IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Detecting Malware Attacks On Mobile Phones Caused By Adware

[1]Ankit Pareek, [2]Ameyaa Chivukula

[1]Student, [2]Student

[12]Department of Data Science and Computer Applications,

[12]Manipal Institute of Technology, Manipal Academy of Higher Education, Udupi, Karnataka, India

Abstract: Adware, often dismissed as a low-priority threat in mobile security, has evolved into a sophisticated attack vector capable of compromising user privacy, draining resources, and facilitating data exfiltration. This paper presents a novel, behavior-based approach to detecting adware attacks on mobile phones, addressing the growing challenge of aggressive ad libraries, click fraud, and covert data transmission. Our methodology integrates both static and dynamic feature extraction, focusing on permission overuse, sensitive API call patterns, and behavioral anomalies such as excessive CPU usage, battery drain, and ad network interactions. We employ a Random Forest classifier optimized for mobile deployment, achieving a detection accuracy of 96.12% and an AUC-ROC score of 0.9914. Explainability techniques, including SHAP analysis, provide transparent insights into model decisions, highlighting key features like READ PHONE STATE and RECEIVE BOOT COMPLETED as primary indicators. The proposed system, with its lightweight architecture (1.7 MB) and low false positive rate (1.42%), offers practical applications for enterprise mobile device management, app store review automation, and end-user security tools. Future directions include incorporating network traffic analysis and adaptive learning to counter advanced adware evasion techniques.

Index Terms - Mobile adware detection, Android malware, behavior-based analysis, machine learning, Random Forest, mobile security, mobile device management (MDM).

I. INTRODUCTION

The proliferation of smartphones has revolutionized the way we interact with technology, offering unprecedented convenience and connectivity. However, this rapid expansion has also attracted malicious actors, leading to a surge in mobile malware threats. Among these, adware a class of malware designed to deliver intrusive advertisements has emerged as a particularly insidious threat, often overlooked in traditional security paradigms. Unlike conventional malware, which typically focuses on data theft or system disruption, adware operates by covertly embedding aggressive advertising mechanisms within seemingly benign applications. These mechanisms can result in click fraud, unauthorized data transmission, battery and resource drain, and privacy violations. The subtlety of adware's behavior, coupled with its integration into legitimate advertising frameworks, poses a significant challenge for detection systems, particularly those relying solely on static signatures or heuristics. This paper addresses this challenge by proposing a novel, behavior-based detection framework for adware on mobile devices. By focusing on the behavioral footprint of adware, our approach aims to detect not just known threats but also evolving variants that might evade static detection. Furthermore, recognizing the importance of privacy and resource efficiency in mobile environments, we emphasize lightweight, on-device machine learning techniques, such as model quantization and TinyML. Our system ensures real- time detection without the need to offload sensitive user data to cloud servers. To enhance transparency and trust in our detection outcomes, we integrate explainable AI techniques like SHAP (SHapley Additive exPlanations), enabling security analysts, app store reviewers, and end-users to understand the rationale behind each classification decision.

The contributions of this paper are threefold:

- A behavioral modeling approach for detecting Android adware, leveraging both static and dynamic features.
- A lightweight Random Forest classifier, optimized for mobile deployment, achieving high accuracy (96.12%) and a low false positive rate (1.42%).
- Integration of explainability techniques to provide trans- parent, interpretable insights into model decisions, enhancing trustworthiness and usability in real-world scenarios.

