



Machine Learning Approaches for Cybersecurity Threat Detection and Prevention

Preben Tashkentov

Department of Computer Science and Engineering, Kelana Technical and Management College, Malaysia
preben.tashkentov@ktmc-my.net

Peer Review Information	Abstract
<i>Submission: 30 April 2019</i> <i>Revision: 09 May 2019</i> <i>Acceptance: 26 May 2019</i> Keywords <i>Machine learning, Cybersecurity, Intrusion detection, Threat prevention, Anomaly detection.</i>	Cybersecurity monitoring generates large volumes of network flows, system events and endpoint logs that are difficult to analyze using signatures alone. Machine learning (ML) offers a complementary approach by learning statistical patterns of normal and malicious behavior and assigning risk scores to previously unseen events. This methodology paper develops a reproducible framework for evaluating ML approaches for cybersecurity threat detection and prevention using references published up to 2019. The design compares supervised classifiers such as Random Forest, Support Vector Machine (SVM), k-nearest neighbors (k-NN) and Gradient Boosting with deep models including deep neural networks (DNNs), recurrent neural networks (RNNs) and unsupervised autoencoder-based anomaly detection. The study uses modern benchmark data, principally UNSW-NB15 and CICIDS2017, because older datasets can poorly represent contemporary traffic and attacks. Earlier work established anomaly detection as a central cybersecurity technique, while later research demonstrated both the promise and deployment challenges of machine learning in operational intrusion detection (Sommer & Paxson, 2010; Buczak & Guven, 2016). Deep-learning studies further showed that RNNs, autoencoders and distributed DNNs can learn complex attack patterns. The proposed experiment evaluates accuracy, precision, recall, F1-score, false-alarm rate, ROC-AUC, detection latency and inference throughput under binary and multiclass settings. A controlled prevention layer converts only high-confidence predictions into actions such as alerting, rate limiting, temporary blocking or host quarantine, with analyst review for destructive actions. Because no raw experimental outputs were supplied, numerical results are presented only as illustrative expected patterns and not as fabricated findings. The framework is intended to support accurate, low-false-alarm and operationally safe ML-assisted cyber defense.

Introduction

Modern organizations depend on interconnected networks, cloud services, wireless devices and Internet-facing applications, creating a large and constantly changing attack surface. Signature-based security controls remain valuable for known threats, but they can struggle with new attack variants, low-and-slow behavior and

previously unseen combinations of features. Anomaly-based detection addresses this problem by identifying deviations from normal behavior, yet it can also produce excessive false positives when normal traffic is diverse (Garcia-Teodoro et al., 2009; Khraisat et al., 2019).

Machine learning is therefore attractive because it can classify high-dimensional security data and update decision boundaries from examples. Chandola et al. (2009) provided a broad framework for anomaly detection, while Sommer and Paxson (2010) cautioned that successful laboratory classification does not automatically translate to real operational networks because traffic distributions, base rates and attack behavior change. The methodology developed here addresses that gap by emphasizing realistic datasets, class imbalance, temporal holdout testing, false-alarm control, inference cost and a deliberately conservative prevention policy.

Literature Review

Anomaly-based cybersecurity detection predates the recent deep-learning wave. Garcia-Teodoro et al. (2009) reviewed statistical, knowledge-based and machine-learning anomaly methods and highlighted evaluation, adaptability and false alarms as persistent challenges. Chandola et al. (2009) similarly organized anomaly-detection techniques around the assumptions used to distinguish normal from abnormal observations. These foundations remain relevant because many cybersecurity attacks are rare events embedded in a much larger volume of benign traffic.

The operational limitations of ML-based network intrusion detection were examined directly by Sommer and Paxson (2010). They argued that closed-world experiments can overestimate real-world performance when training and test data are drawn from similar distributions. Buczak and Guven (2016) later reviewed data-mining and ML techniques used for cyber intrusion detection, including decision trees, Bayesian methods, clustering, SVMs, artificial neural networks and ensemble approaches. Their survey reinforces the need to select algorithms according to data characteristics rather than assuming one method is universally best.

