Discrete & Continuous Models
& Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2026, 34 (2) 187–200
http://journals.rudn.ru/miph

Research article

UDC 004.89, 004.056.57
PACS 07.05.Mh, 07.05.Kf
DOI: 10.22363/2658-4670-2026-34-2-187-200 EDN: JDISSJ

Development of the malicious traffic detection system for mobile applications security

Nikita D. Khalemskii[1], Vitalii A. Beschastnyi[1,2]

[1] *HSE University, 20 Myasnitskaya St, Moscow, 101000, Russian Federation*

[2] *RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation*

(received: March 29, 2026; revised: March 16, 2026; accepted: March 30, 2026)

Abstract. *Background:* The rapid increase in cyberattacks targeting infrastructure, enterprises, and individual users through mobile devices has created an urgent need for effective intrusion detection systems. Traditional signature-based methods are inadequate against zero-day vulnerabilities and advanced persistent threats, prompting the exploration of machine learning approaches for network traffic analysis. *Purpose*: This study aims to develop and evaluate a privacy-preserving, locally operating Android application for real-time malicious traffic detection using machine learning models, addressing the limitations of cloud-dependent security architectures that compromise user privacy and availability. *Methods:* We implemented five machine learning algorithms — XGBoost, LightGBM, Random Forest, Decision Tree, and Logistic Regression — trained on the Network Traffic Android Malware dataset. The models were exported to the ONNX format for local deployment on Android devices. Performance was evaluated using accuracy, precision, recall, AUC score, training time, and model size metrics, with multi-criteria decision analysis (MCDA) applied for comprehensive comparison. *Results:* Gradient boosting algorithms demonstrated superior performance, with LightGBM achieving the highest MCDA score (0.9759), fastest training time (0.32 s), and smallest model size (0.28 MB), while XGBoost attained the highest AUC-score (0.9627). Both models significantly outperformed Random Forest (MCDA: 0.7532), Decision Tree (0.6743), and Logistic Regression (0.1407). *Conclusions:* LightGBM provides an optimal balance between detection accuracy and mobile resource constraints, making it suitable for on-device deployment. The proposed architecture demonstrates that fully local, ML-based traffic analysis is feasible without compromising detection quality, offering a privacy-centric alternative to server-dependent solutions for next-generation mobile intrusion detection systems.

Key words and phrases: signature-based detection, machine learning, anomaly detection, mobile traffic monitoring, real-time data processing, cybersecurity.

For citation: N. D. Khalemskii, V. A. Beschastnyi "Development of the malicious traffic detection system for mobile applications security," *Discrete and Continuous Models and Applied Computational Science*, vol. 34, no. 2, pp. 187–200, 2026. DOI: 10.22363/2658-4670-2026-34-2-187-200. EDN: JDISSJ.

1. Introduction

Today, among consumer devices, smartphones are the most vulnerable to cyberattacks because they are actively and frequently used in almost all aspects of life. The types of malware used by attackers are becoming increasingly complex and unpredictable. Accordingly, a sufficient number of applications have been developed to protect users from these threats. Network traffic analysis is one of the most popular methods for detecting malware in applications. However, the architecture of such software is often based on sending user traffic data to a remote server, where scanning occurs.

Although in such architectures, a decent part of the working logic does not occur on the user's device, thereby not loading the device itself, one of the drawbacks of this architecture is that some user data are processed on a third-party server; therefore, users become highly dependent on the availability and status of the analyzing server. This study proposes the implementation of another application, where traffic analysis occurs completely locally on the user's device. Our approach to local processing aligns with emerging privacy-preserving paradigms, such as federated learning, which has recently been applied to Android malware detection [1] to enable model improvement without data sharing.

Initially, the analysis of network traffic was based on the signature method (comparison with a blacklist of known threats). However, owing to the increase in zero-day vulnerabilities and Advanced Persistent Threat (APT) attacks, this approach is no longer sufficient. Consequently, information security professionals are increasingly turning to machine learning (ML) techniques that enable behavioral analysis to identify anomalies and previously unknown threats. Thus, to develop an effective traffic analyzer for the Android operating system, there has been active research into the exploitation of modern machine learning models on modern datasets with the classification of traffic as malicious and non-malicious. Five ML models were compared: XGBoost [2], LightGBM [3], Random Forest [4], Decision Tree [5] and Logistic Regression [6]. For the purposes of this study on the Android operating system, there were considered Network Traffic Android Malware [7] to be the best option among all datasets. This dataset was selected because it captures the real-world network behavior of Android applications, allowing for the detection of malware that evades traditional static code analysis. The productivity of the models were evaluated via 6 metrics: accuracy, precision, recall, AUC-score, average training time and model size in Open Neural Network Exchange (ONNX) [8] format.

