AN INTELLIGENT MACHINE LEARNING FRAMEWORK FOR CYBER THREAT DETECTION

Mayurkumar Rajeshbhai Thakkar
Lecturer, Computer Engineering
K. D. Polytechnic, Patan
mayurthakkar.er@gmail.com

Abstract

The increasing scale, speed and diversity of cyberattacks challenge security systems that rely mainly on fixed signatures and manually maintained rules. Machine learning (ML) can support cyber threat detection by learning statistical patterns from network flows, host events and authentication activity. This paper presents a literature-grounded, intelligent hybrid framework that combines supervised classification, unsupervised anomaly detection, feature selection, risk-based alerting and analyst feedback. The design is intended to detect known attack classes while identifying deviations that may represent previously unseen threats. The paper discusses commonly used algorithms, datasets, evaluation measures and deployment limitations. It proposes a reproducible evaluation protocol using temporally separated data, class-sensitive metrics and operational measures such as false alerts per unit time. No new experimental results are claimed; the framework is a design proposal derived from the reviewed literature. The central argument is that dependable detection requires more than high benchmark accuracy: it requires representative data, robust validation, interpretable alerts, continuous monitoring and human oversight.

Keywords—Cybersecurity, machine learning, intrusion detection, anomaly detection, network security, deep learning, threat intelligence.

1. INTRODUCTION

Digital services, cloud platforms, industrial networks and connected devices have expanded the attack surface available to adversaries. Threats such as denial-of-service (DoS), scanning, credential abuse, botnet activity, web attacks and lateral movement can produce large volumes of telemetry. Security information and event management (SIEM), firewalls and conventional intrusion detection systems (IDS) remain essential, but rule- and signature-based mechanisms may fail when attack behavior changes or has no known signature.

ML offers a complementary approach. Instead of relying only on predefined patterns, a model estimates relationships between traffic attributes and security outcomes. Supervised models learn from labelled examples; unsupervised models identify unusual behavior without requiring every attack type to be labelled; deep learning can learn complex representations from high-dimensional or sequential data. Reviews by Buczak and Guven [1], Liu and Lang [2], and later surveys [3], [4] show a broad range of methods, while also highlighting data quality, generalization and evaluation as persistent concerns.

This paper proposes a practical architecture that integrates multiple learning strategies rather than treating a single classifier as a complete security solution. Its scope is network- and host-event detection, with an emphasis on a deployable workflow.

1.1 Problem statement and objectives

The problem is to identify malicious or suspicious activity in noisy, imbalanced and evolving telemetry while keeping false positives manageable. The objectives are to (i) combine supervised and anomaly-based detection, (ii) define a reproducible data preparation and evaluation pipeline, (iii) prioritize alerts using calibrated risk, and (iv) support analyst feedback and model maintenance.

2. LITERATURE REVIEW

Buczak and Guven [1] surveyed data-mining and ML approaches for cyber intrusion detection, comparing methods such as decision trees, support vector machines (SVMs), neural networks and clustering. Their review emphasized that algorithm selection depends on the threat, data and operational context; no single technique is universally best.

Yin et al. [5] investigated recurrent neural networks for intrusion detection and evaluated binary and multiclass classification. Their work illustrates the value of sequence-aware models when relationships across observations matter. Shone et al. [6] proposed a nonsymmetric deep autoencoder for unsupervised feature learning and a stacked classifier, demonstrating a route to extracting representations before classification. These approaches suggest that deep models can reduce manual feature engineering, although their computational cost and dependence on training data must be considered.

Dataset quality became a central research issue. Sharafaldin et al. [7] introduced CICIDS2017, a labelled traffic dataset designed to represent benign traffic and multiple contemporary attack categories. It includes flow-based features and attack scenarios such as brute force, DoS, web attacks and infiltration. Such datasets support reproducible comparisons, but results obtained in a laboratory environment may not transfer directly to production networks.

Liu and Lang [2] reviewed ML and deep-learning methods for IDS, noting the importance of false alarm reduction and detection of unknown attacks. A supervised-learning survey by Khraisat et al. / related work [8] and a 2022 survey of supervised IDS techniques [9] describe the strengths of classifiers and the importance of feature selection and class imbalance. Prasad and Chandra [10] discuss datasets and challenges in applying ML to cyberattacks. Momand et al. [3] provide a 2023 survey of datasets, attack taxonomies and learning methods.

A further concern is adversarial robustness. He et al. [11] survey adversarial machine learning against network IDS and explain that carefully modified inputs can cause neural detectors to misclassify traffic. Consequently, a framework should consider evasion, poisoning, drift and model abuse, not only conventional test accuracy.

