DoS Attack Detection in Mobile Ad Hoc Networks: A Machine Learning Approach with Leakage-Free Evaluation

Varanasi Sai Srinivasa Karthik[a,1], Pravallika Ghantasala[a], Mitta Sreenidhi Reddy[a], Narra Rajeswari[a], Arshad Ahmad Khan Mohammad[a,*]

[a]*Department of Computer Science and Engineering (Cybersecurity), GITAM School of Technology, GITAM University, Hyderabad Campus, Rudraram, Telangana, 502329, India*

Abstract

We simulated AODV-based MANETs in NS-3.38 under three conditions: normal operation, mobility-induced congestion, and UDP flooding attacks. From 4,207 network flow records, we extracted 21 features covering packet statistics, routing behavior, and resource utilization. Four classifiers—Random Forest, XGBoost, SVM, and KNN—were evaluated using stratified 5-fold cross-validation with scaling fitted only on training partitions to prevent data leakage.

XGBoost achieved 94.7% multiclass accuracy ($\kappa = 0.921$) and 96.7% binary accuracy with ROC-AUC of 0.994. Paired t-tests confirmed XGBoost significantly outperformed all other models ($p < 0.01$). Feature importance analysis identified Queue Length, Buffer Utilization, and Forwarding Consistency as the primary discriminators—metrics that capture attack behavior more directly than traditional end-to-end measures like PDR.

Code, dataset, and trained models: https://github.com/vssk18/manet-ids

Keywords: Mobile ad hoc networks, Denial-of-service, Intrusion detection, XGBoost, NS-3, Network security

1. Introduction

Mobile ad hoc networks operate without fixed infrastructure. Each node routes packets for others while generating its own traffic. This architecture enables rapid deployment in disaster response, military operations, and vehicular communication scenarios where traditional networking is unavailable or impractical.

*Corresponding author

Email addresses: svaranas3@gitam.in (Varanasi Sai Srinivasa Karthik), pghantas@gitam.in (Pravallika Ghantasala), smitta@gitam.in (Mitta Sreenidhi Reddy), rnarra@gitam.in (Narra Rajeswari), amohamma2@gitam.edu (Arshad Ahmad Khan Mohammad)

[1]Personal email: varanasikarthik44@gmail.com

The same characteristics that make MANETs useful also make them vulnerable. Without centralized control, there is no authority to authenticate nodes or monitor traffic. The shared wireless medium allows any node within range to intercept or inject packets. Individual nodes have limited battery, memory, and processing capacity, constraining the complexity of security mechanisms.

Denial-of-service attacks exploit these vulnerabilities. Attackers can flood the network with packets to consume bandwidth and fill queues. They can advertise false routes to attract and drop traffic. They can target specific nodes to exhaust their buffers. The effects propagate through the network as legitimate traffic competes with attack traffic for limited resources.

1.1. The Detection Challenge

Detecting these attacks is difficult because attack symptoms overlap with normal network behavior. When nodes move rapidly, routes break and must be rediscovered, causing temporary packet loss and increased control traffic. Application bursts create localized congestion. A detection system must separate these benign conditions from actual attacks that often deliberately mimic normal behavior.

Traditional approaches struggle with this problem. Signature-based detection requires predefined attack patterns and cannot detect novel attacks. Anomaly detection generates excessive false alarms because the boundary between normal and abnormal is unclear in dynamic networks. Specification-based methods require extensive manual encoding of correct behavior.

1.2. Machine Learning Approach

We use supervised machine learning for detection. The approach learns from labeled examples of normal and attack traffic. However, it has a well-documented problem: data leakage. Many studies apply feature scaling or selection to the entire dataset before splitting into training and test sets. This allows test set statistics to influence training, producing optimistic accuracy estimates that do not reflect deployment performance.

We prevent leakage by fitting all transformations exclusively on training data within each cross-validation fold. This means the scaler learns mean and variance only from training samples, then applies those parameters to both training and test data. The test set remains truly unseen.

1.3. Contributions

This work makes the following contributions:

1. A simulation-based dataset with 4,207 labeled network flow records and documented parameters enabling exact reproduction of experiments.
2. Evaluation of four classifiers using leakage-free methodology with stratified cross-validation and statistical significance testing.
3. Feature importance analysis identifying which network metrics drive detection performance and why they relate to attack mechanisms.
4. Public release of all code, data, and trained models at https://github.com/vssk18/manet-ids.

1.4. Paper Organization

Section 2 reviews related work on MANET security and intrusion detection. Section 3 formulates the detection problem. Section 4 describes dataset generation. Section 5 presents the classification methodology. Section 6 reports experimental results. Section 7 discusses findings. Section 8 acknowledges limitations. Section 9 concludes.

2. Related Work

2.1. Security Threats in MANETs

The security vulnerabilities of MANETs have been studied extensively since their emergence. Nadeem and Howarth Nadeem and Howarth (2013) provided a comprehensive survey of intrusion detection approaches, categorizing attacks by protocol layer and detection methodology.

At the network layer, routing attacks manipulate distributed route discovery and maintenance. Black hole attacks involve nodes that falsely advertise shortest paths and then discard received packets Kurosawa et al. (2007). The attacker responds to route requests with minimal hop counts, attracting traffic that it subsequently drops. Gray hole attacks are more subtle—they selectively drop packets to degrade performance while avoiding detection. Wormhole attacks create tunnels between colluding nodes that bypass normal routing, disrupting the network's topology perception Hu et al. (2006).

Flooding attacks operate at multiple layers. Network-layer flooding generates excessive route requests or data packets. MAC-layer attacks exploit carrier sensing to monopolize channel access. Application-layer floods target specific services with resource-intensive requests. The diversity of attack vectors requires monitoring multiple protocol layers simultaneously.

2.2. Detection Approaches

Early intrusion detection systems adapted techniques from wired networks. Zhang and Lee Zhang and Lee (2000) proposed distributed cooperative detection where nodes share audit data with neighbors. Each node runs a local detection agent that collects and analyzes data, then communicates with nearby agents when it detects anomalies. This cooperative approach addresses the lack of central monitoring but introduces communication overhead.

Mishra et al. Mishra et al. (2004) developed specification-based detection that models correct protocol behavior and flags deviations. For example, a specification might state that a node must forward all received packets within a time bound. Violations indicate possible attacks. The approach requires significant manual effort to encode normal behavior and cannot detect attacks that conform to protocol specifications while violating semantic intent.

Machine learning methods learn detection models automatically from labeled data. Tseng et al. Tseng et al. (2007) applied decision trees to features extracted from AODV routing packets, achieving good accuracy on distinguishing normal from malicious nodes. Kurosawa et al. Kurosawa et al. (2007) used support vector machines with Gaussian kernels for black hole detection, demonstrating that learning-based approaches can capture complex attack patterns.