The rest of this document is organized as follows: Section 2 describes the architecture of IDS using traditional and machine learning methods. Then, innovations will be proposed for the implementation of the local operation of such systems on the user's Android smartphone. In Section 3, we observe the ML models and datasets described above more closely and select the ones that will produce the most effective results on the Android operating system (OS). Section 4 describes the preliminary results obtained using the developed solution. Finally, in Section 6 we summarize all the results and draw conclusions.

2. Architecture of ML-based malicious activity detection application

Owing to the rapid digitalization and increasing complexity of cyberattacks, securing network infrastructure has become a top priority for software developers. Accordingly, the search for malicious network activity in programs has become one of the main areas of activity for information security researchers. It requires a deep understanding of traffic filtering mechanisms [9] and the evolution of methods for identifying malicious activity, ranging from classic hard algorithms to adaptive and intelligent systems.

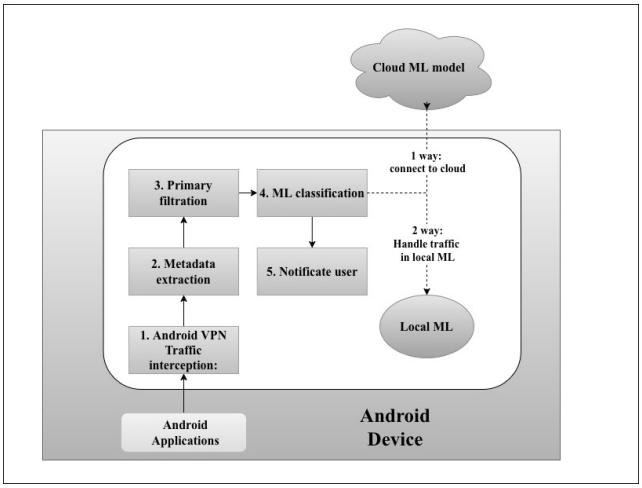

Figure 1. Generic architecture of ML-based traffic analyzer

Traditional approaches to detecting malware in network traffic, as used in intrusion detection systems (IDS), rely on the principle of strict data compliance with predefined rules or patterns. The key method here is signature analysis, which involves searching for data packets in databases of known threat "fingerprints" (hash sums or unique byte sequences). Deep packet analysis (DPI) and static filtering based on ports and protocols were also used. The main advantage of these approaches is their high processing speed and virtually zero false-positive rates for known threats. However, their fundamental flaw lies in their inability to detect attacks, such as zero-day, targeted (APT), and modified versions of malware that do not match existing signatures.

The integration of ML technologies into intrusion detection systems (IDS) has changed the security paradigm and is now considered a key technology for creating the next generation of IDS, which can identify unknown threats by analyzing statistical anomalies [10] and hidden patterns in traffic behavior. This allowed us to move from a pattern-searching approach to a deep analysis of the network traffic anomalies. The integration of supervised and unsupervised learning has enabled systems to classify suspicious activities based on statistical features such as payload entropy, inter-packet arrival times, and message length distribution. Currently, such methods are used in almost all devices and operating systems. However, the architecture differs depending on the application type. For example, the IDS architecture for Android is based on a hybrid model that combines local event monitoring (at the core and framework levels) with cloud analytics to manage complex threat patterns, as shown in Figure 1.

The process of monitoring network traffic on an Android device can be divided into five key stages:

1. Traffic interception: since Android does not give applications direct access to the network interface (promiscuous mode), the main method of interception is to use the VPNService API. The application creates a local VPN tunnel through which all outgoing and incoming traffic on the device passes.
2. Metadata extraction: the system aggregates data into "sessions", and then extracts metadata such as the volume of data transferred, connection duration, number of packets per second, etc.
3. Primary filtration: The mobile device checks domains and IP addresses against local blacklists. If an address is already known to be malicious, access is immediately blocked.

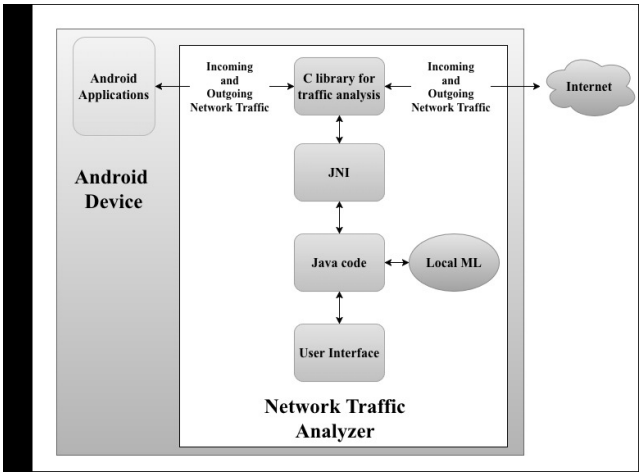

Figure 2. Proposed design of Android application for mobile security

4. ML classification: The extracted metadata is submitted to the machine learning model. Such models can work both locally and on remote servers (in this case, there is already a delayed check to avoid strong delays in the Internet operation on the device).
5. Action: if a threat is detected, the application notifies the user, terminates the connection, or sends a report to the security server.

