Science and Technology Publishing (SCI & TECH)
ISSN: 2632-1017
Vol. 9 Issue 12, December - 2025

# Analysis Of Android Malware Detection Via Network Traffic Static Behavioral Profiling Using Decision Tree Binary Classifier And Cicmaldroid2020 Dataset

**Precious D. Agburuga**[1]
Department OF Electrical and Electronic Engineering
Federal University Otuoke, Bayelsa State, Nigeria
agburugapd@fuotuoke.edu.ng

**Godwill Ajumo Samuel**[2]
Department of Computer Engineering Technology
Captain Elechi Amadi Polytechnic, Rumuola
Port Harcourt Rivers State
godwill.samuel@portharcourtpoly.edu.ng

**AGUIYI Nduka Watson**[3]
Department OF Electrical and Electronic Engineering
Federal University Otuoke, Bayelsa State, Nigeria
aguiyiwatson@gmail.com; aguiyinw@fuotuoke.edu.ng

***Abstract*—Rapid proliferation of sophisticated Android malware necessitates the development of advanced detection frameworks that move beyond traditional signature-based scanning. This research paper presents a robust methodology for binary malware classification by integrating static structural profiling with network traffic behavioral analysis. The system utilizes the CICMalDroid2020 dataset, which provides a diverse collection of over 17,000 samples including Adware, Banking malware, and SMS threats. By extracting both application permissions and statistical network flow features, the research captures a holistic digital footprint of malicious intent. The core of the detection engine relies on a Decision Tree binary classifier to categorize applications as either benign or malicious. Experimental results demonstrate that the model achieves an impressive accuracy of 96.4% and a recall of 97.1%, proving highly effective at identifying threats that attempt to mask their behavior behind legitimate permissions. Analysis of the decision logic reveals that network metrics such as Inter-Arrival Time (IAT) and packet length distribution are critical indicators of command-and-control communication. This study underscores the importance of hybrid analysis in providing transparent, rule-based security solutions for the Android ecosystem.**

***Keywords—Android Malware Detection; Static Analysis; Network Behavioral Profiling; Decision Tree Classifier; CICMalDroid2020 Dataset; Binary Classification; Mobile Security; Machine Learning; Feature Extraction; Traffic Analysis.***

## 1. Introduction

The global mobile landscape has witnessed an unprecedented expansion of the Android operating system, which currently powers billions of active devices worldwide [1,2]. This ubiquity makes Android a primary target for cybercriminals who develop increasingly sophisticated malicious software to compromise user privacy and financial security [3,4]. Recent statistics indicate that mobile malware variants have grown by over 30% annually, with attackers employing complex delivery mechanisms to bypass traditional security filters [5]. These threats range from silent spyware that monitors user activity to aggressive banking Trojans designed to intercept one-time passwords and drain financial accounts [6].

Traditional defense mechanisms primarily rely on signature-based detection, which compares the file hash of an application against a database of known threats [7]. While this method remains effective for identified malware, it fails to recognize zero-day attacks or polymorphic threats that alter their code structure to evade detection. Static analysis offers a more proactive approach by inspecting an application's metadata and permissions without execution, yet it remains limited when malware authors use code obfuscation or dynamic loading [8,9]. As malicious applications become better at mimicking the structural appearance of legitimate tools, the need for a deeper analysis of an application's functional intent becomes paramount.

Again, network behavioral profiling has emerged as a critical secondary layer of defense by examining how an application communicates with external servers [10]. Every malicious action, such as data

exfiltration or receiving commands from a remote host, leaves a distinct statistical trace in the network traffic [11,12]. By analyzing metrics like packet size, flow duration, and communication frequency, security systems can identify anomalies that static checks might overlook. The CICMalDroid2020 dataset provides a comprehensive foundation for this research, offering detailed network logs and static features across diverse malware families to facilitate the development of robust detection models [13,14].

Accordingly, this study addresses the growing complexity of the mobile threat landscape by combining static structural data with network communication patterns. Utilizing a Decision Tree binary classifier allows for the creation of an interpretable model that maps these hybrid features to a definitive security status. The primary goal of this research is to provide a transparent and efficient framework that balances the speed of static inspection with the predictive accuracy of behavioral analysis. By establishing clear rules for classification, the proposed methodology assists security analysts in understanding the specific triggers that define an application as malicious in a live environment.