II. Literature Review

The rapid expansion of Android smartphones has increased exposure to malware threats, with adware emerging as a stealthy yet understudied vector. While many studies address malware detection, few focus explicitly on adware, particularly using behavior-based methods. Static only and signature-based methods struggle with post installation threats like Man in the Disk (.MitD) and privilege escalation [1], [2]. Hybrid models that combine static and dynamic analysis improve resilience against obfuscation tactics used by adware, such as junk code and encrypted strings [3], [4]. Techniques using PCA and gain ratio further enhance detection performance across malware categories including adware [5], [6]. Advanced detection models using Ant Colony Optimization, Stacked Autoencoders, and CNNs demonstrate strong adaptability to evolving threats and high classification accuracy [7], [8]. Lightweight, privacy focused solutions using on-device machine learning align with the need for user-centric protection [9]. However, antivirus effectiveness varies, revealing inconsistencies in current detection capabilities [10]. Recent work on permission weighted models using KNN and Naïve Bayes offers improved static detection but is limited to known threats [11]. CNN-based malware classification via feature to image transformation shows promise in static detection [12]. GAN-based adversarial attacks like DOpGAN reveal vulnerabilities in existing models and emphasize the importance of robustness against manipulated malware samples [13]. A comprehensive taxonomy of evasive techniques shows that many Android malware samples actively avoid detection in analysis environments, necessitating stronger sandboxing solutions [14]. Random Forest-based models offer effective static classification with high accuracy [15], while deep learning frameworks trained on large datasets have proven effective in identifying and classifying adware in real-world applications [16]. Despite progress, adware is still frequently downplayed as low severity. With its involvement in click fraud, data leaks, and resource misuse, it requires dedicated, explainable, and adaptive detection approaches. This research addresses that need with a behavior-based, lightweight, and privacy aware detection model for Android adware.

A summary of the referenced studies, including their focus areas, methods, metrics, and key findings, is provided in Table 1.

Table 1: Papers Referred on Android Malware Detection: Methods, Metrics, and Algorithms

Sl. No	Paper Title	Methodology	Accuracy	Conclusion
1	An Enhanced Framework to Mitigate Post-Installation Cyber Attacks on Android Apps	Post-Installation App Detection Method (focus on MitD attacks)	-	Introduced a method to monitor and regulate sensitive data flows to detect and mitigate post-installation attacks like MitD.
2	Evaluation and Classification of Obfuscated Android Malware Through Deep Learning Using Ensemble Voting Mechanism	Hybrid static + dynamic analysis, deep learning with ensemble voting	-	Demonstrated effectiveness in detecting obfuscated malware variants using a lightweight, scalable DL model on Kronodroid dataset.
3	AntDroidNet Cybersecurity Model: A Hybrid Integration of Ant Colony Optimization and Deep Neural Networks	Hybrid of Ant Colony Optimization (ACO) + Deep Neural Networks (DNN)	99.89%	ACO for feature selection and DNN for classification yielded high detection accuracy and low false positives.
4	Enhanced Malware Detection for Mobile Operating Systems Using Machine Learning and	Machine Learning, PCA for dimensionality reduction, dynamic/static analysis	-	Proposed a robust detection method leveraging PCA and hybrid analysis for smartphone malware.

	Dynamic Analysis			
5	Investigation of Cyber Attacks Using Post-Installation App Detection Method	Post-Installation App Detection Method	97%	Focused on MITD attacks, successfully monitored and prevented post-installation threats.
6	Recommendation Mobile Antivirus for Android Smartphones Based on Malware Detection	Comparative analysis using VirusTotal and Euphony app	-	Ranked top antivirus software (Ikarus, Fortinet, etc.) based on detection capabilities for Android malware.
7	A Malware Detection System Using a Hybrid Approach of Multihead Attention-based Control Flow Traces and Image Visualization	Multi-head attention, API Call Graphs + image-based analysis	99.27%	Combining structural (CFG) and visual features resulted in high-performance Android malware classification.
8	Lightweight On-Device Detection of Android Malware Based on the Koodous Platform and Machine Learning	Lightweight neural network model using Koodous dataset	F1 = 0.77, Precision = 0.9	Designed for on-device real-time detection; emphasizes low latency and high precision.
9	Android Malware Detection Approach Using Stacked AutoEncoder and CNN	Stacked AutoEncoder (SAE) + Convolutional Neural Networks (CNN)	98.5%	High accuracy classification using Drebin-215 dataset; binary visualization of malware samples.
10	Android Malware Classification Using Gain Ratio and Ensembled Machine Learning	Gain ratio for feature selection + ensemble ML (RF, ET, k-NN)	94.57%	Ensemble classifiers with selected features enhanced malware detection on CICMalDroid2020 dataset.
11	Android Malware Detection using Permission Based Static Analysis	Static analysis with permission weighting; classifiers: KNN, Naïve Bayes	KNN: 93%, NB: 91%	Permission-based detection with weighted scoring improved accuracy over prior models.
12	Visualising Static Features and Classifying Android Malware Using a CNN Approach	Static analysis with feature embedding (BERT) + image creation + CNN	91%, F1: 89%	Image-based CNN classification using static features effectively distinguished malware.
13	GEAAD: Generating Evasive Adversarial Attacks Against Android Malware Defense	Opcode modification using GANs (An-drOpGAN, DOpGAN)	-	Developed adversarial Android malware to evade ML detectors; emphasized need for adversarial detection defenses.
14	Unmasking the Veiled: A Comprehensive Analysis of Android Evasive Malware	Sandbox-based dynamic analysis with taxonomy of evasive techniques	-	Created DroidDungeon sandbox; revealed 14% of malware avoid execution under traditional analysis due to evasive checks.
15	Android Malware Detection Using the Random Forest Algorithm	Static analysis using Random Forest classifier	98.47%, F1: 98.6%	RF classifier outperformed other models; highly effective in Android malware detection.
16	Deep Learning-Based Attack Detection and Classification in An- droid Devices	Deep learning with supervised learning on labeled multi-class	F1: 98.9%, FP rate: 0.99%	Multi-class DL classifier showed high effectiveness; created Android app for real-

