iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** IV **Month of publication:** April 2024

DOI: https://doi.org/10.22214/ijraset.2024.61156

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue IV Apr 2024- Available at www.ijraset.com

Design and Development of Android App Malware Detector API Using Androguard and Catboost

Vamsee Krishna K[1], Sujith Kumar P[2], Deepak S[3], Gopala Krishnan C[4], Rajeswari S[5]
[1,2,3,4]*Students,* [5]*Supervisor, Department of Computer science and Engineering, Adithya institute of technology, Coimbatore, Anna University, TamilNadu, India.*

Abstract: ***Android has been constant target of attacks through the use of malicious applications (malware) that can harm users, for instance, by leaking sensitive data (e.g., bank account details), blocking access to information and demanding monetary compensation for the ransom, or even leveraging social engineering scams. A large number of malicious applications is distributed in a daily basis through different distribution vectors, such as application markets, including official and third party stores, or untrusted sources like Web repositories. The increasing prevalence of Android malware necessitates advanced detection mechanisms to safeguard users and their devices. The aim of the project is to presents an innovative approach to Android malware detection, integrating machine learning, blockchain technology, and web development tools. The proposed system is to develop a Andriod App Malware Detector API capable of identifying and mitigating malicious applications effectively. The system begins with the collection of diverse datasets comprising both benign and malicious Android applications, ensuring the representation of various threat vectors and scenarios. Androguard, a powerful tool for Android app analysis, the system extracts relevant features such as permissions, API calls, and intents, providing valuable insights into the behaviour and characteristics of each application. The AndroMal Model is designed and trained using CatBoost Algorithm and integrated with a blockchain framework for secure metadata storage. Consensus mechanisms and smart contracts enhance transparency and reliability. Privacy considerations, continuous monitoring, and user feedback mechanisms further fortify the system. The proposed design promises an effective, scalable, and user-centric solution for Android malware detection.***
Keywords: Android, Malicious Applications, Datasets, Andro guard, Intents, Threat Vectors

I. INTRODUCTION

Android App is a software designed to run on an Android device or emulator. The term also refers to an APK file which stands for Android package. This file is a Zip archive containing app code, resources, and meta information. Apps are normally distributed through app markets such as Google Play Store, so it is possible to enable installation from APK file or via USB connection in device settings. To install or distribute APK in stores, it should have a unique package name (e.g., com.example.app) stored in the meta-information.

A. Types of Android Applications

Native apps are built for particular operating systems, which are mostly App Store. Native apps are generally built to make the most of all the features and tools of the phones such as contacts, cameras, sensors, etc. Native apps ensure high performance and stylish user experience as the developers use the native device UI to build apps.
Android and IOS. Also, there are more OS for mobile applications: Blackberry and Windows. This is available for download on Google Play Store

Fig .1. Android

©IJRASET: All Rights are Reserved

B. Dataset to create a Model Using Machine Learning

Machine Learning algorithm is trained using a training data set to create a model. When new input data is introduced to the ML algorithm, it makes a prediction based on the model.

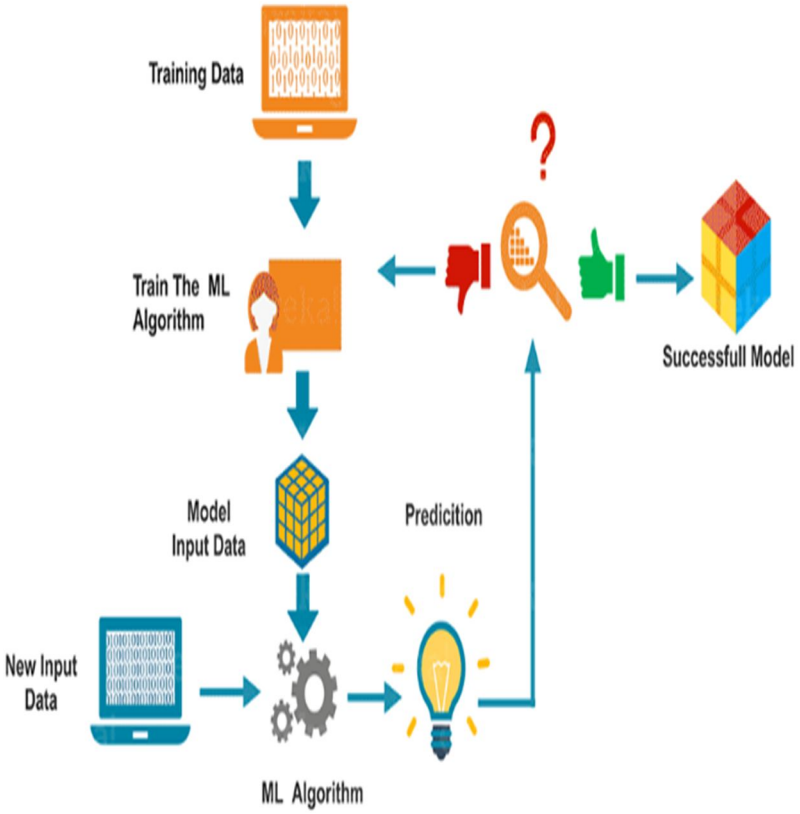


Fig .2. Dataset to create Model Using Machine Learning

The prediction is evaluated for accuracy and if the accuracy is acceptable, the Machine Learning algorithm is deployed. If the accuracy is not acceptable, the Machine Learning algorithm is trained again and again with an augmented training data set. This is just a very high-level example as there are many factors and other steps involved.