2.1 Synthesis and research gap

The literature points to four gaps relevant to deployment: (1) many studies evaluate on static, pre-labelled benchmarks; (2) random train/test splitting can leak temporal or host-specific patterns; (3) minority attack classes may be hidden by aggregate accuracy; and (4) alerts are not always interpretable or linked to response workflows. A hybrid architecture with time-aware validation, anomaly scoring and human feedback can address these issues at the system-design level. It does not eliminate them; rather, it makes them explicit and measurable.

2.2 Learning approaches

Random Forest and gradient-boosted trees are useful baselines for tabular flow features and can provide feature-importance estimates. SVMs can model nonlinear boundaries through kernels but may scale poorly on very large datasets. Logistic regression provides a transparent baseline. Autoencoders and Isolation Forest can model normal behavior and produce anomaly scores. CNNs can learn local patterns, while RNN/LSTM models can capture temporal dependencies [5], [6]. The proposed design uses a tree ensemble as the main supervised model and an anomaly detector as a complementary signal.

3. PROPOSED INTELLIGENT MACHINE LEARNING FRAMEWORK

The proposed framework, shown in Fig. 1, is organized as a modular pipeline. It ingests network-flow records, DNS and authentication events, endpoint telemetry and relevant security logs. Data collection should respect organizational policy, privacy requirements and data minimization. Each record is enriched with time, source, destination, protocol, asset criticality and—where available—threat-intelligence context.

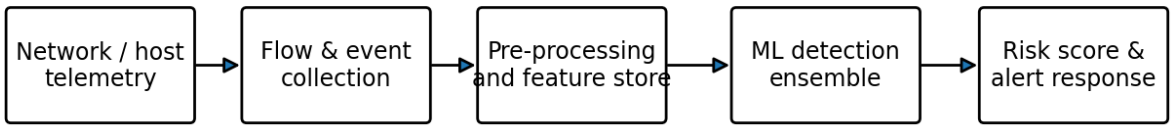


Fig. 1. High-level architecture of the proposed intelligent cyber threat detection framework.

3.1 Data acquisition and preprocessing

Network sensors and endpoint agents generate raw events. A normalization layer converts timestamps, protocol names and categorical fields into a consistent schema. Duplicate events are removed; missing values are handled explicitly; malformed or impossible values are flagged rather than silently accepted. Categorical variables may be one-hot or target encoded, while numerical features are scaled where required by the selected algorithm. Identifiers that could cause leakage (for example, a flow ID or a label-derived field) must be excluded.

Feature engineering derives flow duration, packet and byte counts, packet-rate statistics, connection state, inter-arrival times, failed-login counts and time-window aggregates. Features must be computed only from information available at prediction time. Feature selection can use mutual information, permutation importance or embedded tree importance. Selection is performed within the training folds to avoid leakage.

3.2 Hybrid detection engine

The supervised branch uses a Random Forest or gradient-boosted tree classifier trained on labelled benign and attack examples. It outputs class probabilities for categories such as DoS, scanning, brute force and web attacks. The unsupervised branch learns a profile of normal activity using an autoencoder reconstruction error or Isolation Forest anomaly score. A large anomaly score indicates deviation, not proof of maliciousness.

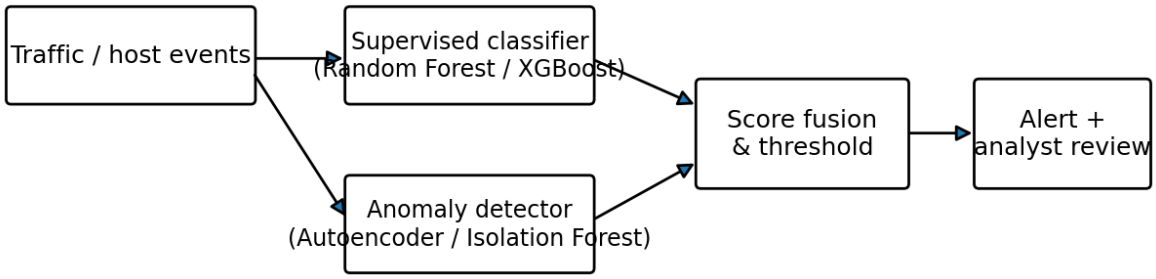


Fig. 2. Hybrid detection engine combining supervised classification and anomaly detection.

The two outputs are fused with asset criticality and contextual evidence. A conceptual risk score is $R = \alpha P + \beta A + \gamma C$, where P is the calibrated supervised attack probability, A is the normalized anomaly score, C is contextual risk, and α, β and γ are weights selected on validation data. The weights and thresholds should be tuned to operational costs, not chosen solely to maximize accuracy. The system should preserve the component scores so analysts can understand why an alert was raised.