## 2. Methodology

This work employs a structured approach to identifying malicious Android applications by integrating static structural analysis with network communication patterns. This systematic workflow ensures that the final classification model benefits from both the inherent properties of the APK (Android Package Kit) files and the predictive nature of their expected network behavior. By utilizing a high-fidelity dataset and a supervised learning framework, the study establishes a robust pipeline for binary malware classification.

### 2.1. Dataset Selection and Acquisition

The foundation of this research rests on the use of the CICMalDroid 2020 dataset, chosen for its comprehensive nature containing over 17,000 Android samples [15]. By incorporating five distinct categories, Adware, Banking malware, SMS malware, Riskware, and Benign applications, this dataset provides a broad spectrum of malware behaviors essential for robust analysis.

Data acquisition involves downloading the raw samples and associated network logs from the Canadian Institute for Cybersecurity repository. The integrity of the files are established by verifying the checksums and ensuring that the benign samples represent a wide variety of legitimate categories such as tools, productivity, and social media. This diversity prevents the model from developing a bias toward specific types of functional software. The acquisition process also includes a preliminary audit of the dataset to remove any corrupted files or incomplete logs that might skew the statistical distribution. The CICMalDroid 2020 dataset provides a comprehensive collection of Android samples for binary classification, totaling 17,341 applications, with 8,412 malicious and 8,929 benign samples, as shown in Table 1.

**Table 1 The CICMalDroid 2020 dataset class distribution**

| Binary Class | Instance Count |
|---|---|
| **Benign** | 8,929 |
| **Malware** | 8,412 |
| **Total** | **17,341** |

The final composition of the training and testing sets relies on a balanced sampling strategy to ensure the binary classifier performs effectively. Researchers partition the CICMalDroid2020 data into two primary labels: Malicious and Benign. Adware, Banking, SMS, and Riskware categories are merged into a single "Malicious" class to facilitate binary classification. This consolidation creates a clear boundary for the Decision Tree algorithm to learn the distinguishing features between harmful and safe software.

### 2.2. Feature Extraction (Static Profiling)

Feature extraction focuses on uncovering the structural and functional characteristics of Android applications without executing the code. The process begins with the decompression of the APK files using tools like APKTool to access the AndroidManifest.xml and classes.dex files. These files contain essential metadata regarding the application's intent and its requested capabilities. Static profiling targets specific attributes such as requested permissions, hardware components, and declared services that often hint at suspicious motives. The categorized primary network traffic features extracted from the CICMalDroid2020 dataset are presented in Table 2. These features are derived from the raw PCAP files and represent the statistical properties of the network flows used to train the Decision Tree classifier.

**Table 2 The network feature categories and descriptions**

| Feature Category | Specific Features | Description |
|---|---|---|
| Flow Statistics | Flow Duration, Flow Bytes/s, Flow Packets/s | Measures the total time and throughput of a communication session between the device and a remote host. |
| Packet Lengths | Max/Min Packet Length, Mean Packet Length, Packet Length Std | Quantifies the size distribution of data packets, which often distinguishes automated malware heartbeats from human interaction. |
| Inter-Arrival Time (IAT) | Flow IAT Mean, Flow IAT Max, Flow IAT Std | Analyzes the time intervals between successive packets to detect periodic signaling common in Command and Control (C2) communication. |
| Flag Counts | SYN Flag Count, PSH Flag Count, ACK Flag Count | Tracks the TCP control flags used, identifying scanning behavior or specific synchronization patterns used by malicious payloads. |
| Bulk Metrics | Forward/Backward Bulk Rate, Avg Bulk Size | Calculates the efficiency of data transfer in both directions, often highlighting large exfiltration events. |
| Subflow Features | Subflow Fwd Packets, Subflow Bwd Bytes | Breaks down the main flow into smaller components to observe the behavior of specific segments of the network session. |
| | | |

The extraction phase specifically targets API calls and permission strings that correlate strongly with malicious activity. For example, the combination of "SEND_SMS" and "RECEIVE_BOOT_COMPLETED" frequently appears in SMS-based malware but rarely in basic utility apps. This research maps these strings into a numerical vector format suitable for machine learning processing. Each application is represented by a binary or frequency-based feature vector that captures the presence or absence of critical security-sensitive functions.

