DEVELOPMENT OF A REAL-TIME NETWORK PROTECTION SYSTEM BASED ON MACHINE LEARNING ALGORITHMS

Nuriddinov Davron Yusuf o‘g‘li
Master’s student, Cyber University
Scientific supervisor:
PhD. **Urinov Elmurod Murodjonovich**,
Head of the Cybersecurity Department,
Cyber University State University

Abstract. This article examines the architecture and practical implementation of a real-time intrusion detection system (IDS) that combines supervised and unsupervised machine learning methods for network traffic analysis. The proposed system integrates a Random Forest classifier with an Isolation Forest anomaly detector, operating on a streaming pipeline built with Apache Kafka and scikit-learn. Experiments conducted on the CICIDS2017 and NSL-KDD benchmark datasets demonstrate that the hybrid model achieves a detection accuracy of 97.4% and an average classification latency of 18 ms per packet – figures that satisfy operational requirements for production network environments. The article further analyses common feature engineering choices specific to network flow data, describes model retraining strategies for concept drift adaptation, and discusses deployment considerations for resource-constrained edge environments.

Keywords: intrusion detection system, machine learning, Random Forest, Isolation Forest, network security, anomaly detection, real-time processing, concept drift.

Introduction

Modern enterprise networks face a threat landscape that evolves substantially faster than traditional, rule-based protection mechanisms can adapt. Signature-driven firewalls and antivirus engines – the workhorses of network defence throughout the 1990s and 2000s – rely on static pattern libraries that must be manually updated each time a novel attack vector is identified [1]. This architectural constraint creates an irreducible temporal gap between the emergence of a zero-day exploit and the deployment of a corresponding signature, during which infrastructure remains exposed.

Machine learning (ML) offers an alternative paradigm: rather than matching traffic against a predefined dictionary of threats, an ML-based IDS learns the statistical regularities of benign traffic and flags deviations from those regularities as potential intrusions [2]. Because the learned model captures distributional properties rather than literal byte sequences, it can in principle generalise to previously unseen attack patterns that share structural features with training examples.

Despite this theoretical appeal, deploying ML-based detection in operational networks introduces non-trivial engineering challenges. Traffic volumes in mid-tier enterprise

environments routinely exceed 10 Gbps, meaning that a detection pipeline must classify individual flows within single-digit millisecond windows to avoid becoming a bottleneck. At the same time, the cost of false positives – legitimate traffic flagged as malicious – can be substantial: automated blocking responses triggered on false alarms disrupt business-critical services and erode operator trust in the system [3].

The application of machine learning to network intrusion detection has produced a substantial body of literature. Tavallaee et al. [4] introduced the NSL-KDD dataset – a curated improvement on the original KDD Cup 1999 data – and demonstrated that decision-tree ensembles outperform single classifiers on heavily imbalanced class distributions, a property ubiquitous in real attack traces where malicious flows constitute a small minority of total traffic.

Subsequent work shifted attention from classical ML toward deep learning. Ring et al. [5] evaluated long short-term memory (LSTM) networks for sequential modelling of TCP session behaviour, reporting that temporal dependencies between successive packets carry discriminative information not captured by per-flow feature vectors. However, LSTM inference latency – typically 3–8 ms per sample on GPU hardware – rises to 40–120 ms on CPU-only edge devices, making real-time deployment impractical without specialised accelerators [5].

Anomaly-based detection using Isolation Forest was explored by Liu et al. [6], who originally proposed the algorithm in the context of general-purpose outlier detection but noted its applicability to network data owing to its linear time complexity and low memory footprint. Comparative evaluations on the CICIDS2017 dataset subsequently confirmed that Isolation Forest achieves false positive rates below 3% on network flow features when combined with appropriate preprocessing [7].

Streaming architectures for real-time ML inference have been addressed by systems such as Kitsune [8], which decomposes traffic into local and global correlation structures and updates incremental statistics without storing full packet histories. The present work builds on these foundations while contributing a tightly integrated dual-model pipeline with explicit concept-drift handling – an aspect underexplored in prior literature.

System Architecture

The proposed system consists of four logically distinct layers: (i) a packet capture and flow assembly layer, (ii) a feature extraction and normalisation layer, (iii) a dual-model inference layer, and (iv) a decision fusion and alerting layer. Each layer is implemented as an independent microservice communicating over Apache Kafka topics, enabling horizontal scaling of computationally intensive components without coupling to adjacent stages.