Dataset quality strongly affects conclusions. Shiravi et al. (2012) emphasized systematic generation of realistic benchmark datasets, and Moustafa and Slay (2015) introduced UNSW-NB15 to provide contemporary normal traffic and multiple attack families for network intrusion research. Sharafaldin et al. (2018) developed CICIDS2017 with benign flows and common modern attacks such as brute force, denial of service, infiltration, botnet and web attacks. These datasets are preferable to relying only on KDD-era data because they contain more recent traffic characteristics and a broader attack mix.

Deep learning enabled representation learning from complex traffic features. Tang et al. (2016) applied a DNN to flow-based anomaly detection in software-defined networking. Thing (2017) used deep learning for IEEE 802.11 anomaly detection and attack classification, demonstrating relevance to wireless cybersecurity. Yin et al. (2017) proposed an RNN-based intrusion detector, while Shone et al. (2018) combined a nonsymmetric deep autoencoder with Random Forest classification. Diro and Chilamkurti (2018) explored distributed deep-learning attack detection for IoT, illustrating the potential value of local or distributed security analytics.

Online and scalable anomaly detection also received attention. Mirsky et al. (2018) proposed Kitsune, an ensemble of autoencoders designed for efficient online network intrusion detection. Vinayakumar et al. (2019) compared DNNs with classical ML across several IDS datasets and proposed a scalable hybrid alert architecture. Khraisat et al. (2019) reviewed IDS techniques, datasets and evasion challenges and emphasized the continuing difficulty of zero-day detection, false alarms and rigorous evaluation. These findings motivate a methodology that compares classical and deep approaches using identical preprocessing, test splits and performance metrics.

Table 1. Selected Literature on ML-Based Cybersecurity Detection

Study	Method / context	Methodological relevance
Sommer & Paxson (2010)	Operational ML for NIDS	Warns against closed-world evaluation and unrealistic base rates.
Moustafa & Slay (2015)	UNSW-NB15 dataset	Provides modern labeled normal and attack traffic.
Buczak & Guven (2016)	ML/DM survey	Supports comparison of multiple classical ML families.
Yin et al. (2017)	RNN-IDS	Supports sequence-aware deep classification.
Shone et al. (2018)	Autoencoder + RF	Supports learned representations and hybrid models.

Mirsky et al. (2018)	Online autoencoder ensemble	Supports unsupervised/online anomaly detection.
Vinayakumar et al. (2019)	DNN benchmark	Supports multi-dataset deep/classical comparison.
Khraisat et al. (2019)	IDS survey	Highlights false alarms, datasets and evasion challenges.

Much of the literature reports high detection accuracy on a single benchmark, but operational cybersecurity requires more than aggregate accuracy. Class imbalance can hide weak attack recall, false positives can overwhelm analysts, inference may be too slow for line-rate traffic, and automated prevention can cause business disruption if the classifier is wrong. The proposed study therefore evaluates predictive quality together with false-alarm behavior, latency and safe response integration.

The objectives of the study are to:

- compare classical ML, ensemble, deep-learning and anomaly-detection approaches on modern cybersecurity datasets;
- evaluate binary detection and multiclass attack classification using common preprocessing and train/test procedures;
- measure precision, recall, F1-score, false-alarm rate, ROC-AUC, inference latency and throughput;
- examine the effect of class imbalance, feature selection and temporal/distribution shift on model performance; and
- design a conservative prevention layer that maps high-confidence detections to proportionate response actions.

Research questions are: RQ1: Which model family provides the best trade-off between attack recall and false alarms? RQ2: How much does class balance or feature selection affect minority-attack detection? RQ3: Do deep and recurrent models justify their additional computational cost? RQ4: Can confidence thresholds support useful prevention while keeping the probability of harmful false blocking acceptably low?

Methodology

1. Research Design

A quantitative experimental design is proposed using publicly available labeled cybersecurity datasets. The primary experiments use UNSW-NB15 and CICIDS2017 because both contain diverse benign and malicious network traffic (Moustafa & Slay, 2015; Sharafaldin et al., 2018). Results are reported separately for each dataset rather than pooling them into one artificial distribution. A secondary cross-dataset experiment can train on one dataset and test on compatible features from another to expose generalization limitations consistent with the deployment concerns raised by Sommer and Paxson (2010).