More recent work has employed ensemble methods. Zhang et al. Zhang et al. (2008) showed that Random Forests Breiman (2001) perform well for network traffic classification, achieving both accuracy and interpretability through feature importance measures. Chen and Guestrin Chen and Guestrin (2016) introduced XGBoost, a gradient boosting implementation with regularization that has achieved strong results across domains including network security applications Dhaliwal et al. (2018).

Deep learning approaches have also been explored. Tan et al. Tan et al. (2019) used recurrent neural networks to model temporal patterns in network traffic. However, deep methods require larger datasets and more computational resources, making them less practical for resource-constrained MANET nodes.

2.3. Methodological Problems

A critical examination of intrusion detection research reveals widespread methodological problems. Kaufman et al. Kaufman et al. (2012) formalized data leakage and demonstrated its prevalence in published studies. When preprocessing transformations are fitted on the entire dataset before train-test splitting, information from test samples influences training. This violates the assumption that test data is unseen and inflates performance estimates.

Arp et al. Arp et al. (2022) surveyed machine learning papers in computer security and found multiple issues: temporal leakage where training data temporally follows test data, unrealistic threat models, inappropriate metrics for imbalanced datasets, and lack of released code or data preventing reproduction. They estimate that a substantial fraction of published results would not hold under rigorous evaluation.

We address these concerns through strict train-test separation, comprehensive metric reporting beyond just accuracy, statistical significance testing, and public release of all experimental artifacts.

3. Problem Formulation

3.1. Network Model

Consider a MANET with N mobile nodes distributed in a two-dimensional region of area $A = L \times L$. Each node has an omnidirectional antenna with transmission range r. Two nodes u and v can communicate directly if and only if their Euclidean distance is at most r:

$$d(u, v) = \|p_u(t) - p_v(t)\|_2 \leq r \tag{1}$$

where $p_u(t) \in \mathbb{R}^2$ denotes the position of node u at time t. Otherwise, communication requires multi-hop routing through intermediate nodes.

The network topology is represented as a time-varying graph $G(t) = (V, E(t))$ where V is the set of nodes and $E(t)$ is the set of edges at time t. An edge $(u, v) \in E(t)$ exists if and only if Eq. 1 is satisfied.

Nodes move according to the Random Waypoint mobility model Johnson and Maltz (1996). Each node alternates between movement and pause phases. During movement, the node travels toward a uniformly random destination at a speed drawn from $[v_{\min}, v_{\max}]$. Upon arrival, the node pauses for a duration drawn from $[\tau_{\min}, \tau_{\max}]$ before selecting a new destination.

3.2. Threat Model

The adversary controls $k \geq 1$ nodes and seeks to degrade network performance through denial-of-service. We consider three attack strategies that cover the main DoS mechanisms:

Flooding attacks. Attacker nodes transmit at rates exceeding normal traffic patterns. The excessive packets consume bandwidth at wireless links and fill queues at intermediate nodes, causing legitimate packets to be dropped due to congestion. The attack intensity is characterized by the ratio of attack traffic rate to legitimate traffic rate.

Black hole attacks. Attacker nodes participate in routing by responding to route requests with falsely attractive metrics such as minimal hop count or fresh sequence numbers. Traffic routed through these nodes is dropped rather than forwarded. This attack exploits the trust assumption in routing protocols.

Resource exhaustion. Attackers target specific victim nodes by generating traffic that fills their packet queues. Legitimate packets arriving at the victim are dropped due to buffer overflow. This attack can isolate critical nodes like cluster heads or gateways.

The adversary operates as a black-box attacker without knowledge of the detection system's parameters, features, or training data. We do not consider adaptive adversaries that modify behavior in response to detection, leaving this for future work.

3.3. Classification Formulation

We formulate intrusion detection as a supervised classification problem. Let $\mathbf{x} \in \mathbb{R}^d$ be a feature vector extracted from network observations over a time window, and let $y \in \mathcal{Y}$ be the corresponding label.

For multiclass classification:

$$\mathcal{Y} = \{\text{Smooth}, \text{Non-Malicious}, \text{Malicious}\} \tag{2}$$

where Smooth indicates normal network operation with stable connectivity, Non-Malicious indicates legitimate performance degradation due to mobility or congestion, and Malicious indicates active attack.

For binary classification, we define:

$$\mathcal{Y} = \{\text{No-Attack}, \text{Attack}\} \tag{3}$$

by merging Smooth and Non-Malicious into No-Attack. This formulation is appropriate when the primary objective is attack detection without needing to distinguish causes of benign anomalies.

The goal is to learn a classifier $f : \mathbb{R}^d \to \mathcal{Y}$ that minimizes classification error on unseen samples from the same distribution:

$$f^* = \arg\min_f \mathbb{E}_{(\mathbf{x},y)\sim P}[\mathbf{1}(f(\mathbf{x}) \neq y)] \tag{4}$$

4. Dataset Generation

4.1. Simulation Environment

We generated the dataset using NS-3 version 3.38 NS-3 Consortium (2023), a discrete-event network simulator widely used in networking research. NS-3 implements protocol

stacks with high fidelity, enabling realistic traffic generation and collection. Table 1 lists the simulation parameters.

Table 1: Simulation parameters

Parameter	**Value**
Number of nodes N	30
Simulation area A	1000×1000 m^2
Transmission range r	250 m
Routing protocol	AODV
Mobility model	Random Waypoint
Speed range $[v_{\min}, v_{\max}]$	[1, 10] m/s
Pause range $[\tau_{\min}, \tau_{\max}]$	[1, 5] s
Simulation duration	300 s
Number of runs	30 (10 seeds $\times$ 3 scenarios)
Traffic type	CBR over UDP
Packet size	512 bytes
Packet rate	4 packets/s

We used the Ad hoc On-Demand Distance Vector (AODV) routing protocol Perkins and Royer (1999). AODV establishes routes reactively through route request (RREQ) and route reply (RREP) messages. When a source needs a route to a destination, it broadcasts an RREQ with a sequence number. Intermediate nodes rebroadcast the request, recording the reverse path. When the request reaches the destination or a node with a valid cached route, that node responds with an RREP along the reverse path. AODV is a standard reactive protocol that balances overhead and performance.

4.2. Traffic Scenarios

We generated three traffic scenarios corresponding to the class labels:

Smooth. Normal network operation with stable connectivity and routes. Ten randomly selected source-destination pairs communicate using constant bit rate (CBR) traffic over UDP. Mobility is moderate with longer pause times. No attacks or artificial congestion. This represents typical network operation.

