caught on the tail end of a deployment due to the dynamic nature of networked systems. New applications emerge, encryption algorithms evolve, seasonal usage patterns shift, and, finally, attackers try their luck at pushing the limits of what a deployed model will and will not detect. This is all concept drift, i.e. the relationship between the input features and the correct label changes but the model was not trained for it. Empirical studies of this problem do not mince words over the magnitude of the effect: one recent work on network intrusion data measured the magnitude of the drop in F1 score of a classifier trained on a 1-month set of traffic data and tested on the next 1-month set, and reported a drop from 0.96 to 0.687, just from the temporal shift [4].

The typical answers to the problem are adequate but incomplete. When there is no change, periodic retraining is a waste of computation, and when there is a sudden change, it reacts too slowly. They are efficient and well understood, but answer the wrong question — has the distribution changed? — and do not answer the questions an operator might seek answers for: is it benign; is it a signature of an attacker deliberately trying to change the inputs to the model to make it worse; and is it safe to retrain on the data used that caused? the alarm without reintroducing the kind of preprocessing errors that have been shown to inflate reported IDS accuracy by double digits. Active-learning approaches to drift adaptation reduce labeling cost but still typically need a human-defined trigger to run, and rarely produce any explanation for why a particular batch of traffic was selected for retraining [8].

This paper proposes closing that reasoning gap with an agentic AI layer sitting on top of conventional statistical drift detection. Rather than treating drift detection and retraining as two disconnected scripts, the framework proposed here uses a small set of cooperating agents — one that watches the statistics, one that reasons over what the statistics mean, one that curates a safe retraining batch, one that executes the retrain, and one that validates the result before it ever reaches production — so that every retraining decision comes with a human-readable rationale and an auditable trail, not just a triggered flag.

2. Background

2.1 Concept Drift in Network Intrusion Detection

The usual forms of concept drift in this context are as follows. The normal drift of legitimate and malicious traffic over time: a new version of software, a new normal usage, a bit more new attack tooling based on the same techniques. Adversarial or adversarial-induced drift is another type, in that it is a deliberate effort by an adversary to alter the input distribution in such a way that it hampers detection, sometimes referred to as a mimicry or evasion campaign. Bayram, Ahmed, and Kassler's work on performance aware drift detectors considers this a continuum ranging from slow drift to sudden drift to periodic drift, requiring different response strategies for each type of drift [13]. As mentioned above, the field is adversarial, with malware and network intrusion research repeatedly demonstrating this fact: Consider the work of Chen et al. for continuous learning for Android malware detection, which is built expressly around the assumption that the data distribution is constantly changing, and the defender must constantly update a model rather than merely deploy it once [12].

2.2 From Static Drift Detectors to Agentic Reasoning

For the first step in this problem — to cheaply and reliably mark the change — statistical detectors are the right choice. Recent work on drift-adaptable intrusion detection illustrates this well: ExpIDS combines an autoencoder-based anomaly signal with decision-tree explanations so that operators get some visibility into why traffic is flagged, and explicitly targets adaptability to distributional drift as a design goal rather than an afterthought [2]. Representation-based methods, like ReCDA, extend this by learning to represent the data with self-supervised representation enhancement, to enable the model to adapt to drifted traffic, without the need of large amounts of newly labeled data [1]. What none of these systems do is reason, in the way a human analyst would, about whether a

detected shift looks like ordinary evolution or looks like an attacker probing for blind spots — that judgment call has historically been left to a human.

Agentic AI is a natural fit for that judgment layer because the underlying problem is not a single classification task but a multi-step investigation: gather evidence, weigh context, decide on an action, and justify the decision. Outside of intrusion detection specifically, this pattern is already appearing in adjacent infrastructure domains. RIVA uses LLM agents that iteratively query cloud telemetry and configuration APIs to verify whether a live system still matches its intended state, treating drift detection as an evidence-gathering task rather than a single statistical test [7]. In MLOps more broadly, a recently proposed multi-agent framework for automated data engineering and AutoML deployment includes drift-aware lifecycle optimization as one of its explicit responsibilities, coordinating data engineering, model selection, and deployment agents around a shared drift signal [10]. This paper adapts that general pattern to the specific, security-sensitive case of network intrusion detection, where the cost of a wrong retraining decision — either missing a real attack or retraining on poisoned data — is considerably higher than in a typical business analytics pipeline.

