Using Concept Drift Detection as an Operational Indicator of Compromise for Continuous Security Monitoring

Jane Piantoni
Faculdade de Tecnologia, Universidade Estadual de Campinas
Limeira/SP, Brasil
j272388@dac.unicamp.br

Abstract

AI-based Intrusion Detection Systems operate in dynamic environments in which evolving network traffic may affect predictive performance. This work investigates whether alerts generated by a concept drift detector can serve as complementary operational signals for continuous security monitoring. A Hoeffding Tree classifier combined with the Adaptive Windowing detector was implemented using the River framework and evaluated on chronological Brute Force and Denial of Service scenarios from the CIC-IDS2017 dataset under prequential evaluation. The analysis examines the temporal association between attack periods and changes in the prediction-error stream. Brute Force traffic produced concentrated alerts near the transition to predominantly malicious traffic, whereas sustained Denial of Service traffic produced recurrent alerts throughout the analyzed stream. These preliminary findings suggest that drift alerts may identify periods requiring additional security investigation. From a software-security perspective, this operational view illustrates how drift monitoring could be incorporated as a post-deployment assurance feature in AI-enabled applications. However, the evidence establishes temporal association rather than attack-specific detection; validation against benign changes and alert-quality metrics is required before such alerts can be treated as reliable indicators of compromise.

Keywords

Concept Drift, Intrusion Detection Systems, Continuous Monitoring, Online Learning, Cybersecurity, Runtime Assurance

1 Introduction

Artificial Intelligence (AI)-based Intrusion Detection Systems (IDSs) support the automatic analysis of large network-traffic volumes. After deployment, however, these systems operate under evolving conditions in which user behavior, services, workloads, traffic volume, and attack strategies may change. Such changes violate stationarity assumptions and may produce concept drift, i.e., changes in the statistical relationship between incoming data and the target concept [4]. If the deployed model cannot accommodate these changes, detection performance may progressively deteriorate [9].

Online learning and drift-aware mechanisms have therefore been investigated for maintaining IDS performance in non-stationary environments [1, 7, 8, 11]. Most studies use drift detectors to trigger adaptation or retraining, while comparatively less attention has been given to the operational meaning of the alert itself. A statistically significant change in the prediction-error stream may indicate behavior that differs from recent operating conditions, but legitimate changes may generate similar effects.

This work investigates whether concept drift alerts can be interpreted as *candidate* indicators of compromise (IoCs), or more conservatively as complementary operational signals requiring further investigation. A Hoeffding Tree classifier and the Adaptive Windowing (ADWIN) detector were implemented with River and evaluated on chronological Brute Force and Denial of Service (DoS) scenarios from CIC-IDS2017 under prequential evaluation.

The contribution is threefold: (i) an incremental IDS monitoring pipeline that exposes drift events from the prediction-error stream; (ii) an empirical comparison of temporal alert patterns under abrupt and sustained attack scenarios; and (iii) a security-oriented interpretation that distinguishes candidate operational signals from validated IoCs. From a secure software engineering perspective, the proposed pipeline can be interpreted as a runtime assurance mechanism for AI-enabled software. Importantly, this work does not evaluate the correctness or security of LLM-generated code itself. Rather, it investigates post-deployment behavioral monitoring that can complement code-level testing and verification in AI-assisted development workflows. By exposing changes in the prediction-error stream as auditable runtime telemetry, the mechanism can support correlation with application logs, model and configuration changes, deployment events, and security monitoring. In this work, an operational indicator is therefore understood as a candidate signal for investigation rather than standalone proof of compromise.

2 Background and Related Work

Incremental classifiers update their models sequentially without complete retraining. The Hoeffding Tree uses the Hoeffding bound to select splitting attributes in high-speed streams [3]. ADWIN complements incremental learning by maintaining an adaptive window and signaling statistically significant changes between sub-window means [2].

Prior studies use drift-aware learning to preserve IDS performance under evolving traffic [1, 7, 8, 11], but they do not establish drift alerts themselves as attack-specific security indicators. Concept drift may result from attacks or from legitimate changes in workloads, services, user behavior, class proportions, or traffic volume. This study therefore evaluates whether alert timing and recurrence can provide complementary evidence for analyst triage when correlated with other telemetry.

