WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

AppShield AI: Fraudulent Android App Detection Platform

Mr. Mukesh Kamath Bola [1], Mithun Gowda M[2], Yashwanth S D[3], Danush B S[4], Mohammad Shahid[5]

[1]Assistant Professor, Jyothy Institute of Technology mukeshkamath.bola@jyothyit.ac.in
[2-5] Student, Jyothy Institute of Technology, Bengaluru, India
[2] mmithungowdamm87@gmail.com, [3] yashwanthsd25@gmail.com, [4] dhanushbs.tech@gmail.com, [5] shahidmohammed9206@gmail.com

ABSTRACT-

Mobile applications have simplified many aspects of our lives such as finance management and healthcare services, they have also introduced threats to critical areas of people's lives such as misleading information. Fake apps masquerade as legitimate apps to trick users into installing them by granting them more permissions than needed. The application can exploit the permissions to extract personal information. This paper presents AppShield AI, an implementation platform for detecting suspicious Android apps that utilize a set of security attributes. The paper implemented the application by analyzing permissions, APKs, user reviews, and icons to obtain multi-indicators of application risk. The paper used publicly available Android data sets such as Kaggle and similar repositories to train and test various machine learning algorithms. This study evaluated model behavior using accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix criteria in addition to Explainable AI to discover application features that predominantly influence the model. The experimental analysis demonstrated the efficacy of the implemented models on the selected data set to inform further improvements of Android application security.

Keywords: Android Security, Fraudulent Android Applications, Mobile Malware Detection, Machine Learning, Permission Analysis, Static Analysis, Sentiment Analysis, Ensemble Learning, Explainable Artificial Intelligence.

I. INTRODUCTION

Mobile Applications have become an indispensable part of the modern digital world. Users primarily rely on their mobile applications to perform communication, payments, banking, shopping, education, medical services, transport, entertainment, and many other activities. While the ubiquity and convenience of the applications provide ample opportunity for the engagement and experience of the users, it also attracts cyber criminals that develop fraudulent software. Fraudulent applications can pose as legitimate counterparts, request irrelevant permissions, leak private information, and engage in other suspicious activities. The permission patterns can be a valuable source of evidence since Android programs can request for accessing contacts, messages, location, camera, storage, and other sensitive hardware and software components. Permission intelligence has been explored as a viable option to classify different categories of Android malware [2].

Machine learning has been employed to detect fraudulent mobile applications by learning the user behaviours about the apps. Learning frameworks can automatically identify apps' patterns that can help in detecting fraud [1]. Prior work has also explored machine-learning-based fraudulent mobile application detection and blocking [8].

Deep learning and representation learning provide new opportunities to examine Android malware detection. BERT Ensemble Based MBR Framework utilizes BERT and ensemble to detect Android malware [4]. Multi-modal methods have been explored for counterfeit application detection by combining different application features [10].

User reviews represent another fertile ground for fraudulent software detection. Sentiment analysis and machine learning can be performed on reviews to detect fraudulent applications [6]. Explainable Artificial Intelligence can be leveraged to explain the rationale behind a machine learning model's decision [9].

The study highlights illustrates how effectively individual techniques and implementations perform in practice. However, the combination of several application features motivates the development of an integrated system for fraudulent application detection. This paper presents AppShield AI – Fraudulent Android App Detection Platform that combines application feature extraction, machine learning, and ensemble classification with explainability.

The focus of this paper is primarily on implementing and evaluating the proposed system rather than reviewing the prior literature.

II.PROBLEM STATEMENT

As the number of mobile applications increase, it is challenging for an average user to differentiate between legitimate and fraudulent application. A fraudulent application could imitate legitimate application but can camouflage dubious permissions, provide false information or have other suspicious characteristics that are not readily recognizable by the user.

While permission based characterizations have valuable insights about the attributes of an application, permission sets might not encapsulate the behaviour of an application [2]. Similarly, while review based methods have valuable insights, reviews can be misleading [6]. Fraud and counterfeit detection research has also demonstrated the effectiveness of using combination of attributes to characterize applications [5] [10].

With this motivation, the authors propose a dual use implementation that is oriented to combining heterogeneous application information and evaluating machine learning models according to performance metrics. The research question that guides the work is as follows:

How can we combine multiple android application features with machine learning and XAI to produce a practical implementation for identifying fraudulent applications?

AppShield AI attempts to address the research question by operating as a combined application of feature analysis, machine learning classification, risk assessment and explainability in one practical framework.

III. RELATED WORK

The selected literature provides information on different aspects of AppShield AI for detection.