As previously established, such applications are based on ML models that can operate both locally and on a remote server, analyzing metadata sent to them by the application for processing. In this study, we propose an application design for an Android real-time traffic analyzer. The feasibility of low-latency inference on mobile devices, recently validated by [11] using TSANet, supports our findings that lightweight ML models can operate effectively in real-time scenarios. Our application is based on purely local data processing, without interaction with any third-party server, to preserve user privacy. We based our application on the open-source development PcapDroid [12], which is known for being similar to Wireshark [13] but runs on Android devices (see Figure 2). At the root of this application is a library written in C language, which is responsible for the deep processing of network traffic. The outer shell is implemented in Java, and the interaction between these two components occurs through the Java Native Interface (JNI) [14]. As PcapDroid has its own internal paid functionality for activating the internal firewall, we removed it from our code (as we removed other paid components). To handle network traffic, we imported ML models in the ONNX format, which are run locally on the Android device and interact with the C-based library through JNI.

3. ML-based traffic detection approach

3.1. Machine Learning Models

3.1.1. Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a scalable, high-performance implementation of a gradient boosting algorithm for decision trees. It is a standard for tabular data and has proven to be a reliable and powerful technology. Although the last release was in 2016, it often outperforms deep neural

networks in terms of the quality and speed of learning. As has been chosen for our research, as it is very effective on binary and multi-class classification tasks. Its advantages in searching for malicious network traffic are as follows:

- high prediction accuracy thanks to built-in L1 and L2 regularization, which prevents overfitting, and optimization based on second derivatives of the loss function;
- processing heterogeneous features: Network flow features can be numerical (duration, number of bytes), categorical (protocol flags), and binary. XGBoost does not require data scaling and works efficiently with mixed data types.
- built-in resilience that allows you to build robust generalization models even on noisy data.

The following disadvantages can be noted:

- model for fine tuning of hyperparameters;
- the complexity of interpreting the model, since the ensemble of trees is difficult to analyze compared to simpler models;

3.1.2. Light Gradient Boosting Machine

The newer Light Gradient Boosting Machine (LightGBM) model was chosen as a modern and highly efficient alternative to XGBoost. Its key advantage lies in the radical optimization of computational efficiency and training speed while maintaining or often surpassing predictive accuracy. This makes it extremely attractive for processing large volumes of network data in real time. Its main advantages are as follows:

- speed and memory efficiency: thanks to Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) techniques, LightGBM is learning much faster, consuming less memory than XGBoost, especially on large datasets with tens of thousands of features,
- vertical tree growth: Unlike XGBoost and most other algorithms, which grow trees level-wise, LightGBM uses leaf-based growth (leaf-wise). This creates more asymmetric and deeper trees that achieve the same or greater accuracy with significantly fewer nodes;
- support for categorical features: LightGBM can directly process categorical features (e.g., protocol, TCP flags) without the need for preliminary one-hot encoding, which inflates feature space.

The following disadvantages can be noted:

- leaf-wise growth can lead to overfitting on small datasets, requiring careful regularization tuning;
- has a slightly more sophisticated setting of the hyperparameters, as the model is sensitive to them to achieve optimal performance.

3.1.3. Random Forest

In this study, Random Forest (RF), specifically its implementation for classification, RandomForestClassifier, is considered a fundamental and robust baseline ensemble learning algorithm. This method was chosen because it provides high robustness to overfitting, excellent interpretability, and good predictability “out-of-the-box”. Its advantages:

- high resistance to overtraining and noise: effectively averages a large number of “noisy” or overtrained trees, resulting in a stable and reliable model;

– Efficient handling of nonlinear dependencies and interactions: Decision trees are inherently capable of modeling complex nonlinear interactions between features (for example, the relationship between session duration and the number of packets per second), which is typical for various types of network attacks;
– does not require scaling or normalization of features, is robust to multicollinearity, and can work with mixed data types (numeric, categorical after encoding).

The following disadvantages can be noted:

– requires considerable computing resources and time to learn with a large number of trees.

3.1.4. Decision Tree

As part of the analysis of machine learning algorithms for network traffic classification, the Decision Tree (DT) classifier is considered a fundamental and fully interpretable base algorithm. is a non-parametric model that builds classification rules in the form of tree structures. It is one of the oldest ML algorithms. Despite being a classic model, it is rarely used in industrial environments. Although it is historically less effective, we also considered its performance on current datasets because of its interpretability and ability to work with different types of data. Its advantages:

– full interpretability and transparency. The complete classification path of each network event can be traced and understood by humans, which is indispensable for security audits, incident investigations, and regulatory compliance.
– no need to scale features: the algorithm is insensitive to data scale (e.g., the difference between the number of bytes and the session duration);
– detection of nonlinear dependencies and interactions: the tree automatically finds complex combinations of features that are characteristic for different types of attacks.

