Dual Analysis Mobile Security: A Hybrid Approach for Android Malware Detection

Shifin Vincent
Department of Computer Science and Engineering
Government Engineering College Wayanad
Kerala, India
shifinvincent@gmail.com

Riji R
Department of Computer Science and Engineering
Government Engineering College Wayanad
Kerala, India
riji@gecwyd.ac.in

***Abstract*—The rapid proliferation of the Android operating system has made it a primary target for sophisticated mobile malware. Modern malicious applications utilize advanced evasion techniques, such as code obfuscation and dynamic payload loading, rendering singular static or dynamic analysis methodologies insufficient. Static methods are frequently circumvented by encryption, while isolated dynamic analysis remains computationally expensive and prone to sandbox evasion. This paper presents a comprehensive Hybrid Analysis Approach that mitigates these limitations by merging structural telemetry with dynamic behavioral logs into a unified, automated detection pipeline. High-risk permissions and intent filters are extracted from the Android manifest and fused with runtime execution traces. These hybrid feature vectors are evaluated against a suite of machine learning classifiers, specifically Random Forest, XGBoost and LightGBM. Experimental results demonstrate that leveraging this fused architecture and soft-voting ensemble significantly improves classification accuracy and minimizes false positives, providing a highly dependable and resilient framework for identifying zero-day Android threats.**

***Index Terms*—Android Malware, Hybrid Analysis, Ensemble Learning, Feature Fusion, Static Analysis, Dynamic Analysis, Mobile Security, Machine Learning.**

I. Introduction

The global ubiquity of smartphones has established the Android operating system as the dominant mobile platform. Its open-source architecture and massive user base have consequently led to an exponential increase in malicious applications engineered to target consumer devices [9]. Modern mobile malware—encompassing banking Trojans, ransomware, and covert spyware—employs highly sophisticated evasion techniques to bypass the standard signature-based security mechanisms deployed by traditional centralized app stores.

Security researchers traditionally rely on singular analysis paradigms to identify threats, but these methods possess critical blind spots when operating in isolation. Static analysis involves inspecting an application's structural files, such as the *AndroidManifest.xml* and decompiled bytecode, without execution. While computationally efficient, static scanning struggles to identify threats when the malicious payload is hidden using packers, encryption, or advanced code obfuscation techniques. Conversely, dynamic analysis monitors live runtime behavior. While highly effective at capturing actual execution paths, local dynamic sandboxing is computationally expensive, time-consuming, and prone to missing malicious behavior hidden behind environment-aware evasions that detect the local virtual machine.

The problem is stated as to develop a robust Hybrid Analysis Approach that mitigates these limitations by merging both methodologies into an automated detection pipeline. By systematically fusing structural vulnerabilities with dynamic execution latency, this research aims to maximize detection rates while maintaining operational efficiency.

To address the limitations of local sandbox evasion, this framework leverages a dual-pipeline approach. A local Dockerized instance of the Mobile Security Framework (MobSF) is utilized for rapid static structural extraction, while dynamic behavioral telemetry is offloaded to the cloud via the VirusTotal API. The resulting fused feature vector is evaluated using an advanced soft-voting machine learning ensemble, integrating gradient-boosted and tree-based algorithms to isolate complex, non-linear malicious patterns.

The primary contributions of this paper are as follows:

1) The design of an automated hybrid extraction pipeline utilizing a local Dockerized MobSF environment for static structural data and the VirusTotal API for rich, cloud-based dynamic behavioral telemetry.
2) The construction of a robust, fused 30-dimensional feature space that maps the correlation between requested structural permissions and actual executed behaviors.
3) The implementation of a heterogeneous soft-voting ensemble—comprising Random Forest, XGBoost, and LightGBM—that significantly improves classification accuracy while minimizing the variance of individual models.

The remainder of this paper is organized as follows: Section II reviews related literature in mobile malware detection. Section III details the proposed hybrid methodology, feature fusion, and ensemble architecture. Section IV presents the experimental setup, evaluation metrics, and performance results. Finally, Section V concludes the paper and discusses avenues for future work.