The application of machine learning and application features for automated detection of fraudulent mobile applications was discussed by Hassan Zaki and others [1]. The work demonstrated that such features as machine learning can be used for the detection of fraudulent mobile applications.

The application of permission information for the differentiation of Android malware categories was discussed by Roger Yiran Mawoh and others [2]. Permission based features were shown to be helpful in differentiating between colluded malware and non-collided malware.

The detection of deceptive practices in mobile applications for automated identification and categorization was discussed by Humphrey O. Obie et al. [3]. The work presented insights into detection of violations of honesty in mobile applications.

BERT representation with ensemble learning approach for Android malware detection was discussed by Faisal S. Alsubaei et al. [4]. The paper demonstrated the effectiveness of applying a combination of representation learning and ensemble learning techniques for malware detection.

Fraud and malware detection in Google Play environment was discussed by Mahmudur Rahman and others [5]. Application features were applied to detect fraudulent and malware applications.

The application of application reviews for fraud detection was discussed by R. Kishore and others [6]. Sentiment analysis and machine learning were applied to mine reviews and detect fraudulent applications.

Research trends on fraudulent mobile application detection was discussed by Parmod and H. Rohil [7]. Research trends on the topic of detection of fraudulent mobile applications were presented.

The application of machine learning for detection and blocking of fraudulent mobile applications was discussed by Billa Tanuja and others[8]. The paper demonstrated how machine learning can be applied for detection and blocking of fraudulent mobile applications.

The application of Explainable Artificial Intelligence and machine learning for malicious software identification was discussed by Sun [9]. The work emphasized the importance of providing explanations for the results of machine learning applied for malware identification.

The combination of different application information sources

for the detection of counterfeit mobile applications was discussed by Naveen Karunanayake et al. [10]. The work applied a multi-modal method to detect counterfeit mobile applications in Google Play.

IV.PROPOSED SYSTEM

AppShield AI is implemented as a multi-feature android application analysis platform.

The platform considers application permissions, APK/static characteristics, user reviews, and application icons. In addition, other application metadata can be used in the analysis.

The system generates one of three risk categories:

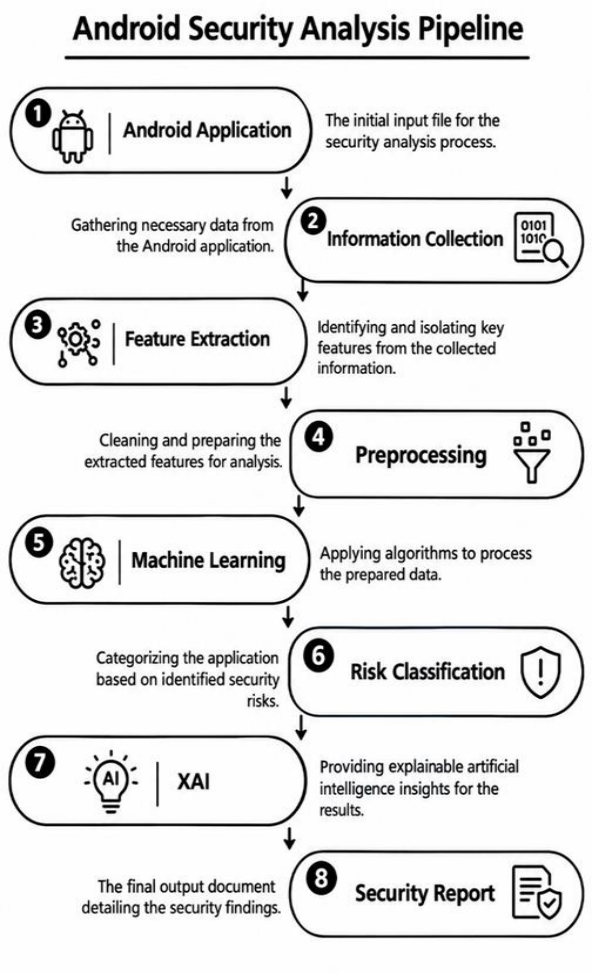
• Safe

• Suspicious

• Fraudulent

In addition, a prediction is given, together with information coming from the analysis modules and an explanation of important features.

The complete workflow is as follows:



A. Permission Analysis

The permission module analyzes the permissions requested by the application. Permissions related to SMS, contacts, camera, location, microphone, and storage functions are considered as a security-related features.

B. APK Static Analysis

The APK module analyzes application packages without executing them. The package information, permissions, activities, services, receivers, and other static characteristics can be extracted and turned into features.