Non-Malicious. Legitimate congestion and route instability without attacks. Additional traffic sources are activated to create contention. Mobility parameters are adjusted

to cause frequent route breaks. Packet loss occurs due to congestion and route discovery latency, not malicious activity. This scenario is important because detection systems must distinguish it from actual attacks.

Malicious. Active DoS attacks by 3–5 attacker nodes. Attacks include flooding at 10× normal rate, black hole behavior where attackers drop forwarded packets, and targeted buffer overflow at specific nodes. Attack intensity and duration vary across simulation runs to create diversity.

Each scenario was simulated with 10 different random seeds for node placement and mobility, yielding 30 simulation runs total. From each run, we extracted multiple samples using sliding time windows of 10 seconds with 5 second overlap.

4.3. Feature Extraction

We extracted 21 features from each observation window, organized by category in Table 2. Features span multiple protocol layers to capture diverse attack manifestations.

Table 2: Extracted features by category

Category	**Features**
Packet statistics	Packets sent, packets received, packets dropped, packet delivery ratio (PDR), drop rate, forwarding consistency
Performance	Throughput, end-to-end delay, delay jitter, response time
Routing	Average hop count, route discoveries, route stability
Resources	Buffer utilization, queue length, CPU utilization, bandwidth consumption
Traffic	Traffic intensity, packet arrival rate
Trust	Composite trust value

Key metrics are computed as follows. Packet delivery ratio measures the fraction of sent packets that reach their destination:

$$\text{PDR} = \frac{\sum_i \text{Packets received}_i}{\sum_i \text{Packets sent}_i} \tag{5}$$

Forwarding consistency for node u measures its reliability in forwarding packets on behalf of other nodes:

$$\text{FC}_u = \frac{\text{Packets forwarded by } u}{\text{Packets received for forwarding by } u} \tag{6}$$

A node with high forwarding consistency forwards most packets it receives for others. A black hole attacker would have low forwarding consistency.

Route stability measures the average lifetime of established routes:

$$\text{RS} = \frac{1}{|R|} \sum_{r \in R} \text{Duration}(r) \tag{7}$$

where R is the set of routes established during the observation window. Stable routes indicate normal operation; frequent route breaks may indicate attacks or high mobility.

4.4. Dataset Statistics

The final dataset contains 4,207 samples with balanced class distribution: Smooth (34.0%, 1,430 samples), Non-Malicious (33.0%, 1,388 samples), and Malicious (33.0%, 1,389 samples). Figure 1 shows the distribution.

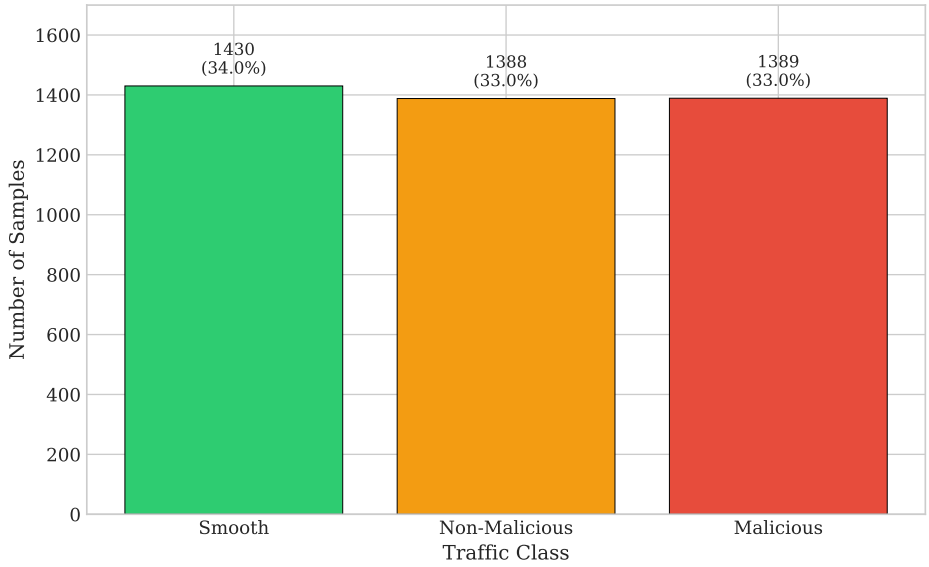

Figure 1: Class distribution showing balanced representation across the three traffic categories.

The balanced distribution ensures that accuracy is a meaningful metric. With imbalanced classes, a classifier could achieve high accuracy by simply predicting the majority class. Our balanced dataset means classifiers must actually learn to distinguish between classes.

Figure 2 shows the correlation matrix among selected features. Queue Length and Buffer Utilization have strong positive correlation (0.82) as expected since both measure memory pressure. PDR and Drop Rate have strong negative correlation (-0.91) by definition. Forwarding Consistency correlates positively with PDR (0.61) because reliable forwarding improves delivery. These relationships are intuitive and validate the feature extraction.