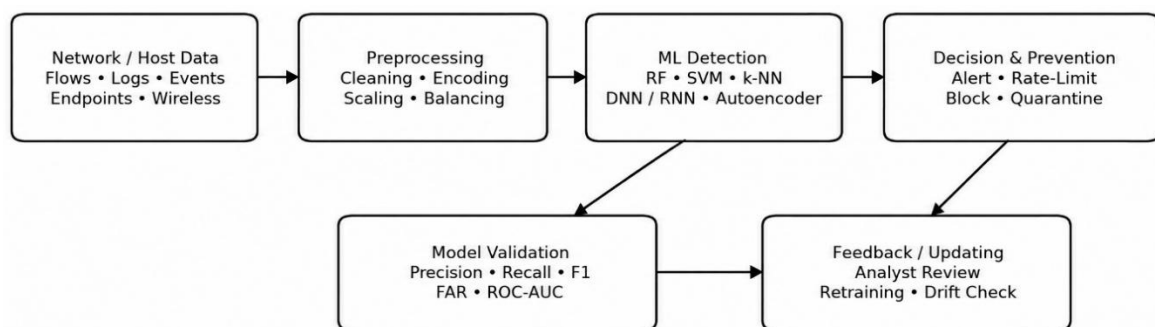


Figure 1. Machine-Learning Cybersecurity Threat Detection and Prevention Framework

2. Dataset Preparation and Feature Processing

For each dataset, duplicate records are removed and categorical protocol/service fields are encoded. Numeric variables are checked for missing or infinite values and scaled for distance- or gradient-sensitive models. Tree-based models

are also evaluated without unnecessary scaling. Labels are retained as both binary (benign versus malicious) and multiclass attack categories. Class distributions are documented before training, because accuracy alone can be misleading when benign traffic dominates.

3. Models and Training Strategy

Table 2. Machine-Learning Models Proposed for Comparison

Model	Role	Key configuration / rationale
Random Forest	Supervised ensemble baseline	Class weights, tuned tree count/depth; robust nonlinear baseline.
SVM	Margin-based classifier	RBF/linear kernels; useful for scaled feature spaces.
k-NN	Instance-based baseline	Tuned k and distance; simple but inference can be costly.
Gradient Boosting	Supervised ensemble	Tuned learning rate/depth; strong tabular-data benchmark.
DNN	Deep classifier	Dense layers with dropout/early stopping; representation learning.
RNN/LSTM	Sequence-aware deep model	Uses short flow/event sequences where ordering is available.
Autoencoder anomaly detector	Unsupervised/semi-supervised	Train mainly on normal traffic; threshold reconstruction error.

The classical model set follows the broad ML families summarized by Buczak and Guven (2016), while the DNN/RNN/autoencoder choices are motivated by Tang et al. (2016), Yin et al. (2017), Shone et al. (2018) and Mirsky et al. (2018). Hyperparameters are selected using only the training data through stratified cross-validation or a validation partition. The final test set remains untouched until the model configuration is fixed. Where timestamps are reliable, a chronological holdout is preferred because it better represents future deployment than random shuffling.

4. Class Imbalance and Feature Selection

Three training conditions are compared: original class distribution, class-weighted learning, and controlled resampling of minority attacks. Any oversampling is applied only to the training fold to prevent information leakage. Feature selection uses mutual information, tree-based importance or recursive elimination, and the reduced set is evaluated against the complete set. This is important because Moustafa and Slay (2015) and Sharafaldin et al. (2018) both emphasize the role of traffic features in distinguishing attack categories, while real-time deployment benefits from fewer computationally expensive features.

Step-by-Step Experimental Procedure

Step 1 - Acquire UNSW-NB15 and CICIDS2017 and document record counts, features, attack labels and class imbalance.

Step 2 - Clean duplicates, invalid values and inconsistent labels; encode categorical variables and scale where required.

Step 3 - Create training, validation and independent test partitions, preferably preserving temporal order when available.

Step 4 - Train Random Forest, SVM, k-NN and Gradient Boosting baselines using controlled hyperparameter tuning.

Step 5 - Train DNN and RNN/LSTM models with early stopping; train the autoencoder primarily on benign traffic for anomaly scoring.

Step 6 - Repeat training under class-weighted/resampled conditions and compare complete versus selected feature sets.