3. Proposed Framework

3.1 Architecture Overview

The framework is organized as a pipeline of six cooperating agents, each with a narrow responsibility, coordinated by a lightweight orchestrator that passes structured state between them. Having a narrow role for each agent is a deliberate design: a narrow agent that just monitors the statistics is easy to trust, versus one agent asked to monitor the statistics, judge the intent, retrain the model and deploy it all at once is much harder to audit and much easier to fool.

Agent	Role	Key Behavior
Monitor Agent	Continuous statistical surveillance	Runs lightweight drift statistics (ADWIN on error rate, PSI on feature distributions) over sliding windows of live traffic and model confidence scores
Diagnosis Agent	LLM-based reasoning over drift evidence	Reviews the statistical signal alongside contextual metadata (traffic source, time window, affected classes) and produces a structured verdict: benign drift, adversarial drift, or false alarm
Curation Agent	Leakage-safe data preparation	Assembles a retraining batch from recent labeled or actively-labeled traffic, reapplying the same fit-on-train-only encoding discipline used in the original pipeline so retraining cannot reintroduce leakage
Retraining Agent	Model update execution	Retrains or fine-tunes the classifier on the curated batch, using incremental or full retraining depending on the diagnosis agent's severity assessment
Validation Agent	Safe rollout gate	Evaluates the candidate model on a held-out recent window and canary traffic; blocks promotion if performance regresses on any existing attack class
Audit Agent	Traceability	Logs every decision, the evidence behind it, and the final action into an append-only record for human review

3.2 Drift Detection Layer

The Monitor Agent does not utilize an LLM — there is no need to add the cost of model calls to a job where classical statistics can do an adequate job. It processes the classifiers' prediction probabilities, the model's rolling error rate (ADWIN or DDM), the population stability index over all the feature distributions, and a simple confidence-drop signal

over the rolling error rate of the model. This is reminiscent of drift adaptive systems of intrusion detection that always couple a statistical trigger with an adaptation mechanism that occurs downstream [2][6]. The results of this stage are not a retraining command but rather an evidence package: drift direction, amount of drift, length of the drift period, and what classes of traffic were affected.

3.3 LLM-Based Diagnosis: Benign Drift vs. Adversarial Drift vs. False Alarm

The main contribution of the framework is the evidence packet, which is handed to the Diagnosis Agent. Instead of having a human analyst interpret a raw drift alert, an LLM reasons over the structured evidence, including whether there is a shift, whether this shift is correlated to a known source subnet or time window, if affected traffic matches known patterns of evasion, and how the drift compares to the system's drift history, and returns a structured verdict with a confidence score and a written rationale, akin to the LLM-as-judge paradigm recently employed to alert triage in neighbouring agentic security systems. It's this step that a statistical-only drift detector cannot do alone: a PSI score or an ADWIN trigger alerts you to the fact that a distribution moved, but not why it moved or if it's safe to learn from. Research on adversarial and attacker-induced drift specifically highlights that naive retraining methods are not necessarily robust to adversarial drift that an attacker deliberately created, and that a detector should find a way to distinguish between true distributional drift and an adversarial drift attempt so that it does not act on it [5]. The Diagnosis Agent's verdict should be that separation step, and all verdicts it gives are recorded along with arguments, to be checked by humans at a later time, not just the verdict.

3.4 Leakage-Safe Retraining Data Curation

If the Diagnosis Agent authorizes retraining, the Curation Agent assembles the batch. This stage is where a large share of published intrusion detection research has been shown to go wrong: encoders, scalers, and feature selectors fit across the full dataset before splitting train and test data leak label-correlated information into the test set, inflating reported accuracy by a meaningful margin. The Curation Agent enforces the same discipline that should apply at initial training time — fitting any encoder or scaler exclusively on the training partition of the new batch and reusing the previously fit transformations wherever the schema has not changed — so that repeated retraining cycles cannot quietly reintroduce the exact leakage problem the system was built to avoid in the first place. This stage also applies class-balancing where the diagnosis flagged a minority-class shift, following the same logic used in prior IDS work on correcting severe class imbalance before retraining.