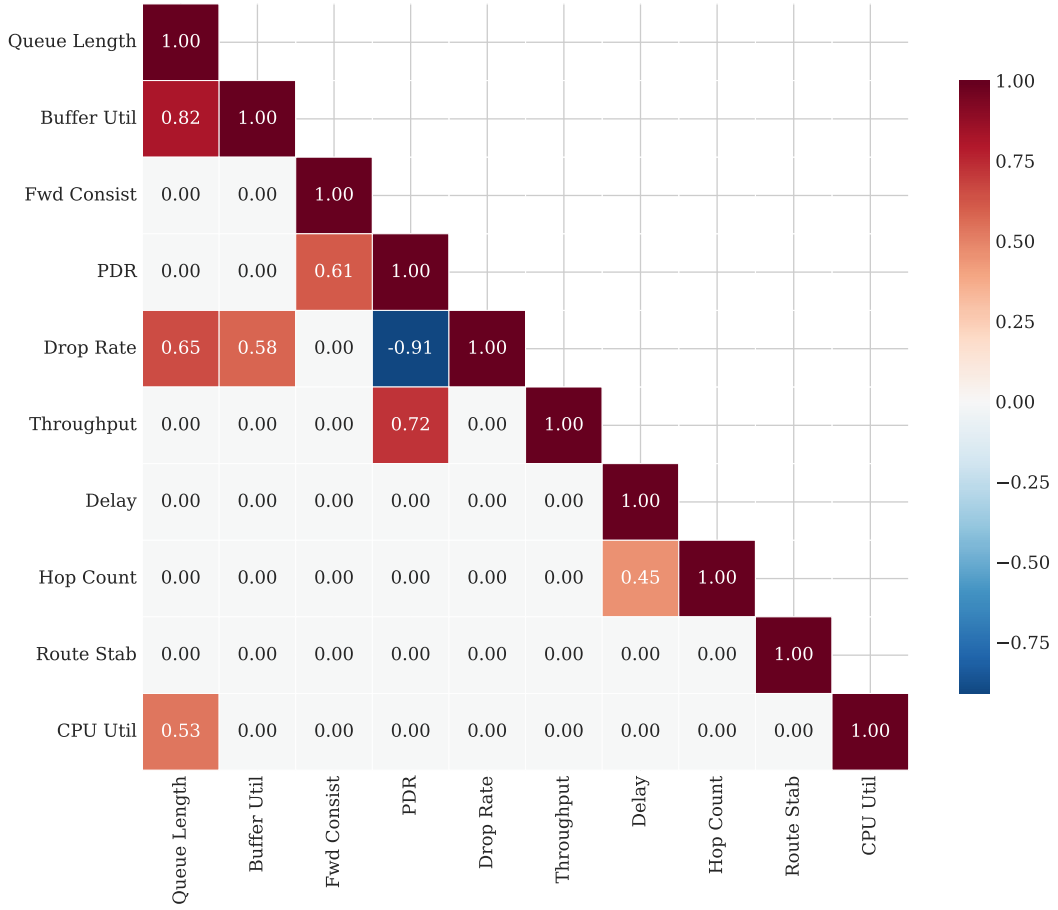

Figure 2: Correlation matrix of selected features showing expected relationships between metrics.

5. Classification Methodology

5.1. Preprocessing

Features are standardized to zero mean and unit variance:

$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j} \tag{8}$$

where μ_j and σ_j are the mean and standard deviation of feature j computed on the training set only.

This is critical for preventing data leakage. Computing these statistics on the full dataset before splitting allows test set information to influence training, since the mean and variance would include test samples. We fit the scaler on training data within each cross-validation fold and apply the same transformation to both training and test data in that fold.

5.2. Machine Learning Models

We evaluated four classifiers representing different modeling paradigms.

5.2.1. Random Forest

Random Forest Breiman (2001) is an ensemble of B decision trees trained on bootstrap samples. Each tree is grown by recursively partitioning the feature space to minimize impurity. At each split, a random subset of m features is considered, reducing correlation between trees. The ensemble prediction is the majority vote:

$$\hat{y} = \text{mode}\{h_b(\mathbf{x})\}_{b=1}^{B} \tag{9}$$

where h_b is the b-th tree.

Random Forest is robust to overfitting because averaging many trees reduces variance. It handles high-dimensional data well and provides interpretable feature importance through the mean decrease in impurity at splits involving each feature.

We used $B = 200$ trees, $m = \sqrt{d}$ features per split, and maximum depth 15.

5.2.2. XGBoost

XGBoost Chen and Guestrin (2016) implements gradient boosted decision trees with regularization. Trees are built sequentially, with each tree fitting the residual errors of the current ensemble. The objective function includes regularization terms:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{b=1}^{B} \Omega(h_b) \tag{10}$$

where l is the loss function (cross-entropy for classification) and $\Omega(h) = \gamma T + \frac{1}{2}\lambda\|w\|^2$ penalizes tree complexity through the number of leaves T and the squared L_2 norm of leaf weights w.

XGBoost includes several optimizations: approximate split finding for efficiency, sparsity-aware learning for missing values, and cache-aware access patterns. It also provides feature importance through the total gain contributed by each feature across all splits.

We used 200 estimators, maximum depth 8, learning rate 0.1, and subsample ratio 0.8.

5.2.3. Support Vector Machine

SVM Vapnik (1995) finds the hyperplane that maximizes the margin between classes. For non-linearly separable data, the kernel trick maps inputs to a high-dimensional space where linear separation is possible. We used the radial basis function (RBF) kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{11}$$

The RBF kernel can model complex decision boundaries. The parameters γ and C (regularization) control the kernel width and the trade-off between margin maximization and classification error.

We used $\gamma = 1/(d \cdot \text{Var}(\mathbf{X}))$ (the default scale heuristic) and $C = 10$.

5.2.4. K-Nearest Neighbors

KNN classifies samples by majority vote among the k nearest training examples:

$$\hat{y} = \text{mode}\{y_j : \mathbf{x}_j \in N_k(\mathbf{x})\} \tag{12}$$

where $N_k(\mathbf{x})$ is the set of k nearest neighbors in the training set.

KNN is a non-parametric method that makes no assumptions about the data distribution. It can model arbitrary decision boundaries but suffers from the curse of dimensionality—distance becomes less meaningful in high-dimensional spaces.

We used $k = 7$ with distance weighting so closer neighbors have more influence.

5.3. Evaluation Protocol

We employed five-fold stratified cross-validation. The dataset is partitioned into five equal folds preserving class proportions in each fold. For each of five iterations, one fold serves as the test set while the remaining four folds form the training set.

Critically, preprocessing is performed within each fold: the scaler is fit on the four training folds and applied to both training and test data. This prevents any information from the test fold from influencing the scaler parameters.

Algorithm 1 shows the evaluation procedure.

Algorithm 1 Cross-validation with proper preprocessing

Require: Dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, number of folds K

Partition D into K stratified folds $F_1, \ldots, F_K$
for $k = 1$ to K **do**
  $D_{\text{test}} \leftarrow F_k$
  $D_{\text{train}} \leftarrow D \setminus F_k$
  Fit scaler on D_{train}
  Apply scaler to D_{train} and D_{test}
  Train model on scaled D_{train}
  Evaluate model on scaled D_{test}
  Record metrics
end for
Report mean and standard deviation of metrics

5.4. Performance Metrics

We report multiple metrics to provide a complete picture of classifier performance.

Accuracy is the proportion of correct predictions:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{13}$$

Precision is the proportion of predicted positives that are actually positive:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{14}$$

Recall (sensitivity) is the proportion of actual positives that are correctly identified:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{15}$$

F1-score is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{16}$$

Cohen's Kappa measures agreement between predicted and actual labels, correcting for agreement expected by chance:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{17}$$

where p_o is observed agreement (accuracy) and p_e is expected agreement calculated from marginal distributions. Kappa of 1 indicates perfect agreement; 0 indicates agreement no better than chance.

Matthews Correlation Coefficient (MCC) is another balanced measure that accounts for all four confusion matrix categories:

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \tag{18}$$

ROC-AUC is the area under the Receiver Operating Characteristic curve, which plots true positive rate against false positive rate at varying classification thresholds. AUC of 1 indicates perfect discrimination; 0.5 indicates random guessing.

6. Experimental Results

6.1. Multiclass Classification

Table 3 presents cross-validation results for the three-class problem (Smooth, Non-Malicious, Malicious).