II. Related Work

The existing literature highlights a distinct methodological divide in mobile malware detection, broadly categorized into

static, dynamic, and hybrid approaches. This section reviews recent advancements and identifies the existing gaps that the proposed soft-voting ensemble framework addresses.

A. Static Analysis Approaches

Early research heavily favored static analysis due to its low computational overhead and rapid processing speeds. Studies leveraging decompilation tools like Apktool have successfully established efficient dataset generation pipelines. For instance, classic frameworks like DREBIN [3] demonstrated high accuracy by extracting thousands of structural features, including requested permissions, API calls, and network addresses from the *AndroidManifest.xml* and decompiled bytecode. However, these frameworks remain structurally dependent on static patterns. Modern malware authors increasingly utilize obfuscation, Java reflection, and native code execution via the Java Native Interface (JNI) to hide their payloads, rendering purely static scanners highly susceptible to false negatives.

B. Dynamic Analysis Approaches

To counter obfuscation, dynamic analysis focuses on application execution and behavioral monitoring. Frameworks such as TaintDroid [4] and isolated sandbox environments provide highly effective zero-day threat detection by tracking information flow, network traffic, SMS broadcasts, and file system modifications in real-time. Despite their accuracy in capturing actual execution paths, these systems possess significant drawbacks. Local dynamic sandboxes demand heavy computational resources and suffer from high latency during batch processing. Furthermore, sophisticated modern malware often employs environment-aware logic bombs that detect the presence of local virtual machines, halting malicious behavior to evade detection.

C. Hybrid and Ensemble Frameworks

Recent advancements have sought to combine static structural indicators with dynamic execution traces to mitigate the weaknesses of singular analysis. Various hybrid frameworks fuse structural and behavioral telemetry, applying machine learning to classify the resulting high-dimensional data [1], [2]. While these systems show improved threat sensitivity, many current implementations face architectural hurdles. They often rely on computationally heavy local sandboxing for dynamic data, or they utilize singular, high-variance classification models (such as standalone decision trees) that struggle to generalize across unseen malware families [5].

This paper addresses these specific gaps. By offloading dynamic behavioral extraction to the cloud via the VirusTotal API, the proposed framework bypasses local sandbox evasion techniques. Furthermore, rather than relying on a single classifier, this research introduces a heterogeneous soft-voting ensemble combining Random Forest, XGBoost, and LightGBM. This gradient-boosted architecture captures both linear structural vulnerabilities and non-linear dynamic API call sequences, achieving robust accuracy while minimizing the variance observed in prior studies.

III. Proposed Methodology

The proposed Hybrid Analysis framework operates by processing raw Android Application Packages (APKs) through a synchronized, multi-stage pipeline designed to mitigate standard signature evasion and sandbox detection strategies. The end-to-end system description transitions from parallel data ingestion to programmatic feature engineering, culminating in a robust machine learning ensemble classification.

A. System Description and Architecture

The core system architecture is split into a distinct parallel extraction topology. Instead of analyzing structural indicators or operational runtime execution in isolation, the architecture ingests a target APK file and routes it to simultaneous local static and cloud-based dynamic analysis engines. Figure 1 illustrates the high-level flow diagram of the integrated feature engineering runtime pipeline.

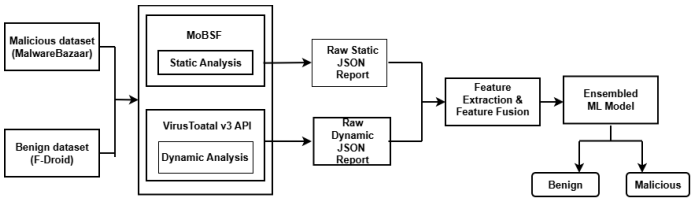

Fig. 1. High-level architecture of the dual analysis pipeline integrating MobSF and VirusTotal.

B. Static Analysis Pipeline via Dockerized MobSF

The structural evaluation layer focuses on identifying malicious structural footprints embedded within the application package without executing its compiled bytecode.

