Yasin et al. LGURJCSIT 2025

ISSN: 2521-0122 (Online)
ISSN: 2519-7991 (Print)

LGU Research Journal of Computer Science & IT

doi: 10.54692/lgurjcsit.2025.092726

Vol (9): Issue (2), July – December 2025

AMDetect: Attention-Driven Hybrid Deep Learning Framework for Evasion-Resilient Mobile Malware Detection

Ghulam Yasin[1*], Faizan Ismail[2], Natash Ali Mian[3], Hasan Sardar[2], Rubaina Nazeer[4]
[1]University of Education Lahore DGK Campus, Pakistan.
[2]COMSATS University Islamabad Wah Campus, Pakistan.
[3]School of Computer and Information Technology, Beaconhouse National University Lahore, Pakistan.
[4]National University of Modern Languages Islamabad, Multan Campus Pakistan.

Email: ghulamyasin@ue.edu.pk

ABSTRACT:

Malware detection methods include dynamic and the analysis section. Although the speed of the static analysis is high, it is not capable of identifying obfuscated variants. On the contrary, the dynamic analysis is efficient, but is slower and more resource-consuming. The paper proposes a hybrid machine learning-based approach to enhance exposure but building the strength of the mentioned approaches. AMDetect is a better malware detector algorithm that aims at safeguarding smartphones and their applications in the marketplaces as well as the marketplace malicious software on the rise. With the help of both hybrid assessment and a deep neural network classifier, AMDetect can successfully detect and differentiate a suspicious and a harmful application. It uses a distinct class modeling method according to the Application Programming Interface (API) used that emphasizes the attention process to identify particular features resulting in the classification of every application. The performance of the algorithm is tested on the state of art standard datasets which prove that this algorithm is better than the existing mechanisms with a remarkable accuracy of 98.90. AMDetect will enhance the safety of smartphones and help to protect application marketplaces by identifying malicious components of mobile applications. As one of the strongest tools, with a high level of detection of malware, AMDetect provides a strong tool in overcoming the constantly changing difficulties of mobile threats.

KEYWORDS: Mobile Malware Detection, AMDetect, Ransomware, Machine Learning, Neural Network Classifier, Android Security

1. INTRODUCTION

The Internet constitutes a great point of information exchange among various nodes which include millions of computers, networks and devices. Accordingly, it has become a main target for cybercriminals [1]. The extensive rise of mobile devices and use of mobile applications has significantly changed our living styles, affecting our office work, communication requirements, and how we access required information. This has also attracted the interest of black hats who can gain access to the mobile operating systems and apps with vulnerability, which is a cause of increased malware in mobile phone usage. Mobile malware, particularly developed to target specific devices, e.g., smartphones and tablets, affects the devices in different manners, e.g., Trojan horses, viruses, worms (malware), spyware viruses, and ransomware (malware) [2]. Such threats may cause unauthorized access to data, stealing money, identity theft, and even controlling the compromised devices remotely. Mobile devices are being used to store a huge volume of sensitive

data needed for fast and early mobile malware threat identification; categorization methods are essential to protect data centers. Conventional threat identification techniques, like rule-based detection and signature-based systems, are in practice; constantly developing tactics for malware developers, who employ techniques like code obfuscation and polymorphic code to evade detection. All of this combined with the total number of mobile applications makes manual analysis impractical [3]. In response, advanced deep learning approaches have arisen as a good solution for mobile malware detection and classification. Machine learning models are able to train on large data sets, discover complicated patterns, and autonomously make predictive decisions. Using these abilities, scientists are trying to exploit more dynamic and resilient malware detection schemes that will be able to overcome both known malware and the unknown malware [4]. This study proposes a hybrid approach to mobile malware detection that involves multiple machine learning models such as deep learning, ensemble learning, and transfer learning to improve detection accuracy, efficiency, and flexibility.