		datasets		world validation.

III. METHODOLOGY

This study proposes a comprehensive framework for detecting Android adware using machine learning, with a focus on behavior-based feature engineering and explainable AI. The methodology is structured into several key phases: data preparation, feature engineering, model development, evaluation strategy, and interpretability. The overall workflow is illustrated in Fig. 1.

Fig. 1 Methodology Flowchart for Adware Detection

3.1 Data Preparation

The dataset for this research comprises Android application samples, both adware-infected and benign, collected from reputable sources including malware repositories and official app stores. Each sample includes metadata, extracted permissions, API calls, and behavioral logs.

Key steps in data preparation include:

- Deduplication: Removal of duplicate samples to ensure data integrity.
- Label Verification: Manual and automated validation of class labels (adware or benign).
- Data Cleaning: Handling of missing values and removal of irrelevant or corrupted samples.

Class Balance Assessment: Analysis of the distribution of adware and benign samples to guide any necessary resampling or balancing strategies.

3.2 Feature Engineering

Feature engineering was designed to capture both static and behavioral characteristics indicative of adware. The process included:

- Permission Analysis: Extraction of requested permissions from app manifests, focusing on those commonly abused by adware (e.g., phone state, boot completion, location, and network access).
- API Call Patterns: Identification and quantification of sensitive API calls, such as those related to network connections and location services.
- Behavioral Indicators: Measurement of runtime behaviors, including pop-up frequency, battery and CPU usage, and data transmission patterns.
- Feature Selection: Statistical and domain-driven selection of the most informative features to optimize detection accuracy and computational efficiency.

All features were normalized or encoded as appropriate for machine learning algorithms.

3.3 Model Development

Multiple machine learning algorithms were considered for the adware detection task, with a focus on models suitable for mobile deployment. The primary classifier selected was the Random Forest algorithm, due to its robustness, interpretability, and proven effectiveness in malware detection tasks. Other models, such as Support Vector Machines and deep learning architectures, were evaluated as baselines.

The model development process included:

- Training and Testing Split: Stratified division of the dataset into training and test sets to ensure representative class distributions.
- Cross-Validation: Use of k-fold cross-validation during training to assess model robustness and prevent overfitting.

- Hyperparameter Optimization: Systematic tuning of model parameters to maximize performance.
- Class Imbalance Handling: Application of oversampling or class weighting techniques as needed.