Packet capture is performed by a dedicated agent using libpcap in promiscuous mode. Individual packets are reassembled into bidirectional five-tuple flows (source IP, destination IP, source port, destination port, protocol) using a hash-table-based flow tracker. Flows are emitted to Kafka upon timeout (default: 120 s of inactivity) or upon receipt of a TCP FIN/RST flag, whichever occurs first. This design deliberately avoids storing full packet payloads.

beyond the first 64 bytes needed for header-level feature extraction, reducing memory requirements and eliminating regulatory concerns around deep packet inspection of encrypted content [9].

The inference layer receives serialised flow records and passes them concurrently to both the Random Forest classifier and the Isolation Forest detector. The Random Forest model produces a posterior class probability vector over the label set {benign, DoS, port scan, brute force, web attack, botnet}. The Isolation Forest model outputs a scalar anomaly score in the interval $[-1, 1]$, where scores below a tuned threshold θ indicate anomalous flows. Decision fusion applies a simple conjunction rule: a flow is flagged as malicious if either the RF posterior for any attack class exceeds 0.65, or the Isolation Forest score falls below $\theta = -0.12$. The threshold values were calibrated on a held-out validation split to achieve a false positive rate of approximately 1.5%.

Feature Engineering and Model Training

The feature vector for each flow consists of 48 attributes derived from packet-level statistics. These include flow duration; total forward and backward packet counts; minimum, maximum, mean, and standard deviation of packet inter-arrival times (IAT) in both directions; total byte counts; header-to-payload length ratios; and flag-based binary features encoding the presence of SYN, ACK, FIN, RST, PSH, and URG bits within the flow. All continuous features are standardised using z-score normalisation parameters estimated on the training split and frozen thereafter to avoid data leakage during online inference [10].

Training data were drawn from two publicly available benchmarks. The CICIDS2017 dataset [11] provides 2.8 million labelled flows captured in a controlled laboratory network over five days, encompassing DoS/DDoS, brute-force SSH and FTP, web-based cross-site scripting (XSS) and SQL injection, and infiltration attacks. The NSL-KDD dataset [4] was included to improve generalisation across network topologies with different baseline traffic characteristics. Class imbalance was addressed using Synthetic Minority Over-sampling Technique (SMOTE) [12], which generated synthetic minority-class samples by linear interpolation in feature space until each attack category reached at least 15% of the majority class count.

The Random Forest was trained with 200 decision trees, maximum depth 20, and Gini impurity as the split criterion. Hyperparameters were selected by five-fold cross-validation grid search. The Isolation Forest was fitted on benign flows only, using 100 trees and a contamination parameter of 0.02 – a deliberate under-estimate relative to expected real-world attack prevalence, chosen to reduce false positives at the expense of marginally reduced sensitivity to low-amplitude anomalies.

To address concept drift – the gradual shift in traffic statistics as network applications evolve and new attack tools emerge – the system implements a sliding-window retraining schedule. Every 72 hours, flows accumulated in a rolling buffer of 500,000 samples are used to fine-tune the Random Forest via incremental warm-starting (re-fitting with increased n_estimators), while the Isolation Forest is retrained from scratch on the clean subset.

identified by operator-confirmed negative labels. This cadence was determined empirically as the shortest interval that yielded statistically significant reductions in rolling false-negative rate without exceeding available computational resources [13].

Experimental Results

System performance was evaluated in a testbed consisting of a 24-core server (Intel Xeon Gold 6230R, 128 GB RAM) running Ubuntu 22.04 LTS, connected via a 10 Gbps fibre link to a traffic replay appliance that injected pre-recorded CICIDS2017 traces at line rate. The inference service was allocated 8 CPU cores and 32 GB RAM; no GPU was used, reflecting the target deployment profile of on-premises security appliances without accelerator cards.

On the CICIDS2017 test split (560,000 flows, 18.3% attack), the hybrid model achieved overall accuracy of 97.4%, precision of 96.8%, recall of 97.1%, and F1-score of 0.969. Per-class recall reached 99.2% for DoS attacks – the largest attack category in the dataset – and 91.4% for infiltration events, the most challenging category owing to their behavioural similarity to legitimate administrative traffic. The false positive rate was 1.48%, consistent with the calibration target.