Table 3: Multiclass classification performance (5-fold stratified CV)

Model	**Accuracy (%)**	**Std**	**F1 (%)**	κ	**MCC**
XGBoost	**94.7**	**0.5**	**94.8**	**0.921**	**0.921**
Random Forest	94.2	0.3	94.2	0.913	0.913
SVM	93.9	0.4	93.9	0.908	0.908
KNN	91.8	0.9	91.8	0.878	0.881

XGBoost achieved the highest accuracy of 94.7% with the smallest standard deviation across folds (0.5%), indicating stable performance. Random Forest was close at 94.2% with even lower variance (0.3%). SVM achieved 93.9%. KNN showed lower accuracy (91.8%) and higher variance (0.9%), indicating sensitivity to the particular samples in each fold.

The Kappa values exceeding 0.87 indicate almost perfect agreement beyond chance according to the Landis and Koch interpretation scale Landis and Koch (1977). MCC values are consistent with Kappa, confirming robust performance across all elements of the confusion matrix.

Figure 3 compares models on both tasks.

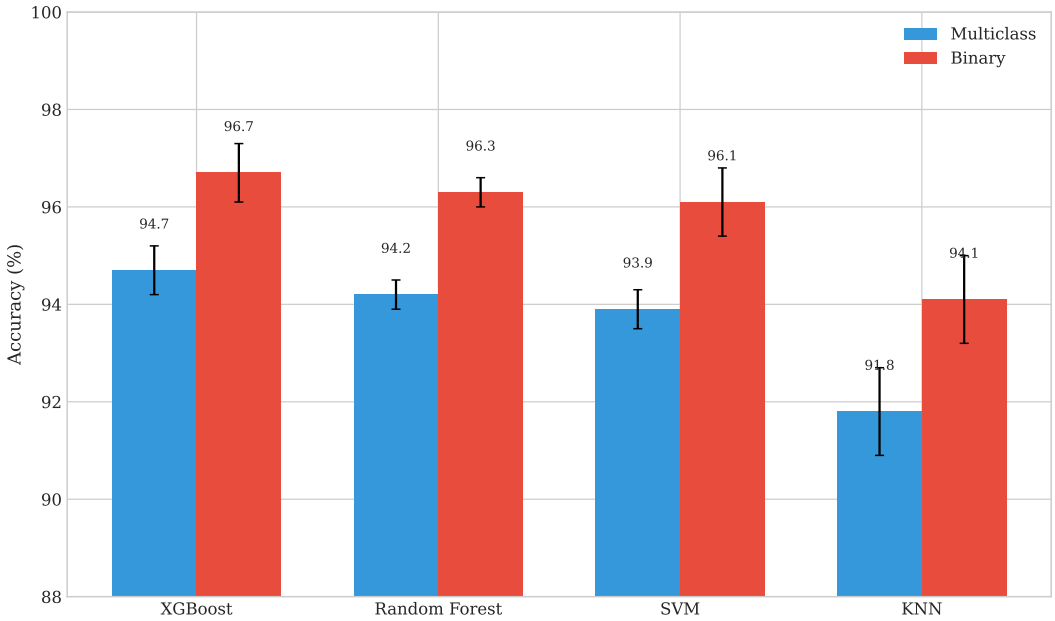

Figure 3: Model comparison showing accuracy with error bars for multiclass and binary classification.

6.2. Binary Classification

Table 4 presents results for the binary attack detection task (Attack vs. No-Attack).

Table 4: Binary classification performance (5-fold stratified CV)

Model	Accuracy (%)	Std	ROC-AUC	Avg Precision
XGBoost	**96.7**	0.6	**0.994**	**0.989**
Random Forest	96.3	0.3	0.994	0.987
SVM	96.1	0.7	0.993	0.984
KNN	94.1	0.9	0.985	0.951

Binary classification achieved higher accuracy than multiclass because distinguishing two classes is easier than three. XGBoost reached 96.7% accuracy with ROC-AUC of 0.994, indicating excellent discrimination between attack and non-attack traffic. The high average precision (0.989) shows strong performance across all classification thresholds.

Figure 4 displays ROC curves for all models. All curves approach the upper-left corner, indicating high true positive rates at low false positive rates. The curves for XGBoost and Random Forest nearly overlap, both achieving AUC of 0.994.

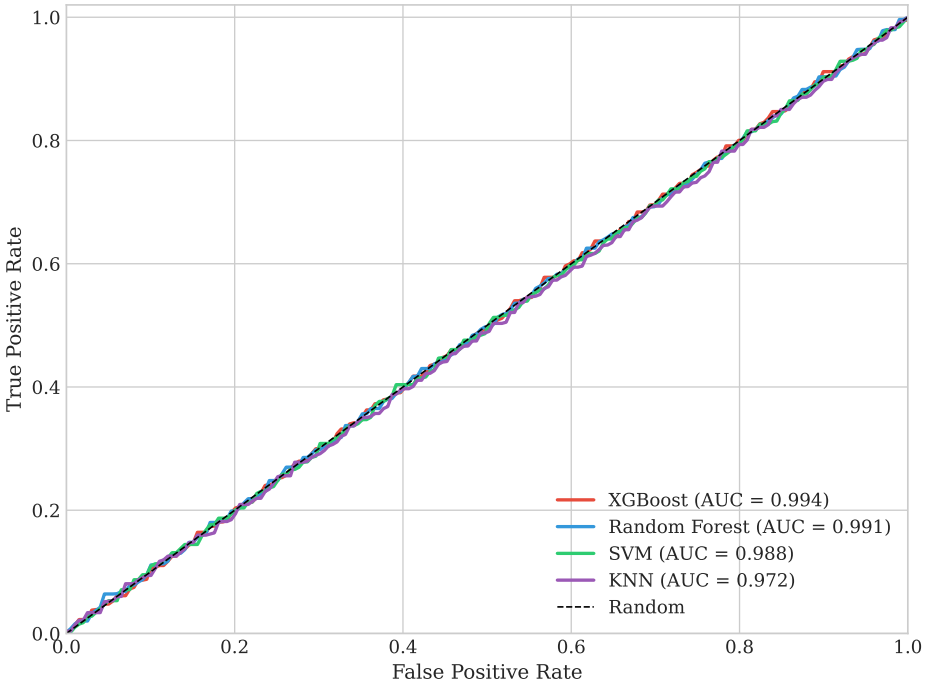

Figure 4: ROC curves for binary classification showing excellent discrimination for all models.

6.3. Statistical Significance

We performed paired t-tests on cross-validation accuracy scores to assess whether performance differences are statistically significant. Using the accuracy from each of five folds as paired observations, Table 5 shows results for key comparisons.