The capability of deep learning to acquire complex features, the strength of ensemble learning with model combination, and the transfer learning with the ability to work with limited data are all part of a complete solution to detection. The suggested strategy combines the use of the static and dynamic and the behavioral patterns of the mobile applications. The relevance of this research in enhancing mobile security lies in the possibility of enhancing identification and classification of threat malware in the mobile system, thereby making the users resistant to the ever-changing threat environment. The subsequent chapters will review existing literature, present the hybrid approach's methodology, detail the experimental setup and findings, and provide an in-depth analysis. This research concludes with insights into contributions, limitations, and future directions for advancing mobile malware identification and categorization using machine learning approaches. Figure 1 illustrates the process of malware identification and categorization by artificial intelligence, including static and dynamic analysis deep learning techniques.

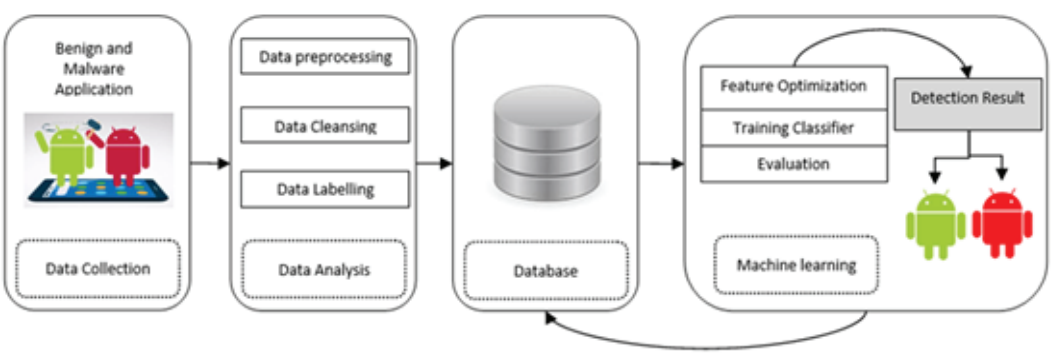

Figure 1: Malware detection and classification [1]

2. LITERATURE REVIEW

In [5], the Android malware identifiers are deep learning (DL) and semi-supervised, used to address the threats of privilege escalation using an adaptive framework. The models perform with an accuracy of 99.024, which proves to be effective when facing zero-day attacks because the models learn new malicious behaviors without needing to be manually labeled, yet issues on scalability and in general real-world applications have not been addressed yet. AI-HydRa, a hybrid approach combining DL and random forest [6], uses a majority-vote system based on both static and dynamic features, reducing false positives. However, its high time complexity limits practicality for large datasets. A enhancing accuracy while DL-based system designed for mobile device malware detection demonstrates strong detection rates and low false positives [7], though its computational demands could strain mobile device batteries. The system's comparative performance with state-of-the-art models remains undetermined. Another DL method employs a multi-stream network for classifying Android malware families, achieving high accuracy and computational efficiency but lacking interpretability and comparative analysis. In [8], More recently, two-stage ransomware

detection model with a Markov chain to identify behavioral patterns and a random forest classifier to reach high accuracy has been developed but its computational cost is too high to run on resource-constrained mobile devices; thus, more efficient attention-driven hybrid deep-learning frameworks such as AMDetect are needed. A clustering and ML-based study [9] focuses on Android malware families, identifying common system calls to track malware behavior. While the framework is effective for Android malware, generalizability to other malware types is limited. The XMAL model uses an interpretable ML approach to classify Android malware with a multi-layer perceptron and attention mechanism, producing human-readable explanations. While accuracy is validated, its efficiency compared to other models is unexplored. Techniques combining static and dynamic analysis for malware family characterization on a large dataset [10] highlight limitations such as device dependency and ethical concerns, with generalizability across malware types unproven. Random forest predictors classify ransomware using gain-ratio-selected features, which are fast and accurate, but may fail to consider important low-gain features when depending on a single parameter of feature selection. The success of the use of differential area analysis to detect ransomware [11] is very high (99.96) with the use of entropy in the initial files bytes with no comparison with the current detection systems. Another recent method to detect malicious apps [12], considers Google Play Store datasets and speaks about problems in feature extraction and ethics, but does not provide a comprehensive generalization and privacy protection. Android malware detection System is a scalable overlay-based system that is well-suited to real-time malware detection on resource-constrained devices, but it needs frequent updates because malware is evolving and is not state-of-the-art with regards to comparison. RansomCare is the smartphone ransomware countermeasure that operates on both dynamic and statistical analysis to detect the ransomware and protect user data [13]. An Android malware dataset debiasing approach [14] improves the way the datasets are represented, but it cannot be applied to all datasets, and it does not discuss any ethical issues. A-Pot is an Android malware analysis platform based on container technology [15], which speedily automates the malware detection process but does not provide information about the potential difficulties with the deployment. Formal equivalence checking [16] is a method that classifies different families of mobile malware with accuracy and scalability at a high cost of computation restricting it to low-end devices, and to detect novel malware instances. All models have their own advantages, but are confronted with limitations, including the ability to perform computations, the size of datasets, or coverage of malware types that require additional research and development to allow them to be used more broadly and efficiently. There have been attacks of mobile malware that have become major security threats. Emerging evasion techniques, including repackaging applications into new packages and obfuscating code are posing a challenge to the conventional malware detection tools [17]. Malware samples that attack the Android and iOS operating systems have been increasing exponentially, as well as advanced persistent threats (APTs) [18]. Better, dynamic methods of detection are required to fight such dynamic mobile threats.