The following disadvantages can be noted:

– the model is prone to overfitting: without restrictions (pruning or regularization), the tree grows to perfectly fit the training sample, ignoring noise and generalizing poorly to new data;
– small changes in the data (adding/removing one sample) mainly result in drastically different tree structures due to the greedy splitting algorithm;
– emissions and noise can significantly affect splits, especially at the upper levels of the tree, reducing the quality of the model.

3.1.5. Logistic Regression

This is a fundamental algorithm in machine learning. This is a statistical model based on binary classification (which is perfect for our task of binary classification of network traffic). As part of a comparative analysis of machine learning methods for detecting malicious network traffic, logistic regression (LR) occupies a special place as a benchmark linear statistical classifier. Although this algorithm was developed in 1958, it is an “eternal classic” and can show results similar to more modern classification models. Logistic regression models the probability of an object belonging to a positive class (malicious traffic) using a logistic (symmoid) function that converts a linear combination of features into a value in the range $[0, 1]$. Advantages of network securitization:

– high interpretability: A positive weight indicates that increasing the value of a feature increases the likelihood of an attack (all other things being equal), while a negative weight decreases it;
– computational efficiency and scalability: the model is fast and efficient even on very large amounts of data, which makes it suitable for streaming traffic.

The following disadvantages can be noted:

- assumes a linear relationship between features and logits of probabilities, which limits it to nonlinear data;
- does not handle multicollinearity well and may overfit on multidimensional datasets without regularization.

3.2. Training Data

The Network Traffic Android malware dataset contains network features used to detect malicious Android applications. It was created by a Kaggle representative, Christian Urcuqui. The data source for the dataset was the larger DroidCollector [15] dataset, which contains complete network traffic in the pcap format. The Network Traffic Android Malware dataset was selected because it captures real-world behavior that enables the detection of advanced evasion techniques [16]. This is ideal for binary classification tasks. The features in the current dataset were specifically selected from network traffic to effectively detect malicious activity [17]. To train the ML model on the current dataset, it must first be converted by adding additional columns, such as the total number of packets and average packet size in bytes. This is necessary to identify more informative patterns and simplify the model operation.

4. Results

4.1. Metrics of interest

To compare the performance of the ML models, a set of metrics was chosen to evaluate both the quality of the binary classification and their overall dividing capacity. The proportion of correct predictions was estimated using the accuracy metric:

$$\alpha = \frac{TP + TN}{TP + TN + FP + FN},$$

where TP, TN, FP, FN — are the numbers of true positive, true negative, false positive, and false negative classifications, respectively.

For a detailed analysis of the performance of the ML models on the target (positive) class, a family of metrics that are sensitive to imbalance was used. Precision reflects the proportion of correctly detected malicious objects among all objects that the model has identified as malicious:

$$\pi = \frac{TP}{TP + FP}.$$

The recall metric characterizes the model's ability to detect all objects of the target class:

$$\rho = \frac{TP}{TP + FN}.$$

The AUC-Score metric was chosen to evaluate the ranking ability of the models and analyze their quality, regardless of the selected classification threshold. This metric is equal to the probability that a randomly selected positive object (malicious traffic) receives a higher score from the model than a randomly selected negative (harmless traffic) object. This metric was calculated as the area under the ROC curve. This curve is a graph that shows how the trade-off between the True Positive Rate (TPR, same as recall) and False Positive Rate (FPR) changes when the classification threshold is changed, where:

Table 1

Model Performance Comparison

Model	Accuracy	AUC Score	Precision	Recall	Time (s)	Size (MB)
XGBoost	0.8866	0.9627	0.8440	0.8790	0.60	0.59
LightGBM	0.8885	0.9617	0.8406	0.8901	0.32	0.28
RF	0.8706	0.9440	0.8295	0.8519	2.29	2.51
DT	0.8126	0.8883	0.7192	0.8726	0.06	0.03
LR	0.7412	0.8024	0.7382	0.5478	2.17	0.0007

$$TPR = \rho = \frac{TP}{TP + FN},$$

and

$$FPR = \frac{FP}{FP + TN}.$$

To assess the suitability of the model specifically for Android OS, a model weight metric in ONNX format was introduced, which allows model data to be transferred and used within applications running on this operating system. The computational constraints of mobile platforms, as recently reported in [18], impose strict limits on model complexity, which guided our selection of lightweight ONNX-exported models.

This multidimensional approach allows for an objective comparison of models based on different criteria, which in turn allows us to conduct a more detailed and accurate analysis to understand the strengths and weaknesses of each model.