3.4 Evaluation Strategy

Model performance was evaluated using a comprehensive set of metrics, including:

- Accuracy: Overall proportion of correctly classified samples.
- Precision, Recall, and F1-Score: Assessment of the model's effectiveness in identifying adware while minimizing false positives and false negatives.
- AUC-ROC: Measurement of the model's ability to distinguish between adware and benign samples.
- Confusion Matrix: Detailed breakdown of true positives, true negatives, false positives, and false negatives.
- Feature Importance Analysis: Identification of the most influential features contributing to the model's decisions, supporting explainability and transparency.

All evaluation metrics were calculated on the held-out test set to provide an unbiased assessment of model performance.

3.5 Visualization

Key visualizations enhance interpretability:

- Confusion Matrix (Fig. 2): Classification performance heatmap
- ROC/Precision-Recall Curves (Fig. 8): Threshold trade-offs and AUC-ROC
- Feature Importance (Fig. 3): Top predictive traits in Random Forest
- SHAP Plots (Figs. 5-7): Global/local feature contributions

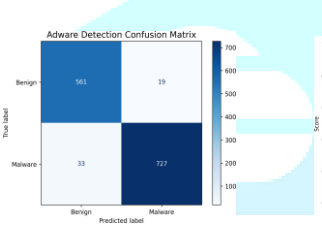

Fig. 2 Confusion Matrix

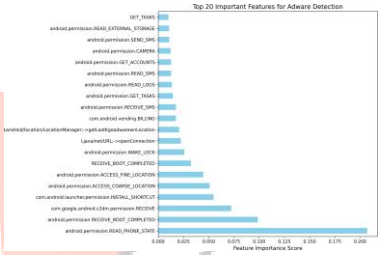

Fig. 3 Feature Importance

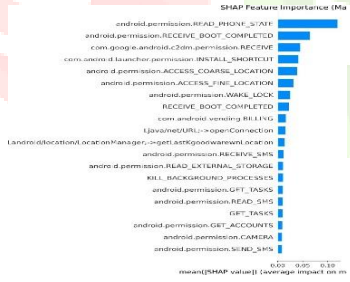

Fig. 5 SHAP Summary Bar Plot

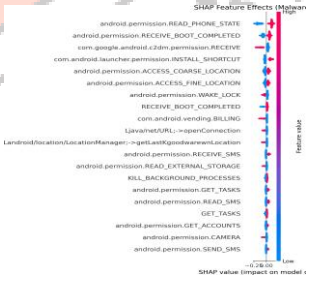

Fig. 6 SHAP Summary Dot Plot

Fig. 6 SHAP Force Plot

IV. Results and Discussion

4.1 Classification Performance

The proposed adware detection system demonstrated exceptional performance using the Random Forest classifier. As shown in the classification report in Table 2, the model achieved a 96.12% accuracy and an AUC-ROC score of 0.9914, indicating a high degree of separation between benign and malicious samples.

Table 2 Model Performance Summary

Metric	Value
Accuracy	96.12%
AUC-ROC	0.9914
Precision (Adware)	97%
Recall (Adware)	96%
F1-Score (Adware)	97%
False Positive Rate	1.42%

Confusion Matrix Analysis (Fig. 2):

- True Negatives: 561
- False Positives: 19
- False Negatives: 33
- True Positives: 727

The confusion matrix highlights the model's ability to accurately detect adware with minimal false alarms. The low false positive rate (1.42%) is particularly significant for real- world usage, ensuring legitimate apps are rarely misclassified.

Visual Summary of Performance Metrics:

See Fig. 8 for a visual comparison of accuracy, ROC curve, and precision-recall curve.

4.2 Feature Importance Analysis

The feature importance analysis, visualized in Fig. 3 and Fig. 4, shows that a small set of critical permissions and behaviors heavily influence detection outcome.