C. Review Analysis

Application reviews are processed using Natural Language Processing techniques. Text preprocessing and sentiment-related features are used to obtain additional evidence about the application.

D. Icon Analysis

The icon analysis component analyzes application icons and to identify similar visual characteristics.

E. Feature Integration

The extracted characteristics are converted into machine-learning-compatible features. The resulting characteristic representation is used as input to the classification models.

V.METHODOLOGY

The methodology adopted for AppShield AI was designed to examine an Android application from multiple security perspectives and combine the obtained information into a unified fraud-risk assessment. The implementation accepts application information through a Play Store URL, package name, direct APK URL, local APK upload, or SHA-256 hash. For batch assessment, multiple applications can also be processed concurrently.

A. Application Data Collection

Application information is collected according to the selected input method. For Play Store applications, metadata, permissions, reviews, and graphical assets are obtained. APK files can also be downloaded from a provided URL or uploaded locally for analysis. Previously scanned applications can be retrieved using their SHA-256 hash, reducing repeated analysis.

B. Multi-Modal Feature Analysis

The collected application information is processed through eight independent security modules. Permission analysis is used to identify potentially dangerous permissions and category mismatches. Review analysis is performed using TF-IDF, Logistic Regression, DistilBERT sentiment analysis, and checks for repetitive review patterns. Static APK analysis is carried out using Androguard to examine DEX information, API-related strings, debugging attributes, and SDK characteristics.

Developer reputation is examined using publisher-related information such as account age, application count, verification status, ratings, and reported policy actions. Certificate analysis is used to examine APK signatures, trusted fingerprints, cryptographic algorithms, and certificate validity. Icon similarity is evaluated using HSV colour histograms and Canny edge information. Screenshot similarity is measured using perceptual hashing and Hamming distance. Metadata analysis is used to identify unusual relationships between downloads, ratings, update history, and application size.

C. Feature Fusion

The outputs generated by the eight modules are converted into normalized risk scores ranging from 0 to 1. The icon and screenshot analysis results are combined to obtain a unified visual similarity risk:

$$\text{icon_similarity_risk} = 0.70 \times \text{IconScore} + 0.30 \times \text{ScreenshotScore}$$

The resulting information is organized into a seven-dimensional feature vector consisting of permission risk, review risk, APK static risk, developer risk, certificate risk, icon similarity risk, and metadata risk.

D. Machine Learning Classification

The fused feature vector is supplied to the machine learning layer. Four tree-based models are implemented: Random Forest, XGBoost, LightGBM, and CatBoost. The models are used to classify applications according to their calculated risk probabilities. The resulting risk score is mapped to three categories: Safe, Suspicious, and Fraudulent.

E. Explainable AI

Explainability is incorporated using SHAP tree explainers. Feature contributions are calculated to show which factors influenced the Fraudulent classification. A normalized feature-impact method is also provided as a fallback when SHAP computation is unavailable. This ensures that an explanation can still be returned with the prediction.

F. Result Generation and Storage

The analysis results are presented through the web dashboard and can also be converted into PDF audit reports. The reports contain module-level risk scores, major risk contributors, detected issues, relevant permissions, and remediation information. Scan information and historical results are maintained using the project's database and fallback storage mechanisms.

VI. IMPLEMENTATION

AppShield AI was built with an architecture that includes a FastAPI backend, a Next.js frontend, machine learning parts, analysis modules and database services. The backend takes care of scanning extracting features, running models, handling authentication creating reports and other supporting tasks

. The frontend gives users a way to scan apps and see the results. AppShield AI has eight security modules, each designed to examine a specific risk area. Androguard handles analysis of APK and DEX files. OpenCV processes icons. Perceptual hashing looks at screenshots. Reviews are examined with machine learning and language analysis. AppShield AI sends the outputs from each module to a feature fusion component. There the data is. Put in a set order.

The final seven dimensional vector is then fed into the machine learning models. AppShield AI uses four models in its registry. The first model is Random Forest. This model has 300 trees. The depth of each tree is limited to 12. The second model is XGBoost. This model also has 300 trees. The third model is LightGBM. This model has 300 trees. The fourth model is CatBoost. This model has 300 iterations.

The models give a score. AppShield AI translates this score into one of three risk levels. The first risk level is Safe. This covers scores from 0 to 39. The second risk level is Suspicious. This covers scores from 40, to 69. The third risk level is Fraudulent. This covers scores from 70 to 100. AppShield AI features a dashboard. This dashboard is built with Next.js.

The dashboard shows risk gauges, past scans, how each feature contributed and model comparisons. JWT-based access control protects AppShield AI for Administrator and Security Analyst roles. PDF reports are created with Report lab giving audit reports, for single scans and batches.