A related deployment challenge is the availability of ground-truth labels. Data-stream applications may experience *verification latency*, in which true outcomes become available only after a delay, and in more extreme settings subsequent observations may remain unlabeled. This issue is relevant to cybersecurity because confirmation of anomalous behavior may depend on later analyst investigation. A recent meta-analysis identifies hybrid, semi-supervised, and

active-learning approaches as promising directions for reducing reliance on immediately available labels in non-stationary streams [5]. This limitation is particularly relevant here because the monitored signal is the classifier's prediction error.

3 Experimental Methodology

3.1 Dataset and Threat Scenarios

The experiments use CIC-IDS2017 [10], which contains enterprise-like benign traffic and labeled attacks represented through flow-based features extracted with CICFlowMeter. Two chronological partitions were selected:

- **Brute Force scenario (Tuesday):** benign traffic interspersed with FTP-Patator and SSH-Patator activity, representing an abrupt transition associated with repeated authentication attempts.
- **DoS scenario (Wednesday):** traffic containing Hulk, GoldenEye, Slowloris, SlowHTTPTest, and Heartbleed activity, representing sustained and heterogeneous disruptions.

These scenarios compare drift behavior under one concentrated transition and one prolonged sequence of malicious behaviors; they do not represent broad attack coverage.

3.2 Online Monitoring Pipeline

The pipeline combines a Hoeffding Tree classifier with ADWIN, both provided by River [6]. Under the prequential, or test-then-train, protocol, each incoming flow is first classified, compared with its available label, and then used to update the model. For each instance t, the monitored binary error is $e_t = \mathbb{1}(\hat{y}_t \neq y_t)$ and is subsequently supplied to ADWIN. A drift alert is emitted when ADWIN detects a statistically significant change in the recent error distribution, with confidence parameter $\delta = 0.002$. Immediate labels are used here because they are available in the benchmark and are required to construct the error stream; this is an experimental assumption rather than an assumption about operational label availability.

The alert was not used to reset or replace the classifier. Alert indices were retained as operational events and compared with the chronological behavior of the attack scenarios, separating the classifier's predictive role from the detector's monitoring role. The evaluation preserves the chronological organization of the selected CIC-IDS2017 scenarios and follows a test-then-train stream rather than a shuffled batch evaluation. This configuration is intended to preserve temporal behavior and to separate predictive adaptation from the analysis of drift events as monitoring signals.

3.3 Evaluation Scope

The exploratory evaluation reports prequential accuracy, alert counts, and temporal distribution. Accuracy is used only to contextualize the prediction-error stream rather than as a comprehensive IDS benchmark; a broader predictive assessment would require class-specific precision, recall, and F1 measures, particularly under class imbalance. Because the design lacks a matched benign-only stream and formal event-level evaluation, alert precision, false-alert rate, attack coverage, and detection delay were not estimated. The results therefore indicate temporal association rather than attack-specific detection. Future validation should quantify alerts inside and outside attack intervals, attack coverage, first-alert delay, repeated alerts per attack, and class-specific predictive performance.

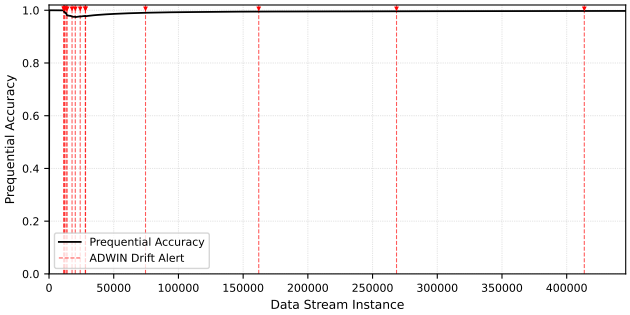

Figure 1: Prequential accuracy and ADWIN alerts in the Brute Force scenario.