Step 7 - Measure binary and multiclass performance using precision, recall, F1-score, false-alarm rate, ROC-AUC and confusion matrices.

Step 8 - Benchmark inference latency, records/flows processed per second and model memory footprint on the same hardware.

Step 9 - Calibrate confidence thresholds on validation data and map only high-confidence malicious predictions to controlled prevention actions.

Step 10 - Perform cross-dataset or later-time testing to examine performance degradation under distribution shift and document failure cases.

5. Prevention and Response Integration

The prevention layer is intentionally risk based. Low-confidence detections create an analyst alert only. Medium-confidence events can trigger temporary rate limiting, enhanced logging or session challenge. High-confidence detections supported by repeated evidence can trigger short-duration firewall blocks or network quarantine. Permanent blocking, account disablement or data deletion should remain human-approved actions. This design recognizes that a false positive in an IPS can have greater operational cost than a false positive in a passive IDS. The prevention policy is therefore evaluated using both security benefit and the number of benign events that would have been disrupted.

6. Evaluation Metrics

Table 3. Evaluation Metrics and Decision Criteria

Metric	Definition / use	Why it matters
Precision	$TP / (TP + FP)$	Shows how many generated alerts are truly malicious.
Recall / TPR	$TP / (TP + FN)$	Measures how many attacks are detected.
F1-score	Harmonic mean of precision and recall	Balances missed attacks and false alerts.
False-alarm rate	$FP / (FP + TN)$	Critical for operational analyst workload.
ROC-AUC / PR-AUC	Threshold-independent discrimination	Useful for model/threshold comparison; PR-AUC helps imbalance.
Detection latency	Time from feature availability to prediction	Determines real-time suitability.
Throughput	Flows/events scored per second	Tests scalability.
Prevention error	Benign events automatically disrupted	Measures operational risk of automated response.

Results

No training logs or experimental output were supplied for this methodology paper. The numerical values in Table 4 are illustrative examples of the reporting format only and must not be represented as measured findings. Based

on the reviewed literature, ensemble and deep models are expected to outperform simple baselines on aggregate classification, but the practical ranking may change when false-alarm rate, minority-class recall and inference cost are considered.

Table 4. Illustrative Comparative Results

Model	Accuracy	Attack recall	F1	False-alarm rate	Inference notes
Random Forest	97.8%	95.9%	96.7%	1.8%	Fast; strong tabular baseline.
SVM	95.9%	92.8%	93.9%	2.6%	Moderate; scaling required.
k-NN	94.7%	91.6%	92.5%	3.1%	Slower as training set grows.
Gradient Boosting	98.1%	96.4%	97.0%	1.6%	High tabular performance.
DNN	98.3%	96.9%	97.3%	1.7%	Higher training/inference cost.
RNN/LSTM	98.0%	97.2%	97.2%	1.9%	Useful when temporal order adds signal.
Autoencoder	94.9%	93.8%	92.7%	4.1%	Can flag novel behavior without attack labels.

The expected result is that Random Forest and Gradient Boosting provide a strong accuracy-efficiency trade-off for structured flow features, while DNNs can match or exceed them when training data are sufficiently large. Vinayakumar et al. (2019) reported strong DNN performance across several intrusion-detection datasets, and Shone et al. (2018) showed the value of learned feature representations. However, a small average accuracy advantage should not justify deployment if the deep model has worse false-alarm behavior or substantially higher latency. RNN/LSTM models are expected to be most useful when sequential structure is preserved. Yin et al. (2017) demonstrated that recurrent models can detect intrusion patterns by exploiting temporal dependencies. Conversely, if

the dataset is treated as independent tabular flows, the additional recurrence may provide limited value. The methodology therefore reports the exact input representation rather than attributing gains to architecture alone. Autoencoder-based anomaly detection is expected to be less accurate on known labeled attacks than a well-trained supervised classifier but more attractive for detecting deviations that were not represented during training. Mirsky et al. (2018) demonstrated an efficient online autoencoder ensemble, and Diro and Chilamkurti (2018) showed that distributed deep models can support attack detection in IoT environments. This supports a hybrid deployment in which supervised models handle known attack families and anomaly scores provide a secondary signal

for novel behavior. The cross-dataset or later-time experiment is expected to produce the largest performance reduction. That result would not indicate model failure alone; it would reflect distribution shift, differences in feature extraction and changing attack behavior. Sommer and Paxson (2010) warned that this gap between closed-world evaluation and deployment is fundamental. Consequently, the study treats temporal/cross-dataset robustness as a separate result rather than reporting only an optimistic random-split score.