Refining the feature set involves a dimensionality reduction step to remove redundant or non-informative attributes. Features that appear in nearly all applications or those that are unique to only a single sample are discarded to prevent overfitting. The resulting static profile provides a snapshot of what the application is capable of doing before it ever connects to a network. This profile acts as the first half of the input data for the Decision Tree classifier.

### 2.3. Network Behavioral Profiling

Network behavioral profiling supplements static analysis by examining the communication patterns inherent in the CICMalDroid2020 dataset. This step involves analyzing PCAP files to extract flow-based features such as packet duration, byte counts, and inter-arrival times. The research focuses on how an application interacts with remote servers, as malware often exhibits unique "heartbeat" patterns or distinct data exfiltration behaviors. By characterizing these flows, the methodology captures the dynamic intent of the code through a static lens of recorded logs.

The profiling stage identifies key network metrics including the number of outgoing connections, the use of non-standard ports, and the ratio of uploaded to downloaded data. Malicious entities frequently communicate with Command and Control servers, resulting in specific statistical signatures in the traffic logs. These signatures are translated into quantitative features that describe the "behavioral footprint" of the application. This approach bridges the gap between what the code says it does and how it actually interacts with the internet.

Integration of network features with the previously extracted static features creates a hybrid profile for every sample. This holistic view ensures that an application which might look "clean" during static permission checks is flagged if its network behavior aligns with known malware archetons. The final feature vector for each sample in the dataset becomes a unified representation of both structural identity and communication tendencies.

### 2.4. Model Training (Decision Tree Classifier)

The model training phase utilizes a Decision Tree binary classifier to learn the complex relationships between the hybrid features and the application labels. Decision Trees are chosen for their transparency and ability to handle both numerical and categorical data with minimal preprocessing. The algorithm works by recursively partitioning the dataset into subsets based on the most informative features. This creates a flowchart-like structure where internal nodes represent feature tests and leaf nodes represent the final "Malicious" or "Benign" classification.

Hyperparameter tuning plays a critical role in optimizing the performance of the classifier. The research employs techniques such as cost-complexity pruning to prevent the tree from becoming overly deep and memorizing the noise in the CICMalDroid2020 dataset. Parameters like the minimum samples per leaf and the maximum depth are adjusted using grid search and cross-validation. This ensures that the model generalizes well to new, unseen Android

applications rather than just performing well on the training data.

The training process involves feeding the preprocessed feature vectors into the Scikit-Learn implementation of the CART (Classification and Regression Trees) algorithm. The Gini impurity index serves as the primary criterion for splitting nodes, as it effectively measures the probability of a random sample being misclassified. Upon completion of the training cycles, the model produces a set of logic-based rules that can be easily audited by security analysts to understand why a specific app was flagged as a threat.

### 2.5. Evaluation and Validation

Evaluation of the trained model relies on a rigorous testing protocol using a hold-out portion of the dataset. Metrics such as Accuracy, Precision, Recall, and the F1-Score provide a multi-dimensional view of the classifier's effectiveness. Accuracy measures the overall percentage of correct predictions, while Precision and Recall help identify how well the model handles false positives and false negatives. A high Recall is particularly vital in malware detection to ensure that very few malicious threats go undetected.

Validation of the results involves the creation of a Confusion Matrix to visualize the performance across the two binary classes. This matrix highlights whether the model struggles more with misclassifying benign apps as malware or vice versa. Additionally, the Area Under the Receiver Operating Characteristic (ROC) curve is calculated to determine the model's ability to distinguish between classes at various threshold settings. An AUC value close to 1.0 indicates a highly superior classification performance.

The final step of the methodology includes a k-fold cross-validation procedure to ensure the stability of the findings. By dividing the dataset into "k" different subsets and rotating the training and testing roles, the research minimizes the impact of data variability. This comprehensive validation suite confirms that the combination of static profiling and network behavioral analysis provides a reliable framework for Android malware detection.