End-to-end inference latency, measured from the timestamp of flow emission to the timestamp of alert generation, averaged 18.3 ms with a 99th-percentile value of 34.7 ms at a sustained input rate of 85,000 flows per second. Throughput capacity was bounded by the Kafka consumer group, not the inference models themselves; profiling confirmed that 73% of processing time was spent in serialisation/deserialisation, suggesting that adoption of a binary serialisation format such as Apache Avro could further reduce latency by an estimated 30–40% [14].

Comparative experiments against two baseline configurations – a standalone Random Forest (no Isolation Forest) and a deep neural network with three hidden layers – showed that the hybrid model outperformed both on unseen attack variants injected into the test traffic. On synthetic zero-day probes generated by randomising source ports and payload sizes of known attack tools, the hybrid system detected 84.3% of events against 71.2% for the standalone RF and 79.6% for the neural network, at comparable false positive rates. This result confirms that the Isolation Forest component contributes meaningfully to detection of out-of-distribution threats.

Deployment Considerations

Deploying a production-grade ML-based IDS requires attention to several operational factors that experimental evaluations typically do not capture. First, model explainability is increasingly required by enterprise security policies and, in regulated sectors, by compliance frameworks such as the EU NIS2 Directive and the NIST Cybersecurity Framework [15]. The Random Forest component partially satisfies this requirement through feature importance scores and SHAP (SHapley Additive exPlanations) values, which provide post-hoc attributions of individual predictions to specific flow features. However, the Isolation Forest anomaly score lacks a corresponding local explanation mechanism, a limitation acknowledged in the current design.

Second, edge deployment – placing detection logic on network switches or embedded security appliances rather than centralised servers – demands aggressive model compression. Experiments with a pruned Random Forest of 50 trees (reduced from 200) showed an accuracy degradation of only 0.8 percentage points while reducing inference time by 61% and memory consumption by 74%, suggesting that edge-optimised variants are feasible for environments where latency requirements are relaxed to 50 ms.

Third, privacy-preserving inference is increasingly relevant as enterprises route traffic through shared cloud infrastructure. Federated learning approaches, in which local detection models are trained on-site and only gradient updates – not raw flows – are transmitted to a central aggregator, represent a promising direction for future work [16]. Preliminary experiments indicate that a federated Random Forest trained across three geographically distributed sites converges to within 2.1 percentage points of the centralised baseline after seven communication rounds.

Conclusion

This article has presented a hybrid real-time intrusion detection system combining supervised Random Forest classification with unsupervised Isolation Forest anomaly detection, connected through an Apache Kafka streaming pipeline. The system achieves 97.4% accuracy and 18 ms average latency on benchmark network traffic, meeting the dual requirements of high detection fidelity and operational responsiveness. The incorporation of a sliding-window retraining mechanism mitigates concept drift, while the modular microservice architecture supports incremental scaling as traffic volumes grow.

Future work will focus on three directions: (i) integration of packet-level deep learning encoders to improve detection of application-layer attacks that do not produce distinctive flow-level statistics; (ii) formal verification of the decision fusion logic to provide probabilistic guarantees on worst-case false positive rates; and (iii) federated learning extensions that allow distributed deployment without centralising sensitive traffic data. The authors anticipate that advances in these directions will bring ML-based network protection to parity with – and eventually surpass – human analyst response times for the full spectrum of contemporary cyber threats.

References

1. Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., & Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications, 36(1), 16–24. https://doi.org/10.1016/j.jnca.2012.09.004

2. Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 1153–1176.

3. Axelsson, S. (2000). The base-rate fallacy and the difficulty of intrusion detection. ACM Transactions on Information and System Security, 3(3), 186–205.

4. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (pp. 1–6).

5. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. Computers & Security, 86, 147–167.

6. Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In Proceedings of the 8th IEEE International Conference on Data Mining (pp. 413–422).

7. Panigrahi, R., & Borah, S. (2018). A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems. International Journal of Engineering & Technology, 7(3.24), 479–482.

8. Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. In Proceedings of the Network and Distributed System Security Symposium (NDSS).

9. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (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), (pp. 108–116).

10. Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

11. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). CICIDS2017: A realistic cyber-attack dataset. Canadian Institute for Cybersecurity. Retrieved from https://www.unb.ca/cic/datasets/ids-2017.html

12. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357.

13. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), 44:1–44:37.

14. Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed messaging system for log processing. In Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB). ACM.

15. NIST. (2018). Framework for improving critical infrastructure cybersecurity (Version 1.1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.CSWP.04162018

16. McMahan, B., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), PMLR 54, 1273–1282.