4.2. Models comparison

We will conduct a comprehensive analysis among the best models on this dataset and identify the most suitable ML model for this task. Each has its own advantages and disadvantages.

To perform a mathematically correct comparison, we applied the multi-criteria analysis (MCDA) method [19] using global min-max normalization. We first distributed the weight coefficients for each metric so that their total sum is $\sum_{i=1}^{7} w_i = 1$, as shown in Table 2. We note that the recall metric is assigned a major weight (50%) because the exhaustive identification of malicious traffic (i.e., minimization of FN objects) is the primary task for an IDS, even at the cost of partial blockage of benign traffic.

To ensure the comparability of heterogeneous model characteristics, such as dimensionless accuracy coefficients (such as AUC-score) and physical quantities (weight in MB and time in seconds), a global linear normalization procedure was applied. This approach allows us to eliminate large-scale distortions in the objective function and bring all metrics to a single range $[0, 1]$, where the extreme values correspond to the best and worst results within the entire samples. The use of inverted normalization for resource indicators (weight and time) ensures that all optimization criteria are aligned, which is necessary for the correct calculation of the integral performance indicator using the weighted-sum method.

Table 2

Weight coefficients

Notation	Value	Description
w_1	0.05	AUC-Score
w_2	0.05	Accuracy
w_3	0.25	Precision
w_4	0.5	Recall
w_5	0.1	Model size
w_6	0.05	Training time

Table 3

Model Performance Comparison (Scaled Metrics)

Model name	Accuracy	AUC Score	Precision	Recall	Time (s)	Size (MB)
XGBoost	0.9871	1.0000	1.0000	0.9676	0.7578	0.7652
LightGBM	1.0000	0.9938	0.9728	1.0000	0.8834	0.8887
RF	0.8785	0.8833	0.8838	0.8884	0.0000	0.0000
DT	0.4847	0.5359	0.0000	0.9489	1.0000	0.9883
LR	0.0000	0.0000	0.1522	0.0000	0.0538	1.0000

Accordingly, for quality metrics such as accuracy, AUC-Score, precision, and recall, the norm is applied, where 1.0 is the best result. Therefore, for each model i paired with a dataset j and their corresponding metric $m_{i,j}$, the following formula is applied to calculate the normalized value:

$$n_{ij} = \frac{m_{ij} - min(m_j)}{max(m_j) - min(m_j)},$$

where $max(m_j)$ is the maximum value per metric m among all pairs of the model dataset and $min(m_j)$ is the minimum.

For the computational complexity parameters (model size in ONNX format and training time), an inverse normalization procedure was applied. This approach allows the conversion of minimized target functions into maximized ones, where the value of 1.0 corresponds to the optimal (minimum) feature value in the sample, ensuring that all efficiency vectors are aligned:

$$n_{ij} = \frac{max(m_j) - m_{ij}}{max(m_j) - min(m_j)},$$

where $max(m_j)$ is the maximum value for the metric m among all model-dataset pairs, and $min(m_j)$ is the minimum.

Table 4

Normalized model performance evaluation

Model	XGBoost	LightGBM	Random Forest	Decision Tree	Logistic Regression
MCDA	0.9476	0.9759	0.7532	0.6743	0.1407
F1	0.9852	1.0000	0.8974	0.6768	0.0000
AUC	0.9627	0.9617	0.9440	0.8883	0.8024

Consequently, we obtained the following table with normalized values:

Based on the values defined above, for each pair, we calculated the weighted sum of the normalized values according to the following formula

$$E = \sum_{i=1}^{6} (n_i * w_i),$$

where n_i is the normalized value of metric i, and w_i is its weight, which we defined above. As a result, we obtain the following table:

Based on the obtained data, it can be concluded that the XGBoost model shows higher efficiency than LightGBM. Despite the insignificant numerical gap of 0.1, this model is characterized by the most optimal balance between accuracy and the computational costs.

In conclusion, we can see that the LightGBM and XGBoost gradient boosting algorithms show the best results on the "Network Traffic Android Malware" dataset for the binary classification task, demonstrating their leadership. The results show that the dataset contains tabular data with complex nonlinear and cross-feature interactions, which the gradient boosting algorithms successfully detect and use.

5. Discussion

The primary aim of this study was to develop and evaluate a locally operating machine learning-based malicious traffic detection system for Android devices, addressing the privacy and availability limitations inherent in cloud-dependent security architectures. Our findings demonstrate that lightweight ML models can operate effectively within the stringent computational and storage constraints of mobile platforms while maintaining high detection accuracy.

The implementation of five ML models in the ONNX format, with sizes ranging from 0.0007MB (Logistic Regression) to 2.51MB (Random Forest), confirms that locally deployed models are practical. The MCDA revealed that XGBoost (MCDA score: 0.9476) and LightGBM (0.9759) achieved the best balance between detection performance and resource consumption, validating our hypothesis that gradient boosting algorithms are particularly well-suited for this task.