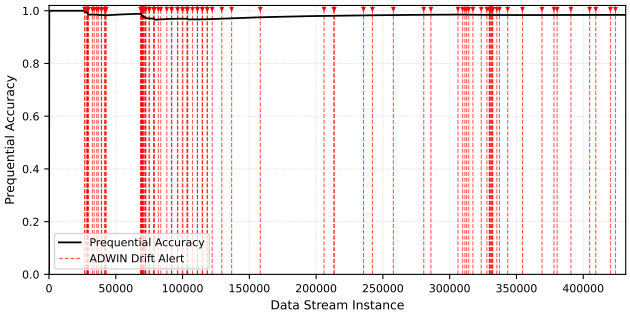

Figure 2: Prequential accuracy and recurrent ADWIN alerts in the DoS scenario.

4 Results and Discussion

4.1 Brute Force Scenario

Figure 1 shows the Tuesday stream. Fourteen drift alerts were observed, with several concentrated near the early transition in which malicious traffic became prominent and additional alerts later in the stream. Final prequential accuracy was approximately 97.4%.

The concentrated early pattern is consistent with an abrupt change followed by model stabilization. Nevertheless, without mapping each alert to labeled attack and benign intervals, the pattern should be interpreted as evidence of behavioral change rather than as a confirmed compromise indicator.

4.2 Denial of Service Scenario

Figure 2 presents the Wednesday stream. Ninety-nine drift alerts were distributed across the analyzed sequence, and final prequential accuracy was approximately 98.4%. In contrast to the Brute Force scenario, the recurrent pattern suggests persistent changes in the prediction-error distribution under heterogeneous and sustained traffic conditions.

Table 1: Observed drift patterns in the evaluated scenarios.

Scenario	Accuracy	Alerts	Temporal Pattern
Brute Force	≈97.4%	14	Concentrated early; isolated later
DoS	≈98.4%	99	Recurrent clusters across stream

The high alert count may capture persistent changes associated with successive DoS behaviors, but it may also indicate excessive sensitivity and generate alert fatigue. Its operational usefulness therefore depends on clustering, suppression rules, and correlation with independent security evidence.

Table 1 shows different temporal alert profiles despite similarly high final accuracy. This distinction supports the operational premise of the study: aggregate predictive performance and runtime stability are not equivalent. Monitoring the error stream may expose changes not apparent from a single final accuracy value, but the results do not demonstrate that drift alerts are reliable IoCs.

4.3 Threats to Validity and Operational Interpretation

The study provides evidence of temporal association rather than causal or attack-specific detection. Concept drift may arise from legitimate operational changes, and the absence of a benign-only baseline and event-level metrics prevents estimation of false alerts and comparison with conventional security detectors. A drift alert should therefore be treated only as a candidate operational signal and correlated with additional telemetry.

ADWIN is fed prediction errors and therefore depends on ground-truth labels. Although the benchmark provides y_t immediately, deployed IDS environments may exhibit verification latency, with labels arriving only after analyst investigation, or may provide labels for only a small fraction of the stream [5]. This limits direct transfer of the present pipeline to fully online deployment. Future work should therefore investigate delayed- and sparse-label settings. One direction is active learning, in which only selected informative instances are submitted for labeling, reducing the labeling burden. However, selective feedback also changes which prediction errors are observed by the detector, so its effect on drift-alert quality must be evaluated rather than assumed. Semi-supervised and label-independent monitoring statistics are complementary alternatives [5].

From a software-lifecycle perspective, concept drift monitoring could be integrated into the deployment and operation stages of AI-enabled applications. Drift alerts, model configuration, detector parameters, and alert provenance could be exposed through observability or security-monitoring platforms. This would allow development and security teams to correlate behavioral changes with application logs, authentication events, model updates, and infrastructure telemetry. Under this interpretation, drift detection becomes a complementary runtime security feature rather than only an internal mechanism for model adaptation. It does not replace secure coding, pre-deployment testing, or conventional intrusion detection, but extends assurance activities to the operational behavior of a deployed learning component.

4.4 Implications for AI-Assisted Secure Software Development

The experimental IDS evaluated in this study is not presented as an assessment of LLM-generated code, and Generative AI was not used to generate the IDS implementation. Its relevance to AI-assisted secure software development lies instead in post-deployment assurance of AI-enabled components. AI-assisted workflows may introduce frequent revisions to application code, model configurations, preprocessing logic, or deployment artifacts; security assurance must therefore extend beyond pre-deployment testing to the behavior observed after deployment.