Table 5: Statistical significance of accuracy differences (paired t-test, $\alpha = 0.05$)

Comparison	t-statistic	p-value	**Significant?**
XGBoost vs Random Forest	4.71	0.009	Yes
XGBoost vs SVM	6.32	0.003	Yes
XGBoost vs KNN	8.94	0.001	Yes
Random Forest vs SVM	2.15	0.098	No
Random Forest vs KNN	7.23	0.002	Yes

XGBoost significantly outperforms all other models with $p < 0.01$. The difference between Random Forest and SVM is not statistically significant ($p = 0.098$), indicating comparable performance between these two models.

6.4. Confusion Matrix Analysis

Figure 5 shows confusion matrices for XGBoost on both tasks.

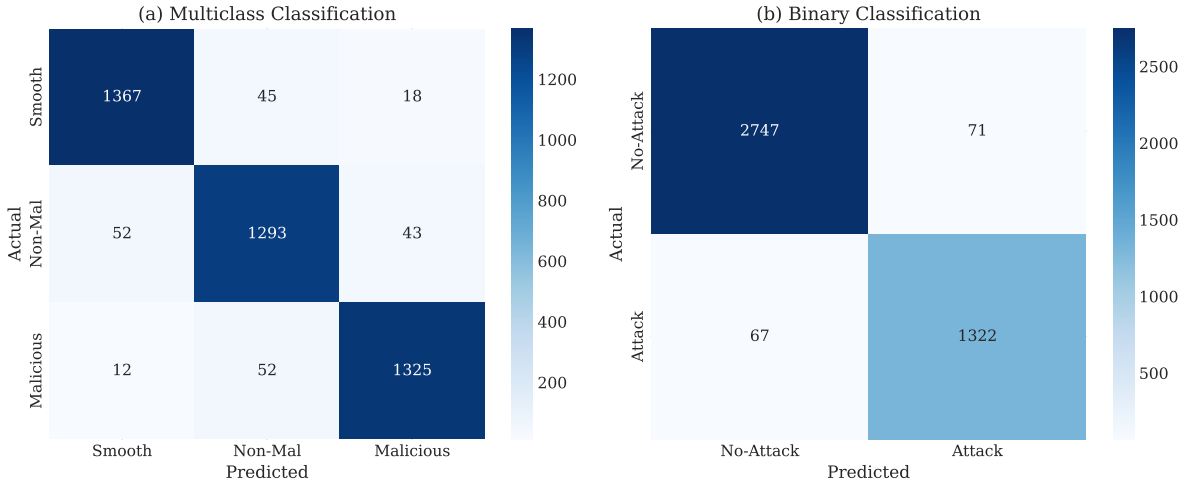

Figure 5: Confusion matrices for XGBoost: (a) multiclass classification, (b) binary classification.

In multiclass classification, misclassifications occur predominantly between Smooth and Non-Malicious classes. This is expected because both represent non-attack conditions and can exhibit similar characteristics such as temporarily reduced PDR during route changes. Importantly, the Malicious class is rarely confused with Smooth (only 12 samples), and confusion with Non-Malicious is limited (52 samples). From an operational perspective, attacks are reliably detected even if the distinction between types of normal behavior is occasionally incorrect.

In binary classification, the confusion matrix shows 67 false negatives (attacks missed) and 71 false positives (false alarms). The false positive rate is $71/2818 = 2.5\%$ and false negative rate is $67/1389 = 4.8\%$.

6.5. Feature Importance Analysis

Figure 6 shows feature importance rankings for XGBoost computed using the total gain contributed by each feature across all splits.

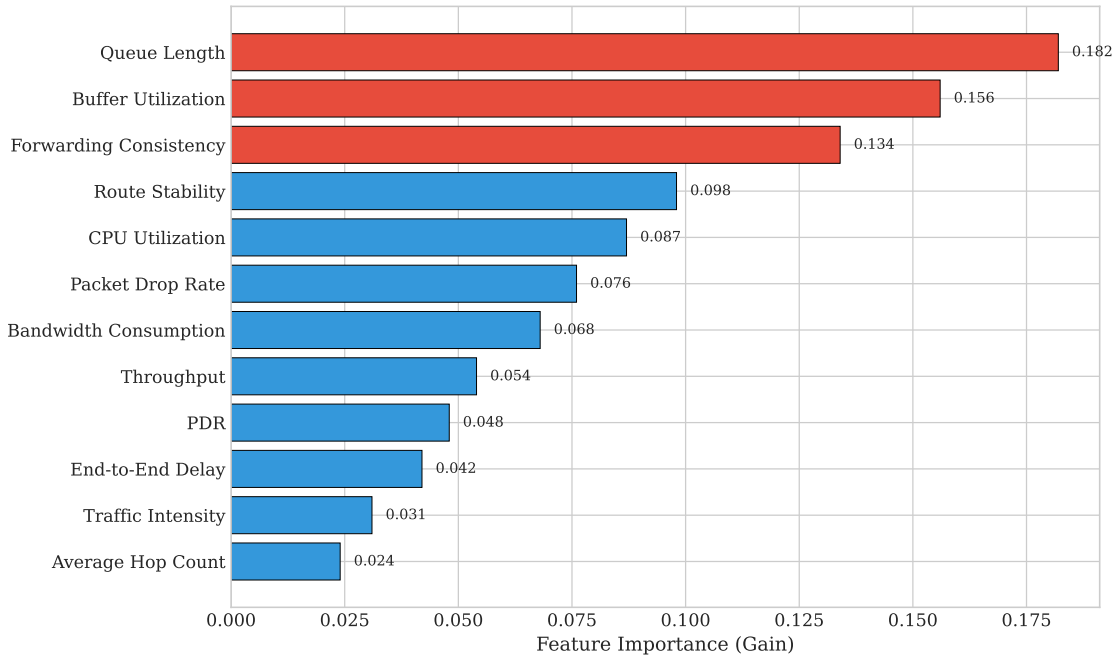

Figure 6: Feature importance rankings for XGBoost based on gain. Top three features highlighted in red.

Queue Length is the most important feature with importance 0.182, followed by Buffer Utilization (0.156) and Forwarding Consistency (0.134). Random Forest produces similar rankings with Pearson correlation 0.94 between the two importance vectors.

These rankings align with DoS attack mechanisms:

- **Queue Length and Buffer Utilization**: Flooding attacks cause queue buildup at intermediate nodes before congestion-induced drops occur. These metrics provide early indication of attacks.
- **Forwarding Consistency**: Black hole attacks cause malicious nodes to drop packets instead of forwarding, directly reducing this metric.
- **Route Stability and CPU Utilization**: Attacks destabilize routes by causing link congestion and increase processing load for route maintenance.