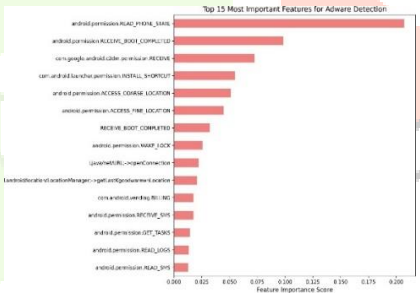

Fig. 4 Detailed Feature Importance

Top Influential Features:

- READ PHONE STATE – 20.71%
- RECEIVE BOOT COMPLETED – 9.86%
- C2DM.RECEIVE – 7.24%
- INSTALL SHORTCUT – 5.48%
- ACCESS COARSE LOCATION – 5.09%

Key Insights:

- Permission Overuse: Features related to phone state and boot completion dominate the decision process, accounting for more than 30% of the model's predictive power.
- Behavioral Triggers: Features like WAKE LOCK and sensitive API calls suggest stealthy behaviors such as running in the background or initiating network requests without user interaction.

4.3 Explainability via SHAP Analysis

To support transparency and trust in model predictions, SHAP (SHapley Additive exPlanations) values were computed. These explain how individual features influence specific predictions and overall model behavior.

SHAP Visualizations:

- Figure 5 – SHAP Summary Bar Plot
- Figure 6 – SHAP Summary Dot Plot

- Figure 7 – SHAP Force Plot

Insights from SHAP:

- READ PHONE STATE and RECEIVE BOOT COMPLETED consistently have the highest impact on pushing predictions toward the adware class.
- The SHAP force plot demonstrates case-specific reasoning, revealing why certain apps were labelled malicious, providing auditability and interpretability for security analysts.

These explanations can be critical for app store reviewers or forensic analysts seeking to understand classification outcomes beyond black-box predictions.

4.4 Practical Implications

Given the model's high precision and interpretability, it is suitable for multiple real-world applications:

- Enterprise Mobile Device Management (MDM): The small model size (~1.7 MB) and low FP rate make it ideal for integration into corporate device protection systems.
- App Store Review Automation: SHAP-based insights allow reviewers to validate whether permissions like READ PHONE STATE are justified by the app's purpose.
- End-User Security Tools: By highlighting suspicious behaviors linked to critical features, the system can alert users about background activity typical of adware.

4.5 Limitations and Future Work

While the current system performs well on permission- and behavior-based adware, it may face challenges in detecting advanced threats that:

- Use reflection or obfuscation to hide sensitive API calls.
- Encrypt network traffic to bypass feature-based analysis.
- Mimic benign app patterns to reduce behavioral deviations.

Future Directions:

- Incorporate network traffic inspection and dynamic analysis to uncover hidden behaviors.
- Explore online learning and reinforcement learning to adapt to evolving adware variants in real-time.

V. CONCLUSION

In this paper, we presented a novel, behavior-based framework for detecting adware on mobile devices, leveraging both static and dynamic indicators such as permission overuse, sensitive API call patterns, and behavioral anomalies like excessive pop- ups and resource consumption. By focusing specifically on adware, an often underestimated but increasingly prevalent threat, we have elevated its significance in the context of mobile security. Our approach combines a lightweight Random Forest classifier with explainable AI techniques like SHAP, ensuring not only high detection accuracy (96.12%) and a low false positive rate (1.42%) but also interpretability and transparency of decisions. This dual focus addresses both technical robustness and the practical need for trustworthiness in real-world deployment scenarios, such as enterprise mobile device management, app store review automation, and end- user security tools. While our system demonstrates strong performance, limitations remain in detecting more sophisticated adware variants that employ obfuscation or mimic benign app behaviors. Future work will explore dynamic network traffic analysis, reinforcement learning, and advanced anomaly detection techniques to further enhance detection capabilities against evolving adware threats. Overall, this research contributes a targeted, efficient, and explainable solution for mobile adware detection, aligning with the growing need for privacy-preserving and user-centric mobile security technologies.