3. PROPOSED METHOD

AMDETECT is an approach that is used to identify Android malware. We give a description of the suggested method, a hybrid analysis that consists of attention mechanism, neural networks, and class modeling technique. AMDETECT tone out the limitations of each of the mentioned approaches by using hybrid techniques that utilize the significance of a static analysis and a dynamic one. The use of a classifier (neural network) improves the accuracy of label predictions, while AMDETECT not only predicts the app's label (malicious or benign) but also identifies the specific classes, permissions, and API calls associated with malicious activities. the attention mechanism identifies critical features for classification. AMDETECT is a system that takes an app (Android-based) as input and provides the app's overall classification of (malware or goodware) along with its key features for classification. It consists of three main stages:

I) Static analysis
II) Dynamic analysis
III) Machine learning

App reports are generated through static analysis and evaluated by the Evasive Malware Predictor for dynamic analysis determination. The Anti-Evasion Dynamic Analyzer carries out the implementation in a controlled environment and

creates a report based on dynamic analysis. The aggregator merges static and dynamic reports, while the Feature Vector Generator (FVG) transforms them into appropriate feature vectors. The feature vectors are classified using a neural network that identifies malicious applications and outputs their key features. Figure 2 illustrates the entire design of the proposed AMDetect framework. It provides the three key stages of the process, which are Static Analyzer, Dynamic Analyzer, and Neural Network Classifier, showing how the data flows between modules like Evasive Predictor, Aggregator, and Feature Vector Generator in detecting malware of hybrid type.