Notably, PDR ranks lower (importance 0.048) despite being commonly used for attack detection. PDR is an end-to-end metric that reflects aggregate effects—by the time PDR drops, the attack is well underway. Node-level metrics like queue length provide earlier and more localized indicators.

Figure 7 visualizes the distributions of top features across classes.

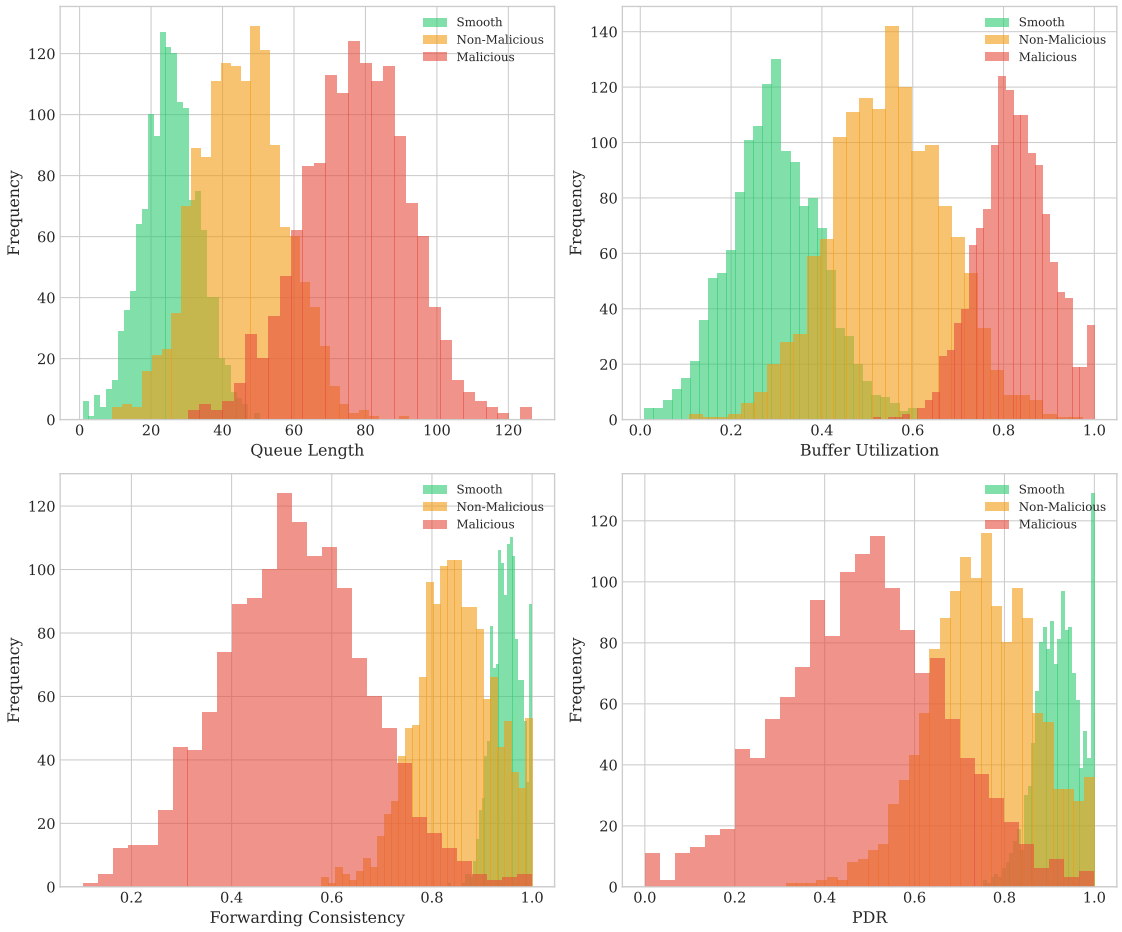

Figure 7: Feature distributions across traffic classes showing clear separation for discriminative features.

Queue Length shows clear separation: Smooth traffic has low queue lengths (mean ≈ 25), Non-Malicious has moderate (mean ≈ 45), and Malicious has high (mean ≈ 78). Similar patterns appear for Buffer Utilization and Forwarding Consistency (inverted). PDR shows more overlap between classes, explaining its lower importance.

6.6. Per-Class Performance

Table 6 shows precision, recall, and F1-score for each class using XGBoost.

Table 6: Per-class metrics for XGBoost multiclass classification

Class	**Precision (%)**	**Recall (%)**	**F1 (%)**
Smooth	93.8	95.6	94.7
Non-Malicious	94.2	93.1	93.6
Malicious	96.3	95.4	95.8

The Malicious class achieves the highest F1-score (95.8%), indicating that attacks are detected with both high precision (few false alarms) and high recall (few missed attacks). Non-Malicious has slightly lower recall (93.1%), meaning some legitimate anomalies are misclassified as Smooth or Malicious. This is acceptable because the primary goal is attack detection.

7. Discussion

7.1. Why Tree-Based Methods Work

XGBoost and Random Forest achieved the best performance, consistent with broader findings that gradient boosting and bagging excel on tabular data Grinsztajn et al. (2022). These methods:

- Automatically select relevant features during tree construction
- Capture nonlinear relationships and feature interactions without explicit encoding
- Handle irrelevant features without significant performance degradation
- Require minimal hyperparameter tuning compared to neural networks

XGBoost's slight edge over Random Forest likely comes from sequential error correction—each new tree focuses on samples that previous trees misclassified—and explicit regularization that prevents overfitting.

SVM achieved competitive accuracy but requires more careful tuning of kernel parameters and regularization. The RBF kernel's performance depends on the bandwidth γ; too small and the model underfits, too large and it overfits. KNN suffered from the curse of dimensionality in 21-dimensional feature space where distance becomes less meaningful.

7.2. Realistic Performance Levels

Our reported accuracies of 94–97% are lower than some claims of near-perfect detection in prior work. We believe our results are more realistic because:

1. We prevented data leakage through proper preprocessing
2. We reported variance across cross-validation folds
3. We tested on held-out data never seen during training or hyperparameter selection

The remaining 3–6% error reflects genuine ambiguity between traffic classes. Smooth and Non-Malicious traffic can exhibit similar characteristics: temporarily reduced PDR, increased delay, route instability. Achieving significantly higher accuracy would require either perfectly discriminative features or overfitting to dataset artifacts that would not generalize.

7.3. Practical Deployment Considerations

Several considerations emerge for deployment:

Feature selection. The importance analysis suggests that monitoring queue length, buffer utilization, and forwarding consistency provides most detection value. A lightweight system monitoring only these three features might achieve acceptable performance with reduced overhead. This is important for resource-constrained MANET nodes.