3.5 Retraining, Validation, and Safe Rollout

The Retraining Agent executes either an incremental update or a full retrain, depending on the severity the Diagnosis Agent assigned. The resulting candidate model is never deployed directly. The Validation Agent evaluates it against a held-out recent window and a canary slice of live traffic, and specifically checks that performance has not regressed on any previously well-detected attack class — a guard against the common failure mode where adapting to new drift silently degrades detection of an older, still-relevant attack pattern, sometimes called catastrophic forgetting. This is only a model that clears this validation gate is promoted; anything else is rejected and logged, with the system falling back to the previous production model. The Audit Agent's log ties every promotion or rejection back to the original drift evidence and the Diagnosis Agent's rationale, so the whole cycle is reconstructable after the fact rather than being a black-box swap of one model for another.

4. How This Differs From Existing Approaches

The table below situates the proposed framework against the main existing strategies for handling drift in intrusion detection and against the closest agentic-AI work from adjacent domains.

Approach	How Drift Is Handled	Limitation This Paper Targets
Periodic/scheduled retraining	Model is retrained on a fixed calendar regardless of whether drift has occurred	Wastes compute when nothing has changed; reacts too slowly when drift is sudden [4]
Statistical drift detectors (ADWIN, DDM, PSI)	Flag a drift event from a shift in error rate or feature distribution	Detects that something changed but not what changed or why, and cannot tell benign drift from an attacker probing the model [5][6]
Active-learning adaptation	Selects a small, informative subset of new traffic for labeling and incremental update	Still needs a human-defined trigger for when to run, and provides no reasoning trace for why a batch was selected [8][2]
Representation-based drift adaptation (e.g., ReCDA)	Uses self-supervised representation learning to adapt the model without full relabeling	Strong technically, but decision-making is opaque; no explanation is produced for operators or auditors [1]
Agentic AI for infrastructure/config drift (e.g., RIVA)	LLM agent reasons over telemetry to verify whether live state matches intended state	Built for cloud configuration, not for model-level statistical drift or retraining decisions [7]
Proposed framework	LLM agent reasons over drift signals, distinguishes benign vs. adversarial drift, and authorizes a leakage-safe retraining cycle with a human-auditable rationale	Combines statistical rigor with explainable, accountable decision-making

5. Risks and Open Challenges

A system that can decide, on its own, to retrain and redeploy a security-critical model is exactly the kind of high-consequence autonomy that deserves scrutiny rather than enthusiasm. A few risks are central to this design and would need to be addressed directly in any implementation, not treated as footnotes.

- Poisoned retraining loops: an attacker who understands that the system retrains on recent traffic has an incentive to slowly shift traffic in a way designed to be classified as benign drift, using the retraining mechanism itself as the attack vector. The task of the Diagnosis Agent is to distinguish between a benign drift and an adversarial drift, and it is precisely the one feature that is the most appealing to an adaptive adversary, and its false negative rate (FNR) when attacked intentionally and adaptively must be measured, not assumed.
- The combination of adapting to new drifts can cause a slow drop in performance on non-drifted classes, which is referred to as catastrophic forgetting, if validation is not explicitly checked on the entire historical class set, but not only on the drifted classes.
- Under triggering and alert fatigue at the retraining level: If the Monitor Agent is too sensitive it may initiate too many diagnosis and retraining cycles to be operationally sustainable, requiring a tuning of severity thresholds and cooldown periods.
- Explanation reliability: An LLM-based Diagnosis Agent may give a plausible sounding, but incorrect, explanation, which is not a failure mode of the statistical detector as it is more likely to be harmful than a low-confidence flag — explanations should be tested for reliability using a ground truth drift label, not just qualitatively reviewed.

- Cost profile: retraining cycles, even incremental ones, are much more costly than a single inference from a classifier, so the added reasoning layer should be shown to be worth the overhead by comparing the cost of the framework to simpler periodic retraining baselines.