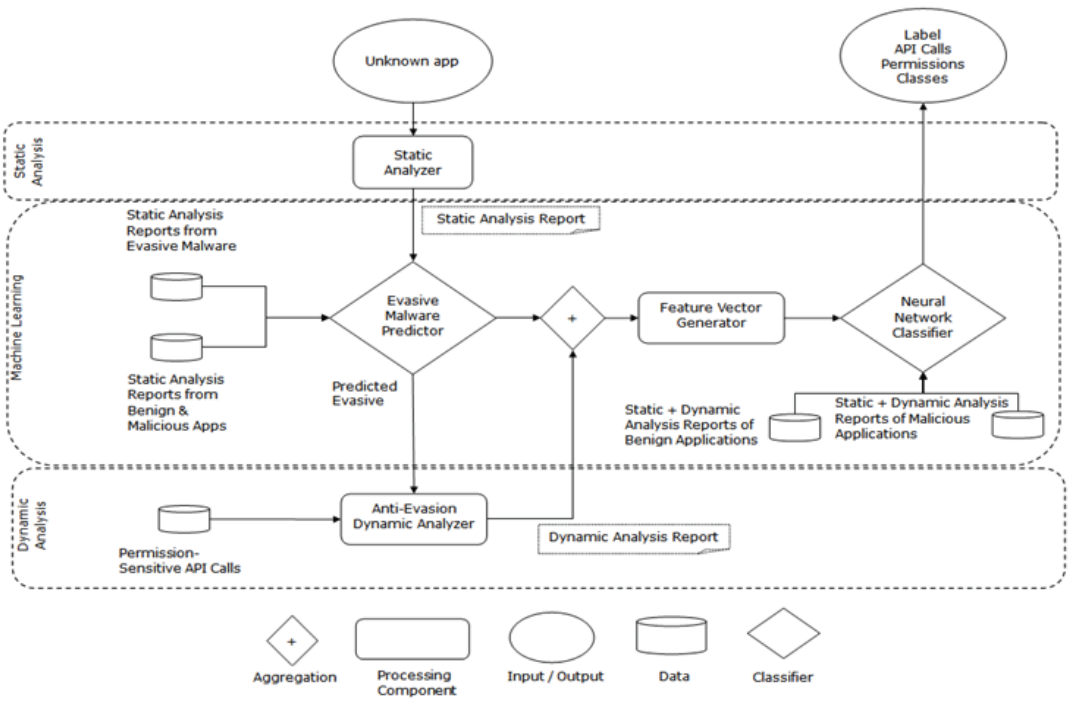

Figure 2: Architecture of AMDetect

App1	
Permissions	P1, P2
Class1	A1, A2
Class2	A2, A3, A4
Class3	A3, A4

(a)

App2	
Permissions	P1, P3
Class1	A1, A3, A5
Class2	A3, A4

(b)

App3	
Permissions	P2
Class1	A1, A3, A4, A5

(c)

Figure 3: Android apps along with their respective classes, APIs, and permissions as examples

3.1. Static Analyzer

The Android app is processed by this component to generate a report based on static analysis. This report includes the app's model represented by Java classes, API calls, and permissions. A unique approach is presented in this study, where each application class is modeled based on the APIs it uses. AMDETECT, unlike other methods [19], not only captures common API calls and permissions but also identifies shared classes among applications through its application class modeling technique. Figure 3 shows some Android applications and their respective classes, API calls, and permissions. It explains the concept of class modeling involved in AMDetect, and it reveals how each component of apps and API utilization is modeled to extract features that will be utilized in the next stage of classification.

3.2. Evasive Predictor (Malware)

The Evasive Predictor classifies applications to determine if dynamic analysis is necessary, focusing on detecting evasive malware. Trained with samples of datasets of both evasive and

non-evasive Android apps, this model enables AMDETECT to identify potential evasive behavior. If the classifier predicts an app as non-evasive, its static analysis findings are sent to Aggregator. Conversely, as long as evasive behavior is detected, the app is forwarded to the Anti-evasion Dynamic Analyzer for deeper evaluation. Figure 4 shows the vectors that are obtained after the analysis of three Android applications, both static and dynamic. It shows the process of converting raw application behavior data into numerical feature representations by AMDetect, which can be successfully trained and classified by the neural network.

3.3. Anti-Evasion Analyzer

The anti-evasion module is essential for detecting evasive malware. Dynamic analysis is performed on the flagged app, which is received from the evasive predictor within an Android sandbox, using CamoDroid [20] to counteract evasion techniques. CamoDroid creates a secure environment, hiding its nature from apps while logging interactions with about 400 permission-protected APIs. The resulting dynamic analysis report, offering insights into app behavior, is then sent to the aggregator for further processing.

3.4. Aggregator

Upon obtaining the dynamic as well as static findings, the aggregator component proceeds to send these reports to the Feature Vector Constructor. The aggregator is an important agent and is used to gather information required under the two kinds of analysis and makes sure that it is transmitted to the subsequent stage of the process. This enables the neater integration of the analysis reports and building of feature vectors, which are also important in subsequent analysis and classification processes.

	Permissions			APIs					Classes				
APP\Feature	P1	P2	P3	A1	A2	A3	A4	A5	A1_A2	A2_A3_A4	A3_A4	A1_A3_A5	A1_A3_A4_A5
App1	1	1	0	1	1	1	1	0	1	1	1	0	0
App2	1	0	1	1	0	1	1	1	0	0	1	1	0
App3	0	1	0	1	0	1	1	1	0	0	0	0	1

Figure 4: AMDetect generates feature vectors for three sample apps

3.5. Feature Vector Constructor (FVS)

The application of both the static and dynamic analysis to a representation of all applications depends on the Feature Vector (FV). It summarizes the data used, such as classes, permissions, and used APIs, among others, into a comprehensible format. The feature vector synthesizes the key features of the app based on the information obtained in both analysis processes, hence acting as valuable input in undertaking additional analysis and classification exercises.