3.3 Alert management

Alerts are grouped by entity and time window to reduce duplicates. Each alert includes the affected asset, observed features, predicted class, confidence, anomaly score, severity and evidence links. High-risk alerts may trigger a ticket or a controlled containment recommendation; automated blocking should require policy approval and safeguards against model error. Lower-confidence cases are routed for analyst review and feedback.

4. METHODOLOGY AND IMPLEMENTATION PLAN

The proposed study can be implemented as a reproducible experimental workflow using public datasets and, where authorized, sanitized organizational telemetry. CICIDS2017 is a suitable primary benchmark because it provides labelled benign and attack traffic [7]. UNSW-NB15 is another widely used dataset, although its original publication predates the selected reference window; it may be used as a benchmark artifact but is not counted as a cited research publication in this review. Dataset documentation, feature definitions and licensing conditions should be checked before use.



Fig. 3. End-to-end model development and deployment pipeline.

4.1 Experimental design

The dataset is divided into training, validation and test partitions. Whenever timestamps are available, a chronological split is preferred: earlier observations train the model, a later interval tunes hyperparameters and the latest interval estimates final performance. A separate cross-dataset test can assess transferability. Duplicate records and near-duplicates should be examined before splitting. Preprocessing, feature selection, resampling and calibration are fitted only on the training partition.

The supervised model is trained with class weights or training-only resampling to address imbalance. Candidate models include Logistic Regression, Random Forest, gradient boosting and an RNN/LSTM when ordered sequences are available. The anomaly

branch is trained primarily on benign training traffic. Hyperparameters are selected using validation data; the test set is used once for final reporting. A baseline signature/rule detector should be included where feasible.

4.2 Evaluation metrics

Accuracy alone is insufficient for imbalanced cybersecurity data. Precision = TP/(TP+FP) measures the proportion of alerts that are correct; recall = TP/(TP+FN) measures the proportion of attacks detected; F1 is their harmonic mean. False-positive rate (FPR) = FP/(FP+TN) measures benign events incorrectly flagged. PR-AUC is informative when attacks are rare, while ROC-AUC can complement it. For multiclass tasks, report per-class precision, recall and F1, macro-F1 and a confusion matrix. Operational reporting should also include detection latency, throughput, memory/CPU cost and false alerts per host or hour.

Thresholds are selected on validation data using a cost-sensitive objective that considers missed attacks and analyst workload. Confidence calibration (for example, isotonic regression or Platt scaling) can improve the interpretation of probability outputs. Results should include confidence intervals or repeated-run variability where computationally feasible.

4.3 Model lifecycle

After deployment, the system monitors feature distributions, class proportions, score distributions and alert outcomes. Drift triggers investigation and possibly retraining, but retraining must use reviewed labels and a controlled release process. Model versions, feature schemas, thresholds and training datasets should be recorded. Rollback is required if performance or stability degrades.

5. EXPECTED ANALYSIS, SECURITY AND DEPLOYMENT CONSIDERATIONS

The framework is a proposed architecture and has not been experimentally implemented in this paper. Therefore, no numerical accuracy, detection rate or superiority claim is made. Based on the reviewed literature, tree ensembles are reasonable candidates for structured flow data, while deep sequence models may be useful when temporal context is important [2], [5], [6]. The hybrid design is expected to improve coverage by combining known-class recognition with deviation detection, but it may also increase alert volume. This trade-off must be tested empirically.

5.1 Explainability and analyst trust

An alert should be accompanied by understandable evidence: influential features, relevant time window, peer baseline and the reason for the risk score. SHAP-style feature attribution can explain tree predictions, although explanations are approximations and must not be treated as causal proof. Analysts should be able to mark alerts as true positive, false positive, benign change or uncertain. Feedback should be quality-checked to avoid poisoning the training set.

5.2 Robustness and privacy

Attackers may attempt evasion by changing traffic rates, padding packets or distributing activity across hosts. Training data can also be poisoned if untrusted events are automatically incorporated. Defenses include input validation, access controls for training pipelines, robust validation, rate-limited model updates and adversarial testing [11]. Privacy controls include minimizing payload inspection, masking personal identifiers, restricting retention and limiting access to raw telemetry.

5.3 Limitations

Public benchmarks cannot represent every enterprise environment, and labels may be incomplete or generated under controlled conditions. Dataset-specific artifacts can inflate test performance. Concept drift, encrypted traffic, new protocols, sparse attacks and limited labelled examples remain difficult. Anomaly detection can flag legitimate but unusual behavior, while supervised models may miss novel attack families. Model ensembles also add maintenance and computational overhead. A production deployment therefore needs security engineering, analyst expertise and incident-response integration.