## 3. Results and discussion

The experimental results demonstrate that the combination of static architectural features and network behavioral profiling provides a highly discriminative foundation for Android malware detection. Utilizing the Decision Tree binary classifier on the CICMalDroid2020 dataset, the model achieved an overall accuracy of 96.4%, successfully distinguishing between benign applications and various malicious families. The integration of network flow metrics, such as Flow IAT and Packet Length Variance, proved particularly effective in flagging SMS malware and Banking Trojans that frequently communicate with remote command-and-control servers.

Table 3 Comparison of Classification Performance

| Metric | Decision Tree Performance |
|---|---|
| **Accuracy** | 96.40% |
| **Precision** | 95.80% |
| **Recall (Sensitivity)** | 97.10% |
| **F1-Score** | 96.40% |
| **False Positive Rate** | 3.20% |

Analysis of the Decision Tree's internal nodes reveals that specific permission combinations and network "heartbeat" patterns serve as the most significant split criteria. Features like READ_SMS paired with a high frequency of PSH flags consistently led to malicious classifications, reflecting the underlying data exfiltration behavior of many samples. The model maintained a robust Precision of 95.8% and a Recall of 97.1%, indicating a low rate of false negatives which is critical for maintaining mobile security. These metrics suggest that the hybrid profiling approach captures essential indicators that either method alone might overlook.

The discussion of these findings highlights the transparency of the Decision Tree algorithm as a major advantage for security practitioners. Unlike black-box deep learning models, the generated tree provides a clear, rule-based logic that explains why a specific application was deemed a threat. For instance, the model often isolated Adware based on its characteristic high volume of small-sized packets and frequent HTTP connections to known advertising domains. This interpretability allows developers and security auditors to verify the model's decision-making process against known cybersecurity frameworks and mobile threat landscapes.

Despite the high performance, certain challenges were observed in the classification of Riskware samples, which often mimic the network behavior of legitimate utility tools. Some benign applications that require extensive permissions, such as navigation or communication apps, occasionally triggered false positives due to their complex network profiles. Future iterations of this research could explore the use of ensemble methods like Random Forest to further reduce these errors by aggregating multiple decision paths. Nevertheless, the current results confirm that network behavioral profiling is a potent supplement to traditional static analysis in the evolving fight against Android malware.

## 4. Conclusion

This research successfully demonstrates that integrating static structural analysis with network behavioral profiling significantly enhances the detection of Android malware. By utilizing the CICMalDroid2020 dataset, the study confirms that a

Decision Tree binary classifier can achieve a high accuracy of 96.4% while maintaining clear, interpretable logic. The findings suggest that relying solely on static permissions is insufficient in a landscape where malware frequently hides its intent until execution. The inclusion of network metrics like Flow IAT and Packet Length serves as a vital secondary layer of verification.

The methodology proves that a supervised learning approach using the CART algorithm effectively maps the "behavioral footprint" of malicious applications. This research highlights how specific communication patterns, such as those found in Banking and SMS malware, provide distinct statistical signatures that traditional signature-based scanners might miss. Furthermore, the transparency of the Decision Tree allows security analysts to derive actionable rules for firewall configurations and endpoint protection. This balance of high performance and explainability makes the proposed framework suitable for real-world mobile security applications.

Final analysis of the results indicates that the hybrid feature set creates a robust defense against common Android threats. While some overlap exists between benign utility apps and certain classes of Riskware, the model provides a reliable baseline for automated threat assessment. This study reinforces the necessity of holistic analysis in mobile forensics, where both the code's declaration and its actual network interaction are scrutinized. The successful validation of this model paves the way for more responsive and lightweight security tools on the Android platform.

## Future Work

**Future iterations of this research could focus on the implementation of ensemble learning** techniques to further refine the classification of edge cases. Algorithms such as Random Forest or Gradient Boosting Machines may reduce the variance observed in the current Decision Tree model, potentially lowering the False Positive Rate for complex Riskware. Additionally, expanding the feature set to include dynamic API hooking results could provide a more granular view of how an application behaves in a runtime environment. This triple-layered approach would likely increase the model's resilience against obfuscated malware.

Another promising direction involves the exploration of real-time traffic analysis on mobile devices with limited computational resources. Optimizing the feature extraction process to run as a lightweight background service could allow for "on-the-fly" detection without significantly impacting battery life. Researchers might also investigate the impact of encrypted traffic on behavioral profiling, as the increasing use of HTTPS and TLS poses challenges for deep packet inspection. Developing methods to classify encrypted flows based solely on metadata remains a critical frontier in cybersecurity.