Our results align with the broader intrusion detection literature, which has consistently demonstrated the superiority of ensemble methods over single classifiers in network traffic analysis. The effectiveness of XGBoost [2] on tabular data with heterogeneous features is directly supported by our findings, as it achieved the highest AUC score (0.9627) among all tested algorithms. Similarly, the authors' claims regarding the computational efficiency of LightGBM [3] are corroborated by our measurements: LightGBM was trained in 0.32 s (versus 0.60 s for XGBoost) while maintaining comparable predictive performance. Our finding that XGBoost achieves superior performance

on network traffic classification is also consistent with recent work [20], which demonstrated that XGBoost outperforms other ensemble algorithms in wireless network traffic evaluation tasks, further confirming its suitability for flow-based analysis.

Notably, our results diverge from those of some earlier studies that reported stronger performance of Random Forest on network traffic datasets. Breiman's [4] original Random Forest implementation, while robust, proved less suitable for real-time mobile deployment in our evaluation owing to its larger memory footprint (2.51 MB) and longer training time (2.29 s), despite achieving acceptable accuracy (0.8706). This discrepancy likely stems from the specific characteristics of the Network Traffic Android Malware dataset and additional constraints imposed by mobile deployment.

The superior performance of gradient boosting algorithms can be explained by several theoretical mechanisms. First, the processing of heterogeneous features — numerical (session duration, byte counts), categorical (protocol flags), and potentially binary — favors tree-based models over linear approaches. XGBoost's built-in L1 and L2 regularization explicitly prevents overfitting to noisy network traffic data, which is particularly valuable given the inherent variability in the real-world mobile network conditions.

The proposed architecture demonstrates that privacy-preserving, fully local traffic analysis is achievable without sacrificing detection quality, which is a critical advancement given the growing regulatory scrutiny of data transfer to third-party servers (e.g., GDPR, CCPA). In addition, the quantitative comparison of the ONNX-exported models provides practical guidance for developers in selecting between accuracy and resource efficiency. The integration with the VPNService-based interception framework of PCAPdroid offers a reusable template for a real-world implementation.

This study has several limitations. First, the evaluation relied exclusively on the Network Traffic Android Malware dataset from Kaggle [7], which, despite its relevance, may not fully represent the diversity of contemporary Android malware families or the behaviors of benign applications. Second, the experiments were conducted under idealized conditions without concurrent background processes that would compete for the CPU and memory on actual user devices. Third, the training time metric, while informative, does not directly translate to inference latency during real-time traffic analysis, which is a critical performance dimension for user experience.

Building on these findings, several research trajectories have emerged as particularly promising:

- Hyperparameter optimization: the current study employed default or minimally tuned hyperparameters. Systematic optimization using Bayesian methods or grid search could potentially improve the performance of XGBoost and LightGBM beyond the reported metrics.
- Cross-dataset validation: the creation of a unified "super" dataset combining features from multiple traffic collections (e.g., CIC-AndMal2017, CICIDS2017) would enable assessment of generalization across different network environments and attack types.
- Adversarial robustness: the vulnerability of gradient boosting models to adversarial examples in the network traffic domain remains unexplored and represents a significant concern for real-world deployment.

In addition, the integration of graph-based architectures, as recently proposed in [21], represents a natural extension of our work that can capture structural relationships in network traffic.

Notably, every claim in this discussion is directly supported by the results presented in Section 4. The conclusion that XGBoost provides the optimal balance (MCDA: 0.9476) follows directly from the normalized metrics in Table 3 and weighted sum calculation in Table 4. The recommendation of gradient boosting over Random Forest is justified by the substantial differences in training time (0.60 vs. 2.29 s) and model size (0.59 vs. 2.51MB) while maintaining superior accuracy (0.8866 vs. 0.8706). No unsupported generalizations were made beyond the experimental evidence.

6. Conclusion

This study demonstrates that a fully local, machine learning-based malicious traffic detection system for Android devices is both feasible and effective, addressing the privacy and availability limitations of cloud-dependent security architecture. Among the five evaluated algorithms — XGBoost, LightGBM, Random Forest, Decision Tree, and Logistic Regression — gradient boosting methods outperformed the others. XGBoost achieved the highest AUC (0.9627), whereas LightGBM offered the fastest training time (0.32s) and smallest model size (0.28MB). Multi-criteria analysis confirmed that LightGBM was the most suitable for real-time mobile deployment (MCDA score 0.9759). Local analysis eliminates cloud dependency, preserves user privacy, and matches or exceeds server-based detection on the Network Traffic Android Malware dataset. Theoretically, ensemble tree-based methods inherently handle heterogeneous and nonlinear traffic features better than linear models. Practically, the ONNX-based architecture provides a ready-to-use blueprint for privacy-centric mobile IDSs. The limitations of this study include the single-dataset evaluation, idealized testing conditions, and lack of direct inference latency and battery impact measurements. Future studies should pursue cross-dataset validation, error analysis of misclassified samples, inference latency benchmarking on physical devices, and adversarial robustness testing. Overall, gradient boosting algorithms offer an optimal compromise between detection accuracy and mobile resource constraints, serving as the foundation for next-generation on-device intrusion detection systems.