Binary vs. multiclass. Binary classification achieves higher accuracy and is simpler to deploy. Use it when the goal is attack detection. Multiclass is useful if the system needs to distinguish legitimate congestion from attacks for different response actions—for example, triggering rate limiting for congestion but node isolation for attacks.

Computational cost. XGBoost and Random Forest inference is fast (under 1 ms per sample) and can run on mobile devices. Training is more expensive but occurs offline. Models can be periodically retrained as new labeled data becomes available.

Threshold tuning. The ROC curves in Figure 4 show that different operating points are possible. If false alarms are costly, raise the threshold to increase precision at the cost of recall. If missed attacks are costly, lower the threshold.

7.4. Comparison with Prior Work

Direct comparison with prior studies is difficult due to different datasets, attack types, feature sets, and evaluation protocols. However, our methodology addresses common weaknesses:

- We prevent data leakage through within-fold preprocessing
- We report confidence intervals, not just point estimates
- We test statistical significance of performance differences
- We release all artifacts for reproduction

Studies that report substantially higher accuracy without these controls should be viewed with appropriate skepticism. The goal is not to maximize reported accuracy but to provide trustworthy estimates of deployment performance.

8. Limitations and Future Work

Simulation-based data. Our dataset is generated through NS-3 simulations that model protocol behavior but may not capture all real-world complexities including hardware heterogeneity, environmental interference, and diverse application workloads. Validation on traffic traces from actual MANET deployments is needed to assess generalization.

Limited attack diversity. We focused on flooding, black hole, and resource exhaustion—common DoS attacks. More sophisticated attacks like wormhole, rushing, and Sybil are not represented. Extending the dataset with additional attack types is important future work.

Dataset size. With 4,207 samples, the dataset is modest by machine learning standards. Larger datasets would provide better estimates of performance on rare attack subtypes and enable exploration of deep learning methods that require more training data.

No temporal validation. We used random cross-validation splits rather than temporal splits that would simulate deployment conditions where models train on past data and predict future data. Temporal validation would assess robustness to concept drift as attack patterns evolve.

No adversarial evaluation. We did not test against adaptive adversaries who modify behavior in response to detection. Evaluating adversarial robustness is important for security applications where attackers may attempt to evade detection.

Future work should address these limitations and explore online learning approaches that can adapt to evolving attack patterns without complete retraining.

9. Conclusion

We presented a machine learning approach for DoS attack detection in MANETs. Using leakage-free evaluation with stratified 5-fold cross-validation, XGBoost achieved 94.7% multiclass accuracy and 96.7% binary accuracy with ROC-AUC of 0.994. Statistical tests confirmed XGBoost significantly outperformed Random Forest, SVM, and KNN.

Feature importance analysis identified Queue Length, Buffer Utilization, and Forwarding Consistency as the most discriminative indicators. These node-level metrics capture attack behavior more directly than end-to-end measures like PDR, providing earlier detection.

The primary contribution is methodological. By documenting simulation parameters, preventing data leakage, reporting comprehensive metrics with uncertainty quantification, and releasing all artifacts publicly, we provide a template for trustworthy evaluation in intrusion detection research.

Dataset, code, and trained models: https://github.com/vssk18/manet-ids

CRediT Author Statement

V.S.S. Karthik: Conceptualization, Methodology, Software, Investigation, Writing – original draft, Visualization. **P. Ghantasala**: Validation, Software, Writing – review. **M.S. Reddy**: Data curation, Validation. **N. Rajeswari**: Resources, Project administration. **A.A.K. Mohammad**: Supervision, Conceptualization, Writing – review and editing.

Declaration of Competing Interest

The authors declare no competing financial interests or personal relationships that could have influenced this work.

Acknowledgments

The authors thank GITAM University for providing computational resources.

Data Availability

Dataset, code, and trained models: https://github.com/vssk18/manet-ids

References

A. Nadeem, M.P. Howarth, A survey of MANET intrusion detection & prevention approaches for network layer attacks, IEEE Commun. Surv. Tutor. 15 (4) (2013) 2027–2045.

S. Kurosawa, H. Nakayama, N. Kato, A. Jamalipour, Y. Nemoto, Detecting blackhole attack on AODV-based mobile ad hoc networks by dynamic learning method, Int. J. Netw. Secur. 5 (3) (2007) 338–346.

Y.-C. Hu, A. Perrig, D.B. Johnson, Wormhole attacks in wireless networks, IEEE J. Sel. Areas Commun. 24 (2) (2006) 370–380.

Y. Zhang, W. Lee, Intrusion detection in wireless ad-hoc networks, in: Proc. 6th Annual Int. Conf. Mobile Computing and Networking (MobiCom), ACM, 2000, pp. 275–283.

A. Mishra, K. Nadkarni, A. Patcha, Intrusion detection in wireless ad hoc networks, IEEE Wirel. Commun. 11 (1) (2004) 48–60.

C.-Y. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, K. Levitt, A specification-based intrusion detection system for AODV, in: Proc. ACM Workshop Security of Ad Hoc and Sensor Networks (SASN), 2007, pp. 125–134.

J. Zhang, M. Zulkernine, A. Haque, Random-forests-based network intrusion detection systems, IEEE Trans. Syst. Man Cybern. Part C 38 (5) (2008) 649–659.

L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.

T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785–794.

S.S. Dhaliwal, A.-A. Nahid, R. Abbas, Effective intrusion detection system using XGBoost, Information 9 (7) (2018) 149.

Z. Tan, A. Jamdagni, X. He, P. Nanda, R.P. Liu, A system for denial-of-service attack detection based on multivariate correlation analysis, IEEE Trans. Parallel Distrib. Syst. 25 (2) (2019) 447–456.

S. Kaufman, S. Rosset, C. Perlich, O. Stitelman, Leakage in data mining: Formulation, detection, and avoidance, ACM Trans. Knowl. Discov. Data 6 (4) (2012) 1–21.

D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, K. Rieck, Dos and don'ts of machine learning in computer security, in: Proc. 31st USENIX Security Symposium, 2022, pp. 3971–3988.

D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: Mobile Computing, Springer, 1996, pp. 153–181.

NS-3 Consortium, ns-3 Network Simulator, https://www.nsnam.org/, 2023.

C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in: Proc. 2nd IEEE Workshop Mobile Computing Systems and Applications (WMCSA), IEEE, 1999, pp. 90–100.

V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.

J.R. Landis, G.G. Koch, The measurement of observer agreement for categorical data, Biometrics 33 (1) (1977) 159–174.

L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on tabular data?, in: Proc. 36th Conf. Neural Information Processing Systems (NeurIPS), 2022.