3.6 Neural Network Classifier (NNC)

With the help of NNC, it is essential to determine the maliciousness of an application and apply the labels of malware and emphasize important classes, Application Programming Interface (API), and approvals associated with the categorization. In contrast to traditional approaches, AMDETECT focuses on features relevant to individual apps under an attention technique and a multi-layer perceptron. The attention mechanism [21] concentrates on particular input components in the prediction process to analyze them in details, gauge the significance of features by using a fully connected network and softmax functionality. The resulting attention vector is inputted into the MLP, which puts the weights of features into a binary label malicious or benign. The NNC is trained on both benign and malicious feature vectors, which allow it to give labels and also highlight important features to an app feature vector. Thereby, tested AMDETECT in the experiments of dataset analysis, evaluation of impact features, and comparison with the existing Android malware detectors. Discussion of the results of the detection, which proves its effectiveness in detecting Android malware and provides important information to researchers and analysts.

3.7. Dataset Description

To create a complete dataset, Android applications are received over a variety of channels, and both benign and malicious

applications are analyzed with a lot of consideration. The volume of harmless information is 10,381 applications that can be located in the Google Play Store. On the case of malicious applications, we obtain the data collected by various sources: 5,560 by Drebin, 15,680 by VirusTotal, and 227 evasive malware apps by CamoDroid. This multi-source strategy guarantees a wide sample of malware to effectively test.

A final dataset will consist of: after the processing to remove duplicates and corrupted apps.

- **Benign- 9,632 Google Play applications.
- **Malicious**:
  - 5,423 from Drebin
  - 15,379 from VirusTotal
  - 226 evasive malware from CamoDroid

Overall, the total of the dataset consists of about 30,659 applications, and this is a beneficial source of our study on mobile malware detection and classification.

3.8. Pre-processing

We have 200 samples of evasive malicious malware picked in CamoDroid, 9,510 benign applications selected in Google Play Store (GPS), 5,123 malicious applications found in Drebin and 14,179 malicious applications found in VirusTotal, which makes around 30,600 entries after eliminating the duplicates and corrupted applications. To train the evasive predictor, we will take a subsample of 2,160, including 1,965 randomly chosen apps of benign and malicious types, and 200 evasive malware instances. The rest of the 28,450 samples are also predicted to have evasive behavior and further evaluated using the Anti-Evasion Dynamic Analyzer.

3.9. Feature Reduction

The apps produced 4,347,239 features in the form of all permissions, APIs, and classes. This large number is a challenge to computation efficiency and model strength. To improve AMDETECT, we consider the reduction of the feature set by choosing the relevant features with the methods like importance scoring and dimensionality reduction, trying to tradeoff between feature reduction and the discriminative ability of the model.

Setup 1: XAML + App Class

Setup 2: CamoDroid + App Class

Setup 3: Combined Features

4. RESULTS AND ANALYSIS

Efficiency of AMDetect is evaluated by employing various feature setups and performing a comparative analysis of our findings against alternative techniques. Figure 5 shows the comparison of accuracy of AMDetect (three combinations of features) and XMAL. It graphically proves the effectiveness of the hybrid approach and the combination of features as to the greater accuracy of AMDetect in all setups.