- **Environment Isolation:** To guarantee stable, automated deconstruction, the framework deploys a containerized instance of the Mobile Security Framework (MobSF) running on a local Docker container environment.
- **Structural Telemetry Extraction:** The input APK file is programmatically uploaded via HTTP POST requests using the MobSF REST API gateway. The local decompilation process systematically unpacks the resource components and parses the underlying *AndroidManifest.xml* file.
- **Feature Identification:** The static parsing layer targets explicit structural features, isolating requested permissions and defined intent filters. This layer specifically isolates malicious permission pairings and intent configuration traits that indicate background persistency or high-risk inter-process communication capabilities. The output is structured as a comprehensive raw static JSON report.

C. Dynamic Analysis Pipeline via VirusTotal API

To mitigate the high latency and processing bottlenecks associated with running local host sandboxes (such as CuckooDroid or DroidBox), the framework offloads dynamic behavioral extraction to a scalable cloud architecture.

- **Evasion Countermeasures:** Advanced mobile threats frequently implement environment-aware logic bombs that detect local virtual machines or emulators, intentionally suppressing malicious payloads upon sensing localized sandboxes. Offloading extraction circumvents this restriction.
- **Behavioral Logging Capture:** The framework executes a custom Python script that calculates the unique SHA-256 cryptographic hash of the input application file. This hash is used to programmatically query the cloud-based VirusTotal v3 API interface.
- **Telemetry Retrieval:** The query retrieves rich, pre-computed historical dynamic execution traces. The process downloads an unstructured raw dynamic JSON report containing multi-perspective telemetry, including live network request volumes, external Domain Name System (DNS) query fields, file system modifications, and dynamic API call tracing loops.

D. Feature Extraction and Tabular Feature Fusion

Prior to feature fusion, the raw application datasets were curated from distinct sources to ensure a balanced and realistic evaluation environment. The benign dataset was sourced from open-source repositories such as F-Droid, while the malicious dataset was aggregated from recognized threat intelligence platforms, including MalwareBazaar.

Once the parallel analysis reports are successfully compiled, they are passed to an automated parsing and engineering module [8]. A custom Python script structures the massive, unstructured text logs from both the MobSF static report and the VirusTotal dynamic report.

Categorical features—such as raw requested permissions—are quantized and mapped into a structured binary-encoded integer dataset [10], where a 1 represents a configuration trait's presence and a 0 indicates its absence. Continuous dynamic indicators, such as contacted IP addresses and net-operation volumes, are mathematically aligned into numerical fields. The module fuses these elements, along with engineered high-risk twosome permission combinations (such as the simultaneous occurrence of `READ_SMS` and `INTERNET`), into a single, balanced 30-dimensional feature vector.

E. Heterogeneous Soft-Voting Ensemble Implementation

The finalized 30-dimensional feature vector is routed directly into a heterogeneous machine learning meta-classifier. To maximize generalization performance and mitigate the high variance typical of individual predictive engines on tabular security datasets, the decision engine aggregates three mathematically synergistic classifiers:

1) Random Forest: A tree-based ensemble classifier that constructs independent decision trees using bootstrap aggregating (bagging) [5]. This model creates an exceptionally robust, low-variance classification baseline that successfully accommodates noisy or missing fields across the hybrid feature matrix without overfitting.

2) Extreme Gradient Boosting (XGBoost): A highly efficient implementation of gradient boosted decision trees [6]. XGBoost builds trees sequentially rather than concurrently, with each subsequent model focusing entirely on correcting the residual errors introduced by its predecessor. Regularization parameters prevent the model from misinterpreting erratic behavioral variations.

3) Light Gradient Boosting Machine (LightGBM): A gradient boosting architecture optimized for speed and high-dimensional spaces through its leaf-wise tree growth strategy [7]. LightGBM splits nodes at the leaf level based on maximum gain rather than moving horizontally level-by-level, discovering complex, hidden classification splits in tabular binary fields at scale.

4) Soft-Voting Mathematical Framework: Rather than deploying a hard-voting model that depends on a simple majority count, the engineered framework utilizes soft-voting logic. This protocol considers the exact mathematical probability score (confidence percentage) issued by each underlying base algorithm. For any incoming unseen APK file, the meta-classifier calculates the simple average of the individual probability outcomes. The final ensemble prediction probability P is expressed as:

$$P = \frac{P_{\text{RandomForest}} + P_{\text{XGBoost}} + P_{\text{LightGBM}}}{3} \tag{1}$$

If the calculated average probability score P is greater than or equal to a 0.5 classification threshold, the pipeline flags the target application as malicious, triggering subsequent alert logs and isolating the file. If P falls below 0.5, the application is certified as benign and deemed safe for execution.