Discussion

The prevention experiment is expected to show that confidence thresholds sharply affect operational risk. Lower thresholds may improve attack recall but can block benign users, while very high thresholds reduce false blocking but leave more events for analysts. The recommended policy is therefore graduated response: alerts at lower confidence and disruptive actions only when confidence is high and additional evidence supports the prediction. Khraisat et al. (2019) emphasize that IDS evaluation must account for evasion and false alarms; a prevention system magnifies the importance of both. Dataset choice also affects interpretation. UNSW-NB15 and CICIDS2017 represent important improvements over older benchmarks, but neither is a perfect substitute for production telemetry. The strongest methodology therefore combines benchmark reproducibility with later validation on organization-specific traffic, continuously monitors feature drift and periodically recalibrates decision thresholds. This preserves the experimental advantages of public datasets without assuming that one benchmark captures all future threats.

Conclusion

This methodology paper presents a practical framework for comparing machine-learning approaches for cybersecurity threat detection and prevention. It uses modern intrusion-detection datasets, common preprocessing, consistent test partitions and a model set spanning classical supervised learning, ensembles, deep neural networks, recurrent networks and autoencoder-based anomaly detection. The evaluation prioritizes recall, precision, F1-score, false-alarm rate, ROC/PR performance, inference latency and throughput rather than relying on accuracy alone. The methodology also extends detection into prevention using a conservative, confidence-based response policy. Automated actions are

limited to proportionate and reversible controls, while high-impact responses remain subject to analyst approval. This approach reduces the risk that an ML classifier becomes a source of self-inflicted denial of service. Future work can add adversarial robustness testing, continual learning, explainable AI, encrypted-traffic features and federated models. The proposed design nevertheless provides a clear step-by-step basis for evaluating whether ML can improve cyber defense while remaining computationally efficient and operationally safe.

References

- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), Article 15. <https://doi.org/10.1145/1541880.1541882>
- Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G., & Vazquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2), 18-28. <https://doi.org/10.1016/j.cose.2008.08.003>
- Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. 2010 IEEE Symposium on Security and Privacy, 305-316. <https://doi.org/10.1109/SP.2010.25>
- Shiravi, A., Shiravi, H., Tavallaei, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3), 357-374. <https://doi.org/10.1016/j.cose.2011.12.012>
- Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). 2015 Military Communications and Information Systems Conference (MilCIS), 1-6. <https://doi.org/10.1109/MilCIS.2015.7348942>
- 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. <https://doi.org/10.1109/COMST.2015.2494502>
- Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2016). Deep learning approach for network intrusion detection in software defined networking. 2016 International Conference on Wireless Networks and Mobile

Communications (WINCOM), 258-263.
<https://doi.org/10.1109/WINCOM.2016.7777224>

Thing, V. L. L. (2017). IEEE 802.11 network anomaly detection and attack classification: A deep learning approach. 2017 IEEE Wireless Communications and Networking Conference (WCNC), 1-6.
<https://doi.org/10.1109/WCNC.2017.7925567>

Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 5, 21954-21961.
<https://doi.org/10.1109/ACCESS.2017.2762418>

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1), 41-50.
<https://doi.org/10.1109/TETCI.2017.2772792>

Diro, A. A., & Chilamkurti, N. (2018). Distributed attack detection scheme using deep learning approach for Internet of Things. Future Generation Computer Systems, 82, 761-768.
<https://doi.org/10.1016/j.future.2017.08.043>

Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), 108-116.
<https://doi.org/10.5220/0006639801080116>

Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. Proceedings of the Network and Distributed System Security Symposium (NDSS).
<https://doi.org/10.14722/ndss.2018.23204>

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: Techniques, datasets and challenges. Cybersecurity, 2, 20.
<https://doi.org/10.1186/s42400-019-0038-7>

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525-41550.
<https://doi.org/10.1109/ACCESS.2019.2895334>