Under this perspective, concept-drift events can be treated as runtime security telemetry. In an AI-assisted development pipeline, drift events could be recorded together with model versions, detector parameters, code or build identifiers, and deployment provenance, and then correlated with application logs and other security evidence. This would allow behavioral changes observed after an AI-assisted code or configuration revision to be investigated in relation to the corresponding software artifact. Drift alerts would not demonstrate that generated code is vulnerable or malicious; rather, they would provide complementary evidence that the operational behavior of an AI-enabled component has changed and may require investigation.

5 Conclusion and Future Work

This paper evaluated concept drift alerts as candidate operational signals for continuous security monitoring in an AI-based IDS. The Brute Force stream produced a smaller and more concentrated set of alerts, whereas the DoS stream produced recurrent clusters across the analyzed sequence.

The results suggest that drift monitoring may identify periods requiring additional investigation, but they do not establish attack-specific or reliable IoCs. They also motivate incorporating drift monitoring as a runtime assurance component in AI-enabled software, capable of publishing drift events and provenance to observability and security-monitoring platforms for correlation with other operational evidence.

Future work will investigate this integration as a runtime DevSecOps and MLOps control, including benign-only and mixed baselines, event-level alert precision and coverage, false-alert rates, detection delay, alert clustering, and correlation with independent security telemetry. Particular attention will be given to delayed- and sparse-label monitoring, including active-learning settings in which only a limited fraction of stream instances is queried for labeling, as well as semi-supervised and label-independent alternatives. Additional datasets, detector configurations, and alert-provenance mechanisms will also be evaluated. A further direction is to study explicitly AI-assisted development scenarios in which controlled code, model, or configuration revisions are correlated with runtime drift events and artifact provenance, assessing whether drift telemetry can complement conventional testing and code-level security analysis after deployment.

Artifact Availability

The scripts, experimental notebooks, configurations, and preprocessing instructions are available from the author upon request. CIC-IDS2017 must be obtained from its official source.

Acknowledgements

In accordance with the Brazilian Computer Society Code of Conduct, the use of Generative AI was limited to English-language revision, terminology refinement, and support for data-visualization scripts. All scientific contributions, experimental decisions, analyses, and conclusions remain the sole responsibility of the author.

References

[1] V. Agate et al. 2024. Enhancing IoT Network Security with Concept Drift-Aware Unsupervised Threat Detection. In *2024 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 1–6.

[2] Albert Bifet and Ricard Gavaldà. 2007. Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 443–448.

[3] Pedro Domingos and Geoff Hulten. 2000. Mining High-Speed Data Streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 71–80.

[4] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A Survey on Concept Drift Adaptation. *Comput. Surveys* 46, 4 (2014), 1–37.

[5] Mobin M. Idrees, Frederic Stahl, and Atta Badii. 2026. Adaptive Approaches to Mitigating Extreme Verification Latency in Dynamic Environments: A Research Meta-Analysis. *IEEE Access* 14 (2026), 91582–91605. doi:10.1109/ACCESS.2026.3702733

[6] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, and Albert Bifet. 2021. River: Machine Learning for Streaming Data in Python. *Journal of Machine Learning Research* 22, 110 (2021), 1–8.

[7] B. Nugraha, K. Yadav, and T. Bauschert. 2025. A Novel Adaptive Concept Drift Detection Approach for Evolving Network Traffic Patterns. In *2025 9th Cyber Security in Networking Conference (CSNet)*. IEEE, 1–8.

[8] S. Ranjeeth and A. Ahmed. 2026. Adaptive Intrusion Detection Using Ensemble Learning and Concept Drift Detection. In *2026 4th International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*. IEEE, 968–972.

[9] P. Shahad and E. D. Raj. 2021. Challenges in Streaming Data Analysis for Building an Adaptive Model for Handling Concept Drifts. In *2021 International Conference on System, Computation, Automation and Networking (ICSCAN)*. IEEE, 1–5.

[10] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*. 108–116.

[11] X. Yu et al. 2025. Malicious Attack Defense in Human-to-Machine Applications Through Concept Drift Adaptation. *IEEE Internet of Things Journal* 12, 24 (2025), 53563–53573.