6. Evaluation Plan

This paper proposes an architecture and not a system – this path of intention is described here and not a result. The natural testbed is a benchmark such as CIC-IDS2018 or a comparable multi-week dataset, split by time rather than randomly so that later weeks stand in for genuine temporal drift, following the same evaluation logic used in prior drift-focused IDS work [4]. A second, controlled evaluation would inject synthetic adversarial drift — traffic deliberately perturbed to mimic benign patterns — to test specifically whether the Diagnosis Agent can distinguish it from organic benign drift, since this is the scenario ordinary temporal splits cannot exercise on their own. Baselines for comparison would include a fixed-schedule retraining pipeline, a statistical-detector-only pipeline that retrains automatically on any triggered alarm, and the full agentic pipeline proposed here, compared on post-drift F1 score, time-to-recovery after a drift event, number of unnecessary retraining cycles triggered, and — specific to the adversarial-drift scenario — the rate at which each approach can be manipulated into retraining on attacker-shifted data. This last metric is the one existing drift-adaptive IDS literature rarely reports, and it is the one this framework is specifically designed to improve.

7. Conclusion

Concept drift is not a peripheral concern for machine-learning-based intrusion detection — it is close to the central reason deployed models underperform their reported benchmark numbers. Statistical drift detectors are necessary but not sufficient, because they can say that something changed without saying whether it is safe to learn from. This paper has proposed an agentic AI framework that adds a reasoning and accountability layer on top of established drift statistics: an LLM-based diagnosis step that distinguishes benign evolution from adversarial manipulation, a curation step that carries forward leakage-safe preprocessing discipline into every retraining cycle rather than only the initial training run, and a validation gate that treats model promotion as something to be earned against held-out and canary evidence, not assumed. None of this removes the need for careful evaluation — particularly against an adaptive adversary who knows the retraining loop exists and might try to exploit it — but it reframes model maintenance for security systems as an accountable, auditable process rather than either a rigid calendar or a black-box autopilot, which is the direction both the drift-adaptation and agentic-AI-safety literatures appear to be converging on independently.

References

[1] "ReCDA: Concept Drift Adaptation with Representation Enhancement for Network Intrusion Detection," in Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2024).

[2] A. Kumar, K. W. Fok, and V. L. L. Thing, "ExpIDS: A Drift-adaptable Network Intrusion Detection System With Improved Explainability," arXiv:2509.20767, 2025.

[3] "Generative Active Adaptation for Drifting and Imbalanced Network Intrusion Detection" (NetGuard), arXiv:2503.03022, 2025.

[4] "Generative Active Adaptation for Drifting and Imbalanced Network Intrusion Detection," arXiv:2503.03022, 2025 (temporal F1 degradation result).

[5] "Learn to Adapt: Robust Drift Detection in Security Domain," arXiv:2206.07581.

[6] "Binary Anomaly Detection in Streaming IoT Traffic under Concept Drift," arXiv:2510.27304, 2025.

[7] "RIVA: Leveraging LLM Agents for Reliable Configuration Drift Detection," arXiv:2603.02345, 2026.

[8] "Managing Concept Drift in Online Intrusion Detection," CEUR Workshop Proceedings, Vol. 3962, Paper 42.

[9] A. Rath, "Agent Drift: Quantifying Behavioral Degradation in Multi-Agent LLM Systems Over Extended Interactions," arXiv:2601.04170, 2026.

[10] "Trustworthy Self-Composable Big-Data-as-a-Service: An LLM-Orchestrated Multi-Agent Framework for Automated Data Engineering, AutoML, MLOps Deployment, and Drift-Aware Lifecycle Optimization," arXiv:2606.17915, 2026.

[11] P. Trirat, W. Jeong, and S. J. Hwang, "AutoML-Agent: A Multi-Agent LLM Framework for Full-Pipeline AutoML," arXiv:2410.02958, 2025.

[12] Y. Chen, Z. Ding, and D. Wagner, "Continuous Learning for Android Malware Detection," in 32nd USENIX Security Symposium (USENIX Security 23), 2023.

[13] F. Bayram, B. S. Ahmed, and A. Kassler, "From Concept Drift to Model Degradation: An Overview on Performance-Aware Drift Detectors," Knowledge-Based Systems, vol. 245, 2022.

[14] "Evaluating Agentic AI in the Wild: Failure Modes, Drift Patterns, and a Production Evaluation Framework," arXiv:2605.01604, 2026.