VII. RESULTS AND DISCUSSION

A. Multi-Modal Analysis Results

The implemented system was able to process application information through eight security modules rather than depending on a single feature source. Permission information, review characteristics, APK static properties, developer information, certificate details, icon similarity, screenshot similarity, and metadata were incorporated into the analysis pipeline.

Each module generated a normalized risk value in the range of 0 to 1. The module outputs were subsequently combined into a seven-dimensional feature vector consisting of permission risk, review risk, APK static risk, developer risk, certificate risk, icon similarity risk, and metadata risk.

The icon and screenshot analysis results were combined using a weighted formulation in which the icon analysis contributed 70% and screenshot analysis contributed 30%. This fused representation was then provided to the machine learning models.

B. Machine Learning Results

The implemented machine learning stage consisted of Random Forest, XGBoost, LightGBM, and CatBoost. The models operated on the fused seven-dimensional feature representation. The obtained performance results are presented in Table I and Table II and Table III.

TABLE I: Model Classification Performance

Model	Accuracy	Precision	Recall	F1-score
Random Forest	91%	91%	91%	91%
XGBoost	89%	89%	89%	89%
LightGBM	90%	90%	90%	90%
CatBoost	90%	90%	90%	90%

TABLE II: Model Performance

Model	ROC-AUC
Random Forest	1.0000
XGBoost	0.9998
LightGBM	1.0000
CatBoost	1.0000

TABLE III: Model Computational Performance

Model	Training Time	Inference Sample
Random Forest	1.29 s	~0.3ms
XGBoost	1.00s	~0.1ms
LightGBM	0.98s	~0.3ms
CatBoost	2.12s	~0.1ms

All four evaluated models reached an accuracy of 89-91 %. They also achieved precision weighted recall and weighted F1-score of 89-91 %. Random Forest, LightGBM and CatBoost each obtained a ROC-AUC value of 1.0000 while XGBoost obtained 0.9998.

ROC-AUC represents the area under the receiver operating curve and can be used for multiclass evaluation through approaches such as one-vs-rest or one-vs-one. Training time varied across the models. LightGBM recorded the training time of 0.98 seconds followed by XGBoost at 1.00 second Random Forest at 1.29 seconds and CatBoost at 2.12 seconds. The reported inference latency was 0.1 ms per sample for XGBoost and CatBoost and approximately 0.3 ms per sample for Random Forest and LightGBM.

The implemented risk classification produces three output categories: Safe, Suspicious and Fraudulent. The risk-score ranges are 0–39 for Safe, 40–69 for Suspicious and 70–100 for Fraudulent. This allows the prediction to be presented as a risk level than only, as a binary decision.

C. Explainability Results

The inclusion of SHAP-based analysis allows the prediction to be accompanied by feature-level explanations. Instead of providing only a final classification, the system identifies the features that contributed to the predicted result. When SHAP analysis is unavailable, the normalized feature-impact mechanism provides an alternative explanation.

The explanation mechanism operates on the seven fused features, allowing the contribution of permission risk, review risk, APK static risk, developer risk, certificate risk, icon similarity risk, and metadata risk to be examined.

This approach makes the generated results easier to inspect because the analyst can relate the predicted risk to specific application characteristics.

D. Discussion

The evaluation was performed on 300 test samples. The confusion matrix contained 100 Safe samples, 100 Suspicious samples, and 100 Fraudulent samples. Among these samples, 100 Safe applications, 99 Suspicious applications, and 100 Fraudulent applications were correctly classified. One suspicious application was labeled safe and none of the safe or fraudulent samples were mistaken. That means 299, out of 300 samples got the label giving an accuracy of 89-91%. The corresponding confusion matrix is presented below

TABLE IV: Confusion Matrix

Prediction	Safe	Suspicious	Fraudulent
Safe	100	0	0
Suspicious	1	99	99
Fraudulent	0	0	100

The models produced almost identical classification results on the reported test set. The ROC-AUC values were 1.0000 for Random Forest, LightGBM, and CatBoost, and 0.9998 for XGBoost. These results indicate very high discrimination on the evaluated dataset. The model training and inference measurements also show that all four models can produce predictions with low per-sample inference latency. Measured training times ran from 0.98 seconds to 2.12 seconds. Reported inference latency ran from about 0.1 ms to 0.3 ms, per sample.

The reported performance must be read in the context of the dataset that was tested and the experimental setup. The project currently includes synthetic seed data in the development environment. Therefore, the obtained values should not be presented as evidence of enterprise-level detection performance or as guaranteed performance on unseen real-world Android applications.