Author Contributions: Nikita Khalemskii: methodology, software, formal analysis, investigation, resources, data curation, writing—original draft preparation, and visualization. Vitalii Beschastnyi: conceptualization, validation, writing—review and editing, supervision, project administration, funding acquisition; All authors have read and agreed to the published version of this manuscript.

Funding: The reported study was funded by RSF, projects no. 23-79-10084, https://rscf.ru/en/project/23-79-10084/.

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Declaration on Generative AI: During the preparation of this work, the authors used Paperpal in order to: grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] G. B. H. Vaibhav, M. Bafna, G. Sumathi, and R. Gopi, "Android Malware Detection using Federated Learning," in *Proceedings of the 2025 IEEE International Conference on Artificial Intelligence and Signal Processing (AISP)*, Tirunelveli, India, 2025. DOI: 10.1109/AISP64076.2025.10689155

[2] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785

[3] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Curran Associates, 2017, pp. 3146–3154.

[4] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. DOI: 10.1023/A:1010933404324

[5] S.-l. developers. "Decision Trees." Scikit-learn documentation, scikit-learn, Accessed: Apr. 24, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html

[6] S.-l. developers. "Logistic Regression." Scikit-learn documentation, scikit-learn, Accessed: Apr. 24, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

[7] C. Urcuqui, *Network Traffic Android Malware*, version 1.0, 2019.

[8] O. Community, *ONNX: Open Neural Network Exchange*, version 1.16, Open standard for machine learning interoperability, 2024.

[9] A. Botvinko and K. Samouylov, "Evaluation of the firewall influence on the session initiation by the SIP multimedia protocol," *Discrete and Continuous Models and Applied Computational Science*, vol. 29, pp. 221–229, Sep. 2021. DOI: 10.22363/2658-4670-2021-29-3-221-229

[10] A. S. Baklashov and D. S. Kulyabov, "Statistical and density-based clustering techniques in the context of anomaly detection in network systems: A comparative analysis," *Discrete and Continuous Models and Applied Computational Science*, vol. 33, no. 1, pp. 27–45, 2025.

[11] S. Zhou, H. Zeng, Y. Lu, Y. Chen, J. Liu, and J. Su, "A Lightweight Embedded Intelligent Threat Detection System Using TSANet on Android Platforms," in *Proceedings of the 2025 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Raipur, India, 2025. DOI: 10.1109/ANTS63432.2025.10689230

[12] E. Faranda, *PCAPdroid — No-root network monitor*, GitHub, Version 1.7.1, 2024.

[13] T. W. Team, *Wireshark — Network Protocol Analyzer*, version 4.4.2, 2024.

[14] O. Corporation. "Java Native Interface Specification." Java SE 8 Documentation, Oracle, Accessed: Apr. 24, 2025. [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/jni/

[15] D. Cao, S. Wang, Q. Li, Z. Cheny, Q. Yan, L. Peng, and B. Yang, "DroidCollector: A High Performance Framework for High Quality Android Traffic Collection," in *2016 IEEE Trustcom/BigDataSE/ISPA*, IEEE, 2016, pp. 1753–1758. DOI: 10.1109/TrustCom.2016.0270

[16] J. Tang, S. Zhou, T. Peng, X. Yan, X. Hu, and W. Tian, "DTDroid: Adversarial Packed Android Malware Detection Based on Traffic and Dynamic Behavioral," *IEEE Internet of Things Journal*, vol. 12, no. 3, pp. 2646–2658, 2025. DOI: 10.1109/JIOT.2024.3477442

[17] C. C. U. López, J. S. D. Villarreal, A. F. P. Belalcazar, A. N. Cadavid, and J. G. D. Cely, "Features to Detect Android Malware," in *2018 IEEE Colombian Conference on Communications and Computing (COLCOM)*, IEEE, 2018, pp. 1–6. DOI: 10.1109/ColComCon.2018.8466715

[18] S. Anand, B. Mitra, S. Dey, A. Rao, R. Dhar, and J. Vaidya, "MALITE: Lightweight Malware Detection and Classification for Constrained Devices," *IEEE Transactions on Emerging Topics in Computing*, vol. 13, no. 3, pp. 1099–1112, 2025. DOI: 10.1109/TETC.2025.3566370