Integrating the proposed model into a cloud-based collaborative defense system represents a final logical step for this work. By aggregating anonymized network profiles from a vast array of devices, a central intelligence unit could update the Decision Tree rules dynamically as new threats emerge. This would transform the current static model into an adaptive security framework capable of responding to zero-day vulnerabilities in the Android ecosystem. Such a system would leverage the collective behavioral data of millions of users to stay ahead of increasingly sophisticated malware authors.

## References

1. Zarif, A. (2024). Securing the Future of Mobility: Understanding the Security Perspectives of Cybersecurity, Operating Systems (OS) Security, Mobile Computing. *Journal of Computer Science and Engineering*.
2. Rathod, H., & Agal, S. (2023, August). A study and overview on current trends and technology in mobile applications and its development. In *International conference on ICT for sustainable development* (pp. 383-395). Singapore: Springer Nature Singapore.
3. Zarif, A. (2024). Securing the Future of Mobility: Understanding the Security Perspectives of Cybersecurity, Operating Systems (OS) Security, Mobile Computing. *Journal of Computer Science and Engineering*.
4. Ahvanooey, M. T., Li, Q., Rabbani, M., & Rajput, A. R. (2020). A survey on smartphones security: software vulnerabilities, malware, and attacks. *arXiv preprint arXiv:2001.09406*.
5. Ferdous, J., Islam, R., Mahboubi, A., & Islam, M. Z. (2023). A review of state-of-the-art malware attack trends and defense mechanisms. *IEEe Access*, *11*, 121118-121141.
6. Caruso, A. (2024). *Forensic Analysis of Mobile Spyware: Investigating Security, Vulnerabilities, and Detection Challenges in Android and iOS Platforms* (Doctoral dissertation, Politecnico di Torino).
7. Ferdous, J., Islam, R., Mahboubi, A., & Islam, M. Z. (2023). A review of state-of-the-art malware attack trends and defense mechanisms. *IEEe Access*, *11*, 121118-121141.
8. Ngo, Q. D., Nguyen, H. T., Le, V. H., & Nguyen, D. H. (2020). A survey of IoT malware and detection methods based on static features. *ICT express*, *6*(4), 280-286.
9. Vasan, D., Alazab, M., Venkatraman, S., Akram, J., & Qin, Z. (2020). MTHAEL: Cross-architecture IoT malware detection based on neural network advanced ensemble learning. *IEEE Transactions on Computers*, *69*(11), 1654-1667.
10. Krishnan, P., Jain, K., Buyya, R., Vijayakumar, P., Nayyar, A., Bilal, M., & Song, H. (2021). MUD-based behavioral profiling security framework for software-defined IoT networks. *IEEE Internet of Things Journal*, *9*(9), 6611-6622.

11. Vaccari, I., Narteni, S., Aiello, M., Mongelli, M., & Cambiaso, E. (2021). Exploiting Internet of Things protocols for malicious data exfiltration activities. *IEEE Access*, *9*, 104261-104280.

12. Sabir, B., Ullah, F., Babar, M. A., & Gaire, R. (2021). Machine learning for detecting data exfiltration: A review. *ACM Computing Surveys (CSUR)*, *54*(3), 1-47.

13. Ullah, F., Ullah, S., Srivastava, G., Lin, J. C. W., & Zhao, Y. (2024). NMal-Droid: network-based android malware detection system using transfer learning and CNN-BiGRU ensemble. *Wireless Networks*, *30*(6), 6177-6198.

14. Almahmoud, M., Alzu'bi, D., & Yaseen, Q. (2021). ReDroidDet: android malware detection based on recurrent neural network. *Procedia Computer Science*, *184*, 841-846.

15. Batouche, A., & Jahankhani, H. (2022). Handling novel mobile malware attacks with optimised machine learning based detection and classification models. In *Artificial Intelligence in Cyber Security: Impact and Implications: Security Challenges, Technical and Ethical Issues, Forensic Investigative Challenges* (pp. 1-41). Cham: Springer International Publishing.