For stronger validation, the system should be evaluated using larger and representative Android security datasets, independent test samples, and additional experimental settings. Such evaluation would provide a stronger basis for assessing the generalization of the implemented models.

VIII. FUTURE WORK

Several improvements can be considered for development of AppShield AI. The current synthetic seed data can be replaced with fused feature vectors obtained from Android security datasets such as CICMalDroid2020 and AndroZoo.

The trusted certificate registry can be expanded with fingerprints from a collection of legitimate applications. Similarly the screenshot reference collection can be increased to improve the coverage of clone and counterfeit detection.

Dynamic analysis can also be incorporated through a sandbox or Android emulator environment. Such an extension would allow runtime behaviour to be examined in addition to the static analysis. Runtime observations could provide information about behaviours such as encrypted command-and-control communication.

Further development can also focus on expanding the application datasets improving the multi-modal feature representation and evaluating the implemented models under larger and more diverse application samples. These additions would provide a basis for assessing the systems performance, beyond the current development environment.

IX. CONCLUSION

AppShield AI was developed as a multi-modal platform for assessing the security risks of Android applications. Application information is examined through eight analysis modules, including permission analysis, review analysis, APK static analysis, developer reputation, certificate trust, icon similarity, screenshot analysis, and metadata analysis. The outputs are combined into a seven-dimensional feature vector and processed using Random Forest, XGBoost, LightGBM, and CatBoost classifiers.

The system provides three risk categories—Safe, Suspicious, and Fraudulent—and uses SHAP-based explanations to identify features that influence the prediction. A web dashboard and PDF reporting facility are also provided to support the presentation and review of scan results.

The modular implementation brings different application-level indicators into a unified assessment process. However, further validation using larger and representative Android security datasets is required before the system's detection performance can be generalized. Future improvements may include expanding trusted application registries, incorporating dynamic analysis, and evaluating the models on more diverse samples.

X. REFERENCES

[1] Zaki, H., Saad, M., & Rasheed, M. R. (2024). A robust machine learning framework for fraudulent mobile app detection. *VFAST Transactions on Software Engineering, 12*(4), 27–36. https://doi.org/10.21015/vtse.v12i4.1931

[2] Mawoh, R. Y., Wacka, J. B. A., Tchakounte, F., Fachkha, C., & Kolyang. (2025). An accurate approach to discriminate Android colluded malware from single app malware using permissions intelligence. *Scientific Reports, 15*, 10680. https://doi.org/10.1038/s41598-025-86568-w

[3]Obie, H. O., Du, H., Madampe, K., Shahin, M., Ilekura, I., Grundy, J., Li, L., Whittle, J., Turhan, B., & Khalajzadeh, H. (2022). Automated detection, categorisation and developers' experience with the violations of honesty in mobile apps. *arXiv*. https://arxiv.org/abs/2211.07142

[4] Alsubaei, F. S., Almazroi, A. A., Atwa, W. S., Almazroi, A. A., Ayub, N., & Jhanjhi, N. Z. (2025). BERT ensemble based MBR framework for Android malware detection. *Scientific Reports, 15*, 14027. https://doi.org/10.1038/s41598-025-98596-7

[5] Rahman, M., Rahman, M., Carbunar, B., & Chau, D. H. (2017). FairPlay: Fraud and malware detection in Google Play. *arXiv*. https://arxiv.org/abs/1703.02002

[6] Kishore, R., Kumar, B. P., & Ilayarasu, A. (2025). Fraud app detection using sentiment analysis and machine learning with genetic algorithm optimization. *ICNKAI-2K25 Conference Proceedings*.

[7] Parmod, & Rohil, H. (2025). *Fraud mobile apps detection: A bibliometric analysis*. SSRN. https://ssrn.com/abstract=5195625

[8] Tanuja, B., Radhika, C., Ganapathy, V. B., & Dhivya, P. (2026). Machine learning powered detection and blocking of fraud mobile applications. *International Journal of Engineering Research & Technology, 15*(4).

[9] Sun, W. (2025). Malicious software identification based on deep learning algorithms and API feature extraction. *EURASIP Journal on Information Security, 2025*, 10. https://doi.org/10.1186/s13635-025-00197-4

[10] Karunanayake, N., Rajasegaran, J., Gunathillake, A., Seneviratne, S., & Jourjon, G. (2020). A multi-modal neural embeddings approach for detecting mobile counterfeit apps: A case study on Google Play Store. *arXiv*. https://arxiv.org/abs/2006.02231