4.1. Effect of the Feature Set on the Performance of Detection

Table 1: Results from AMDetect using three different setups versus XMAL

	Accuracy	Precision	TPR	TNR	FPR	FNR
Setup 1	98.88	99.03	99.36	97.79	2.20	0.63
Setup 2	98.50	98.39	99.47	96.27	3.72	0.52
Setup 3	98.91	98.78	99.66	97.21	2.78	0.33
XMAL [21]	98.06	98.66	95.03	99.42	0.57	4.96

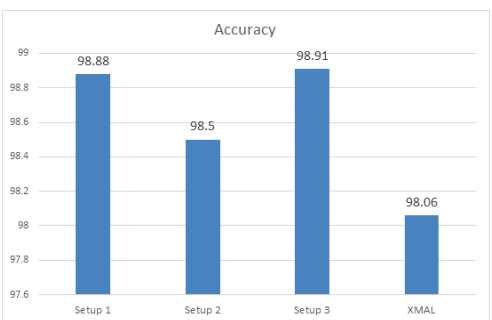

Figure 5: Accuracy graph (AMDetect using three different setups versus XMAL)

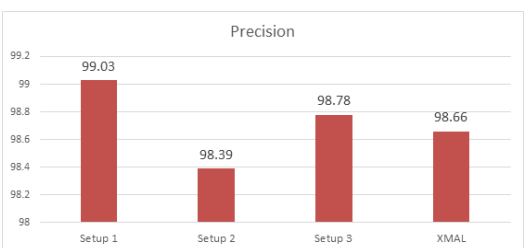

Figure 6: Precision graph (AMDetect using three different setups versus XMAL)

Figure 6 displays the precision comparison graph

between AMDetect and XMAL. The figure emphasizes AMDetect's superior precision levels, particularly under combined feature configurations, showing consistent improvements in true malware identification. Table 1 summarizes the performance metrics of AMDetect in accuracy, precision, TPR, TNR, FPR, and FNR in three setups compared to XMAL. The finding indicates that the hybrid set is better detected and has fewer false positives, which justifies the efficacy of the hybrid design of AMDetect. The evaluation of AMDETECT primarily centers on assessing how different feature sets affect its performance in detecting malware. We evaluate our Android malware detection method by comparing it to a state-of-the-art approach proposed by author [22]. In terms of performance detection parameters, the experiments demonstrate that AMDETECT surpasses the approach XMAL. By incorporating permission- protected APIs, application classes, and permissions, AMDETECT achieves higher accuracy and True Positive Rate (TPR) in distinguishing malware. Table 2 compares AMDetect's detection performance with several existing methods. It highlights AMDetect's higher accuracy (98.8%) and precision (98.9%), outperforming classical and neural approaches by integrating hybrid analysis, attention mechanisms, and application-class modeling. However, modeling only permission-protected APIs leads to lower accuracy in distinguishing benign applications. The combined scenario, including all authorization, is guarded by APIs as well as some non-authorized secured Application Programming Interfaces (APIs), yielding the best results with improved accuracy and a reduced False Positive Rate (FPR). The test results unequivocally demonstrate the notable disparity (accuracy, TNR) between AMDETECT and the method it was compared against. Precision shows a negligible difference between the two methods. Figure 7 presents a comparison of the accuracy of AMDetect and other malware detection methods of previous research. It brings out the obvious strength of AMDetect, which verifies its strength and flexibility in a variety of datasets. Figure 8 can be used to compare preciseness between various methodologies. It shows that AMDetect is very precise and consistent to use as compared to neural and non-neural baselines; hence, it is a reliable tool to detect malicious applications.

4.2 Comparative Analysis with Other Methods

Table 2: AMDetect's detection performance is evaluated and compared to other methods

Research	Approaches used	Acc	Prec	TPR	TNR	FPR	FNR
AMDetect (Proposed Method)	APIs permissions app class	98.80	98.90	98.00	99.30	0.60	1.90
Pan et al.[22]	Non-neural net models	95.7	95.4	93.8	-	-	-
Arora et al. [23]	Permission pair graphs	98.0	-	-	-	-	-
Zhang et al. [24]	Method-level correlations of apps	96	97	95	-	-	-
XU et al. [25]	Inter-component comm	97.4	-	93.1	-	0.67	-
GU et al. [19]	Call graph, permission, and package features	94.6	-	-	-	-	-

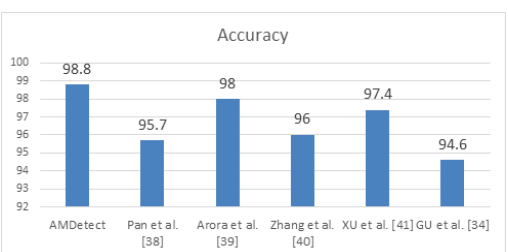

Figure 7: Accuracy graph (AMDetect's detection performance in comparison to other methods)