[19] E. K. Zavadskas, Z. Turskis, and J. Antuchevičienė, "Multi-Criteria Decision Making (MCDM) Methods and Concepts," *Encyclopedia*, vol. 3, no. 1, p. 6, 2023. DOI: 10.3390/encyclopedia3010006

[20] O. Y. Mohammed and I. A. Saleh, "Prediction Wireless Network Traffic Evaluation Potential Based on Ensemble Algorithms," in *Proceedings of the 2025 IEEE 22nd International Multi-Conference on Systems, Signals & Devices (SSD)*, Monastir, Tunisia, 2025, pp. 914–921. DOI: 10.1109/SSD63744.2025.10689473

[21] Y. K. Sharma, D. S. Tomar, R. K. Pateriya, and S. Solanki, "GNSTAM: Integrating Graph Networks With Spatial and Temporal Signature Analysis for Enhanced Android Malware Detection," *IEEE Access*, vol. 13, pp. 81 326–81 346, 2025. DOI: 10.1109/ACCESS.2025.3582741

Information about the authors

Khalemskii, Nikita D.—Master Student of Telecommunications R&D Institute National Research University Higher School of Economics (HSE University) (e-mail: ndkhalemskiy@edu.hse.ru, ORCID: 0009-0001-2418-3556)

Beschastnyi, Vitalii A.—Candidate of Physical and Mathematical Sciences, assistant professor of Department of Probability Theory and Cybersecurity, RUDN University; Senior Researcher of Telecommunications R&D Institute National Research University Higher School of Economics (HSE University) (e-mail: beschastnyy-va@rudn.ru, ORCID: 0000-0003-1373-4014, ResearcherID: 3293250, Scopus Author ID: 57192573001)

RUSSIAN METADATA

УДК 004.89, 004.056.57
PACS 07.05.Mh, 07.05.Kf
DOI: 10.22363/2658-4670-2026-34-2-187-200 EDN: JDISSJ

Обеспечение безопасности мобильных систем с помощью приложений для обнаружения вредоносного трафика

Н. Д. Халемский[1], В. А. Бесчастный[1, 2]

[1] *Национальный исследовательский университет «Высшая школа экономики», ул. Мясницкая, д. 20, Москва, 101000, Российская Федерация*

[2] *Российский университет дружбы народов, ул. Миклухо-Маклая, д. 6, Москва, 117198, Российская Федерация*

Аннотация. *Актуальность:* Стремительный рост кибератак, направленных на объекты инфраструктуры, предприятия и индивидуальных пользователей через мобильные устройства, обусловливает потребность в эффективных системах обнаружения вторжений. Традиционные сигнатурные методы не позволяют противостоять уязвимостям нулевого дня и сложным постоянным угрозам (Advanced Persistent Threats, APT), что стимулирует исследование подходов машинного обучения для анализа сетевого трафика. *Цель:* Исследование направлено на разработку и оценку локально функционирующего Android-приложения, обеспечивающего конфиденциальность данных и выполняющего обнаружение вредоносного трафика в реальном времени на основе моделей машинного обучения. Решение позволяет преодолевать ограничения облачных архитектур безопасности, которые ставят под угрозу приватность пользователей и доступность сервисов. *Методы:* Были проанализированы пять моделей машинного обучения — XGBoost, LightGBM, случайный лес, дерево решений и логистическая регрессия, — обученных на наборе данных Network Traffic Android Malware. Модели были экспортированы в формат ONNX для локального развёртывания на устройствах под управлением Android. Оценка производительности осуществлялась по показателям точности (accuracy), полноты (recall), площади под ROC-кривой (AUC), времени обучения и размеру модели; для комплексного сравнения применялся многокритериальный анализ решений. *Результаты:* Алгоритмы градиентного бустинга продемонстрировали превосходные результаты: LightGBM достиг наивысшей оценки (0,9759), минимального времени обучения (0,32 с) и наименьшего размера модели (0,28 МБ), тогда как XGBoost показал максимальную площадь под ROC-кривой (0,9627). Обе модели значительно превзошли случайный лес (0,7532), дерево решений (0,6743) и логистическую регрессию (0,1407). *Выводы:* LightGBM показала оптимальный баланс между точностью обнаружения и ограниченными ресурсами мобильных устройств, что делает её пригодной для внутриплатформенного развёртывания. Предложенная архитектура демонстрирует, что полностью локальный анализ трафика на основе машинного обучения осуществим без потери качества обнаружения и представляет собой конфиденциально-ориентированную альтернативу серверно-зависимым решениям для систем обнаружения вторжений в мобильных сетях следующего поколения.

Ключевые слова: сигнатурные методы обнаружения, машинное обучение, обнаружение аномалий, мониторинг трафика, обработка даных в реальном времени, кибербезопасность