IV. Experimental Setup and Results

To validate the efficacy of the proposed hybrid extraction and soft-voting ensemble framework, a series of controlled experiments were conducted. This section details the environmental configuration, evaluation metrics, and the comparative performance analysis of the deployed machine learning models.

A. Implementation Environment

The evaluation framework was developed utilizing Python 3 and the Scikit-Learn ecosystem, integrating specialized libraries including XGBoost and LightGBM for advanced gradient boosting.

The experimental testbed was hosted on a Windows 11 machine equipped with an Intel Core i5 processor and 8GB of RAM. To safely process potential malicious payloads, the Dockerized MobSF instance and the Python orchestration scripts were executed within an isolated Ubuntu virtual machine deployed via Oracle VirtualBox.

To guarantee operational stability and strict reproducibility, the underlying classification models were locked to specific hyperparameters during training:

- **Number of Estimators:** Set to 150 across all models to balance predictive accuracy with fast real-time inference speeds.

- **Learning Rate:** Configured to 0.1 for the XGBoost and LightGBM models for smooth, continuous error correction.
- **Random State:** Locked to 42 across all data splits and algorithms to ensure deterministic execution.

B. Evaluation Metrics

The models were evaluated using standard quantitative classification metrics derived from the confusion matrix: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). In this context, a "Positive" classification indicates the successful detection of malware.

- **Accuracy:** The overall percentage of correctly classified benign and malicious applications.
- **Precision:** The ratio of correctly predicted malware to the total predicted malware, indicating the system's resistance to false alarms.
- **Recall:** The ratio of correctly predicted malware to all actual malware samples in the dataset, representing the system's threat sensitivity.
- **F1-Score:** The harmonic mean of Precision and Recall, providing a single metric for stable classification performance.

C. Performance Analysis

The finalized 30-dimensional hybrid feature dataset was partitioned into an 80/20 train-test split, providing a robust training foundation while preserving a statistically significant evaluation set. The framework was evaluated on an unseen testing split of 89 applications (47 Benign, 42 Malicious).

Table I details the comparative performance of the individual base classifiers against the proposed soft-voting ensemble.

TABLE I
CLASSIFIER PERFORMANCE METRICS ON THE FUSED DATASET

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	86.52%	87.00%	87.00%	87.00%
XGBoost	89.89%	90.00%	90.00%	90.00%
LightGBM	88.76%	89.00%	89.00%	89.00%
Soft-Voting Ensemble	**89.89%**	**90.00%**	**90.00%**	**90.00%**

The results demonstrate that while the individual tree-based and gradient-boosted models performed admirably, they exhibited slight variances in threat sensitivity. The soft-voting ensemble successfully mitigated this individual model variance by mathematically averaging their confidence probabilities, resulting in highly stable predictions.

The ensemble architecture achieved the highest overall accuracy at 89.89% and a remarkable 90.00% recall rate. This high recall is particularly critical in cybersecurity contexts, as it ensures that zero-day threats and obfuscated malware rarely slip past the detection threshold.

A deeper examination of the minority misclassifications provides valuable context regarding the operational boundaries of the hybrid approach. False positives—where benign applications were inadvertently flagged as malicious—predominantly occurred in legitimate applications exhibiting aggressive, adware-like behaviors, such as requesting excessive background permissions coupled with high-volume external network telemetry. Conversely, the minimal false negatives were largely attributed to highly sophisticated malware variants employing delayed execution logic (logic bombs) designed to outlast the predefined behavioral monitoring window of the dynamic analysis pipeline.

Despite the mathematical complexity of running three concurrent classifiers, the soft-voting ensemble maintained highly efficient inference times. The parallel tree-building architecture of Random Forest, combined with the optimized leaf-wise growth strategy of LightGBM, ensured that once the feature vector was engineered, the actual classification occurred in near real-time, validating its potential for future on-device deployment.