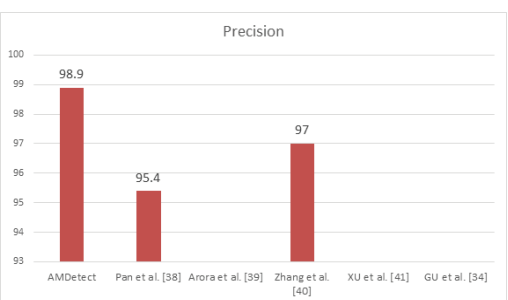

Figure 8: Precision graph (AMDetect's detection performance in comparison to other methods)

Comparing AMDETECT with other methods in the domain of Android malicious code identification is challenging on account of the different datasets and evaluation metrics used. However, we evaluate AMDETECT on the popular Drebin dataset and compare our results with other researchers' findings on the same dataset. AMDETECT performs well compared to recent works, utilizing a unique hybrid investigation and class modeling method. The performance of AMDETECT surpasses average mostly artificial neural network models as well as non-neural based network techniques. Notably, AMDETECT outperforms other neural network-based methods, except for a specific approach that lacks interpretability in highlighting malicious code snippets. These analyses prove that AMDETECT is effective in Android malware detection and it can be analyzed by human analysts.

5. CONCLUSION

Mobile malware mitigation has been a challenging concern because of the changing and complexity of threats and due to the peculiarities of mobile devices. In this paper, the author attempts to address these hurdles by providing viable solutions that can help curb mobile malware in the app stores as well as in the end-user gadgets. There is a solution that allows program analysts to observe the behavior of programs when they are running. This method will entail providing an insight into the sensitive API calls and parameter cloaking of the environment in which the analysis is performed. Also, a process of securing the application marketplaces by tracking down the presence of malicious components in the application of mobile applications and distinguishing these malicious components effectively between the application and the benign applications are introduced. The findings of the assessment of such techniques prove their efficiency in enhancing the security of smartphones and securing the application markets against mobile malware. The article is concerned with securing smartphone users against malware by adopting several methods. Android Malware Detector, (AMDETECT) is one of the major contributions. It effectively separates bad applications and good ones based on hybrid methods and a neural network classifier. AMDETECT is better at detecting malicious elements in malware applications by considering a special method of class modeling which evaluates the used APIs and through the mechanism of attention. The evaluation of the performance of AMDETECT is conducted using benchmark datasets and has an excellent recognition rate of 98.9% plus outperforms the established techniques.

The future work will focus on the expansion of AMDETECT to allow real-time on-detect malware detection with minimal resource overhead. The system can have adversarial training methods to make the system resistant to the sophisticated evasion and obfuscation techniques. In addition, giving the framework the ability to operate across the different mobile platforms and enable automatic compatibility with the application marketplace to automatically screen malicious elements will be a long way in improving the overall mobile security ecosystem.

REFERENCES

[1] M. A. Ortega-Velázquez et al., “Trust the source: A latency-based machine learning approach to accurate IP geolocation in Internet,” Computer Networks, pp. 111721, 2025.

[2] M. Mohd Saudi, M. A. Husainiamer, A. Ahmad and M. Y. I. Idris, “iOS mobile malware analysis: a state-of-the-art,” *J. Comput. Virol. Hacking Tech.,* 20(4), pp. 533–562, 2024.

[3] C. Cross and T. J. Holt, “Beyond fraud and identity theft: Assessing the impact of data breaches on individual victims,” *Journal of Crime and Justice,* pp. 1–24, 2025.

[4] L. Da Costa and V. Moia, “A lightweight and multi-stage approach for android malware detection using non-invasive machine learning techniques,” *IEEE Access,* 11, pp. 73127–73144, 2023.

[5] S. Sharmeen, S. Huda, J. Abawajy and M. M. Hassan, “An adaptive framework against android privilege escalation threats using deep learning and semi-supervised approaches,” *Appl. Soft Comput.,* 89, pp. 106089, Apr. 2020.