REFERENCES

[1] V. Koka and K. Muppavaram, "An enhanced framework to mitigate post-installation cyber attacks on android apps," Engineering, Technology & Applied Science Research, vol. 14, pp. 14 788–14 792, 2024. [Online]. Available: https://doi.org/10.48084/etasr.7467

[2] K. K. e. a. Mamidi, "Investigation of cyber attacks using post-installation app detection method," Cogent Engineering, vol. 11, p. 2411859, 2024. [Online]. Available: https://doi.org/10.1080/23311916.2024.2411859

[3] S. Aurangzeb and M. Aleem, "Evaluation and classification of obfuscated android malware through deep learning using ensemble voting mechanism," Scientific Reports, vol. 13, p. 3093, 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-30028-w

[4] F. e. a. Ullah, "A malware detection system using a hybrid approach of multi-heads attention-based control flow traces and image visualization," Journal of Cloud Computing, vol. 11, p. 75, 2022. [Online]. Available: https://doi.org/10.1186/s13677-022-00349-8

[5] F. M. M. Aledam and B. M. A. Al-Latteef, "Enhanced malware detection for mobile operating systems using machine learning and dynamic analysis," International Journal of Safety and Security Engineering, vol. 14, pp. 513–521, 2024. [Online]. Available: https://doi.org/10.18280/ijsse.140218

[6] D. B. Ansori, J. Slamet, M. Z. Ghufron, M. A. R. Putra, and T. Ahmad, "Android malware classification using gain ratio and ensembled machine learning," International Journal of Safety and Security Engineering, vol. 14, no. 1, pp. 259–266, 2024. [Online]. Available: https://doi.org/10.18280/ijsse.140126

[7] R. R. N. e. a. AlOgaili, "Antdroidnet cybersecurity model: A hybrid integration of ant colony optimization and deep neural networks for android malware detection," Mesopotamian Journal of Cybersecurity, vol. 5, pp. 104–120, 2025. [Online]. Available: https://doi.org/10.58496/MJCS/2025/008

[8] B. e. a. Menaouer, "Android malware detection approach using stacked autoencoder and convolutional neural networks," International Journal of Intelligent Information Technologies, vol. 19, pp. 1–25, 2023. [Online]. Available: https://doi.org/10.4018/IJIIT.329956

[9] M. e. a. Krzyszton´, "Lightweight on-device detection of android malware based on the koodous platform and machine learning," Sensors, vol. 22, p. 6562, 2022. [Online]. Available: https://doi.org/10.3390/s22176562

[10] H. e. a. Saputra, "Recommendation mobile antivirus for android smartphones based on malware detection," IAES International Journal of Artificial Intelligence, vol. 13, pp. 3559–3566, 2024. [Online]. Available: https://doi.org/10.11591/ijai.v13.i3.pp3559-3566

[11] N. A. Mohd Ariffin and H. P. Casinto, "Android malware detection using permission based static analysis," Journal of Advanced Research in Applied Sciences and Engineering Technology, vol. 33, no. 3, pp. 86–97, 2024. [Online]. Available: https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/article/view/8697

[12] Kiraz and A. Dog˘ru, "Visualising static features and classifying android malware using a convolutional neural network approach," Applied Sciences, vol. 14, no. 11, p. 4772, 2024. [Online]. Available: https://doi.org/10.3390/app14114772

[13] N. Ahmad, A. S. Rana, H. J. Hadi, F. B. Hussain, P. Chakrabarti, M. A. Alshara, and T. Chakrabarti, "Geaad: Generating evasive adversarial attacks against android malware defense," Scientific Reports, vol. 15, p. 11867, 2025. [Online]. Available: https://doi.org/10.1038/s41598-025-96392-x

[14] A. Ruggia, D. Nisi, S. Dambra, A. Merlo, D. Balzarotti, and S. Aonzo, "Unmasking the veiled: A comprehensive analysis of android evasive malware," in Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS '24), 2024, pp. 1–16. [Online]. Available: https://doi.org/10.1145/3634737.3637658

[15] A. El Attaoui, N. El Hami, and Y. Koulou, "Android malware detection using the random forest algorithm," Indonesian Journal of Electrical Engineering and Computer Science, vol. 36, no. 3, pp. 1876–1883, 2024. [Online]. Available: http://ijeecs.iaescore.com/index.php/IJEECS/ article/view/33849

[16] A. Go´mez and A. Mun˜oz, "Deep learning-based attack detection and classification in android devices," Electronics, vol. 12, no. 15, p. 3253, 2023. [Online]. Available: https://doi.org/10.3390/electronics12153253