Furthermore, Figure 2 presents the confusion matrix for the soft-voting ensemble on the unseen test dataset, detailing the precise distribution of correct and incorrect classifications across the benign and malicious samples.

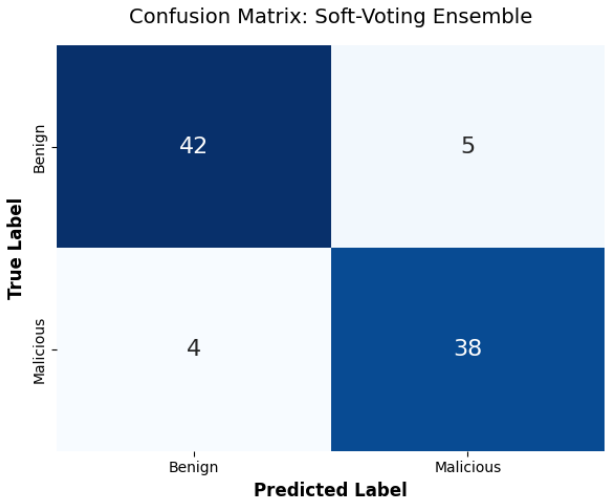

Fig. 2. Confusion Matrix detailing the classification performance of the Soft-Voting Ensemble on the unseen test dataset.

V. CONCLUSION AND FUTURE WORK

This research successfully addresses the critical blind spots inherent in singular Android malware analysis by engineering a fully automated, dual-pipeline hybrid detection framework. By synthesizing structural vulnerabilities—extracted locally via a Dockerized Mobile Security Framework (MobSF)—with dynamic execution traces gathered through the cloud-based VirusTotal API, the proposed architecture effectively circumvents modern evasion techniques such as code obfuscation and local sandbox detection.

The engineered 30-dimensional feature vector was evaluated using a heterogeneous soft-voting ensemble, integrating Random Forest, XGBoost, and LightGBM classifiers. The results demonstrate that this gradient-boosted ensemble mitigates the high variance often observed in standalone models, achieving

a robust classification accuracy of 89.89% and a high recall rate of 90.00%. These metrics confirm the system's high threat sensitivity, making it highly capable of isolating complex, zero-day malicious patterns while maintaining a low false-positive rate.

Future work, slated for development through 2026, will focus on two primary objectives. First, the automated data collection pipeline will be expanded to ingest a broader array of emerging malware families, ensuring the ensemble remains resilient against evolving threat landscapes. Second, research will be directed toward model quantization and pruning techniques. By compressing the gradient-boosted architecture into a lightweight, resource-efficient format, this framework can transition from a server-side analysis tool into an on-device, real-time threat detection engine deployed directly on consumer Android smartphones.

REFERENCES

[1] F. Taher, O. AlFandi, M. A. Kfairy, H. Al Hamadi, and S. Alrabaee, "DroidDetectMW: A Hybrid Intelligent Model for Android Malware Detection," *MDPI Applied Sciences*, 2023.

[2] Q. Xu, D. Zhao, S. Yang, L. Xu, and X. Li, "Android Malware Detection Based on Behavioral-Level Features with Graph Convolutional Networks," *Electronics*, vol. 12, no. 23, p. 4817, 2023.

[3] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.

[4] W. Enck, P. Gilbert, B. G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 2, pp. 1–29, 2014.

[5] S. Y. Yerima, S. Sezer, and G. McWilliams, "Analysis of decision tree based ensembles for Android malware detection," in *10th International Conference on Network and Service Management (CNSM)*, IEEE, 2014, pp. 323–328.

[6] 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*, 2016, pp. 785–794.

[7] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[8] Y. Aafer, W. Du, and H. Yin, "DroidAPIMiner: Mining API-level features for robust malware detection in Android," in *International Conference on Security and Privacy in Communication Systems*, Springer, 2013, pp. 86–103.

[9] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A Review of Android Malware Detection Approaches Based on Machine Learning," *IEEE Access*, vol. 8, pp. 124579–124607, 2020.

[10] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, "A review on feature selection in mobile malware detection," *Digital Investigation*, vol. 13, pp. 22–37, 2015.