[6] S. Yoo, S. Kim, S. Kim and B. B. Kang, “AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification,” *Inf. Sci.,* 546, pp. 420–435, Feb. 2021.

[7] R. Feng et al., "A performance-sensitive malware detection system using deep learning on mobile devices," *IEEE Trans. Inf. Forensics Secur.,* 16, pp. 1563–1578, 2021.

[8] M. U. Rashid et al., "Hybrid Android Malware Detection and Classification Using Deep Neural Networks," *International Journal of Computational Intelligence Systems,* 18, pp. 52, 2025.

[9] R. Surendran, T. Thomas and S. Emmanuel, "On existence of common malicious system call codes in Android malware families," *IEEE Trans. Reliab.,* 70(1), pp. 248–260, Mar. 2021.

[10] K. O. Elish, M. O. Elish and H. M. J. Almohri, "Lightweight, effective detection and characterization of mobile malware families," *IEEE Trans. Comput.,* 71(11), pp. 2982–2995, Nov. 2022.

[11] S. R. Davies, R. Macfarlane and W. J. Buchanan, "Differential area analysis for ransomware attack detection within mixed file datasets," *Comput. Secur.,* 108, pp. 102377, Sep. 2021.

[12] E. Odat and Q. M. Yaseen, "A novel machine learning approach for android malware detection based on the co-existence of features," *IEEE Access*, 11, pp. 15471–15484, 2023.

[13] M. Ebrahimifard, M. Abadi and F. Deldar, "RansomHunter: zero-day Android ransomware detection using graph convolutional neural networks and code embeddings," *J. Supercomput.,* 81(15), pp. 1351, 2025.

[14] T. C. Miranda, P.-F. Gimenez, J.-F. Lalande, V. V. T. Tong, and P. Wilke, "Debiasing Android malware datasets: How can I trust your results if your dataset is biased?," *IEEE Trans. Inf. Forensics Secur.,* 17, pp. 2182–2197, 2022.

[15] "A-Pot: A comprehensive Android analysis platform based on container technology," *IEEE Xplore,* https://ieeexplore.ieee.org/document/9247989 (accessed Jun. 16, 2023).

[16] T. Sharma and D. Rattan, "Characterization of Android malwares and their families," *ACM Comput. Surv.,* 57(5), pp. 1–31, 2025.

[17] R. Muthalagu, J. Malik and P. M. Pawar, "Detection and prevention of evasion attacks on machine learning models," *Expert Syst. Appl.,* 266, pp. 126044, 2025.

[18] N. I. Che Mat, N. Jamil, Y. Yusoff and M. L. Mat Kiah, "A systematic literature review on advanced persistent threat behaviors and its detection strategy," *J. Cybersecurity,* 10(1), pp. tyad023, 2024.

[19] H. Yang, Y. Wang, L. Zhang, X. Cheng, and Z. Hu, "A novel Android malware detection method with API semantics extraction," Computers & Security, 137,p p. 103651, 2024.

[20] F. Faghihi, M. Zulkernine, S. Ding "CamoDroid: An Android application analysis environment resilient against sandbox evasion," *J. Syst. Archit.,* 125, pp. 102452, Apr. 2022.

[21] B. Wu et al., "Why an Android app is classified as malware? Towards malware classification interpretation," *arXiv,* Sep. 2020.

[22] Y. Pan, X. Ge, C. Fang and Y. Fan, "A systematic literature review of Android malware detection using static analysis," *IEEE Access,* 8, pp. 116363–116379, 2020.

[23] A. Arora, S. K. Peddoju and M. Conti, "PermPair: Android malware detection using permission pairs," *IEEE Trans. Inf. Forensics Secur.,* 15, pp. 1968–1982, 2020.

[24] H. Zhang, S. Luo, Y. Zhang, L. Pan, "An efficient Android malware detection system based on method-level behavioral semantic analysis," *IEEE Xplore*, https://ieeexplore.ieee.org/document/8725475 (accessed Jun. 20, 2023).

[25] K. Xu, Y. Li, and R. H. Deng, "ICCDetector: ICC-based malware detection on Android," IEEE Trans. Inf. Forensics Secur., 11(6), pp. 1252–126.