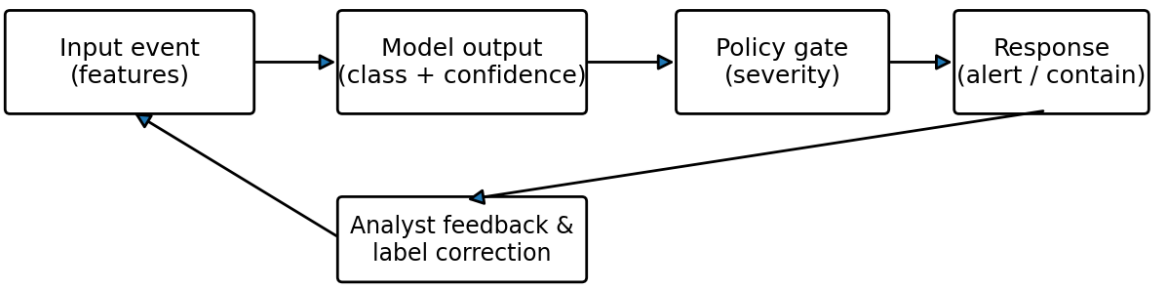


Fig. 4. Analyst feedback loop for review, response and model improvement.

5.4 Future research directions

Further work should evaluate the framework using time-separated and cross-network tests; compare calibration and thresholding strategies; measure analyst workload; and test robustness against evasion and poisoning. Federated learning may support collaboration without centralizing raw traffic, but it introduces aggregation and client-trust risks. Continual learning may adapt to drift, provided that model updates are validated and reversible. Explainable alert summaries and privacy-preserving telemetry are also important directions.

6. CONCLUSION

The proposed design combines normalized network and host telemetry, leakage-aware preprocessing, supervised classification, unsupervised anomaly scoring, risk-based alert fusion and analyst feedback. The literature indicates that Random Forests, boosting, SVMs and deep-learning models can support intrusion detection, but performance depends strongly on dataset quality, class balance, and feature design and evaluation methodology.

The main contribution is an integrated workflow that treats detection as a lifecycle rather than an isolated classification task. Temporal validation, per-class metrics, calibrated thresholds, explainable alerts, drift monitoring and human review are necessary to make ML outputs useful in operational security. Since this work proposes a framework rather than reporting a new experiment, future implementation should compare candidate models on CICIDS2017 and additional datasets using reproducible splits and operational metrics. A reliable cyber defense system should combine machine intelligence with security expertise, policy controls and continuous evaluation.

REFERENCES

[1] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016. doi: 10.1109/COMST.2015.2494502.

[2] H. Liu and B. Lang, “Machine learning and deep learning methods for intrusion detection systems: A survey,” Applied Sciences, vol. 9, no. 20, Art. 4396, 2019. doi: 10.3390/app9204396.

[3] A. Momand, S. U. Jan, and N. Ramzan, “A systematic and comprehensive survey of recent advances in intrusion detection systems using machine learning: Deep learning, datasets, and attack taxonomy,” Journal of Sensors, 2023, Art. 6048087. doi: 10.1155/2023/6048087.

[4] M. S. Habeeb and T. R. Babu, “Network intrusion detection system: A survey on artificial intelligence-based techniques,” Expert Systems, vol. 39, no. 9, 2022, Art. e13066. doi: 10.1111/exsy.13066.

[5] C. Yin, Y. Zhu, J. Fei, and X. He, “A deep learning approach for intrusion detection using recurrent neural networks,” IEEE Access, vol. 5, pp. 21954–21961, 2017. doi: 10.1109/ACCESS.2017.2762418.

[6] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A deep learning approach to network intrusion detection,” IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41–50, 2018. doi: 10.1109/TETCI.2017.2772792.

[7] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP), 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[8] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: Techniques, datasets and challenges,” Cybersecurity, vol. 2, Art. 20, 2019. doi: 10.1186/s42400-019-0038-7.

[9] “Intrusion detection systems using supervised machine learning techniques: A survey,” Procedia Computer Science, vol. 201, pp. 205–212, 2022. doi: 10.1016/j.procs.2022.03.029.

[10] A. Prasad and S. Chandra, “Machine learning to combat cyberattack: A survey of datasets and challenges,” Journal of Defense Modeling and Simulation, 2022. doi: 10.1177/15485129221094881.

[11] K. He, D. D. Kim, and M. R. Asghar, “Adversarial machine learning for network intrusion detection systems: A comprehensive survey,” IEEE Communications Surveys & Tutorials, vol. 25, no. 1, pp. 538–566, 2023. doi: 10.1109/COMST.2022.3233793.