C. CatBoost

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can easily integrate with deep learning frameworks like Google's TensorFlow and Apple's Core ML. It can work with diverse data types to help solve a wide range of problems that businesses face today. To top it up, it provides best-in-class accuracy.Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

"CatBoost" name comes from two words "Category" and "Boosting".

Catboost has two methods: The first is "Prediction Values Change". For each feature, Prediction Values Change shows how much, on average, the prediction changes if the feature value changes. A feature would have greater importance when a change in the feature value causes a big change in the predicted value. This is the default feature importance calculation method for non-ranking metrics. The second method is "Loss Function Change". This type of feature importance can be used for any model but is particularly useful for ranking models. For each feature, the value represents the difference between the loss value of the model with this feature and without it. Since it is computationally expensive to retrain the model without one of the features, this model is built approximately using the original model with this feature removed from all the trees in the ensemble. The calculation of this feature importance requires a dataset.

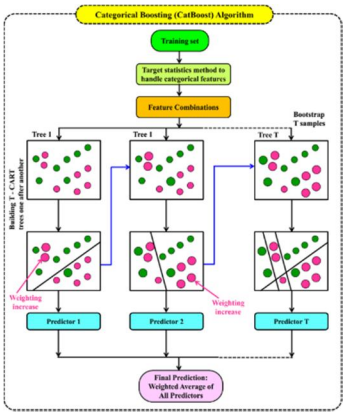


Fig .3. CatBoost

D. Blockchain

Blockchain is a method of recording information that makes it impossible or difficult for the system to be changed, hacked, or manipulated. A blockchain is a distributed ledger that duplicates and distributes transactions across the network of computers participating in the blockchain. Blockchain technology is a structure that stores transactional records, also known as the block, of the public in several databases, known as the "chain," in a network connected through peer-to-peer nodes. Typically, this storage is referred to as a 'digital ledger.' Every transaction in this ledger is authorized by the digital signature of the owner, which authenticates the transaction and safeguards it from tampering. Hence, the information the digital ledger contains is highly secure.

Fig .4. Blockchain

Three types of blockchain

1) Public blockchain.

A public, or permission-less, blockchain network is one where anyone can participate without restrictions. Most types of cryptocurrencies run on a public blockchain that is governed by rules or consensus algorithms.

2) Permissioned or private blockchain.

A private, or permissioned, blockchain allows organizations to set controls on who can access blockchain data. Only users who are granted permissions can access specific sets of data. Oracle Blockchain Platform is a permissioned blockchain.

3) Federated or consortium blockchain.

A blockchain network where the consensus process (mining process) is closely controlled by a preselected set of nodes or by a preselected number of stakeholders.

4) Sidechains

A sidechain is a blockchain running parallel to the main chain. It allows users to move digital assets between two different blockchains and improves scalability and efficiency. An example of a sidechain is the Liquid Network.

E. Objective Aim

The primary aim of this project is to develop a robust and innovative Android malware detection system that combines machine learning, blockchain technology, and advanced analytics. The system is designed to effectively identify and mitigate the threats posed by malicious Android applications, ensuring the security and integrity of users' devices and personal data.

II. METHODOLOGY

The design and development of the Android Malware App Detector Web App Leverage Python's versatile ecosystem, including Flask for web development, MySQL for database management, and Wampserver for local server hosting. Essential data processing is facilitated by Pandas, NumPy, and Scikit Learn, while visualization tasks utilize Matplotlib and Seaborn. Additionally, integration with blockchain technology for secure metadata storage is achieved through JSON, ensuring robustness, while Bootstrap provides a responsive and visually appealing user interface. The Android Malware App Detector Web App consists of modules to ensure efficient and secure malware analysis. Users navigate seamlessly through features like user authentication and file upload, which guarantee data integrity and compliance. Following pre-processing, analysis, and result presentation, administrators oversee system management and monitoring via the Admin Dashboard and Logging and Monitoring modules. Robust security measures and integration capabilities fortify the app's reliability and interoperability, empowering users with actionable insights while safeguarding sensitive data and system integrity.

III. RELATED WORKS

A. *Trust Chain Integration*

The Trust Chain Integration module is pivotal in embedding blockchain technology within the system, ensuring a secure and transparent environment for data storage and management. Beginning with the selection of an appropriate blockchain framework, this module orchestrates the setup of a resilient blockchain network, tailored to the system's requirements. Through meticulous data structure design and transaction handling, metadata pertaining to app analysis, permissions, and prediction results are securely stored on the blockchain, ensuring immutability and transparency. Consensus mechanisms and smart contracts are deployed to validate transactions and enforce predefined rules, fostering trust and reliability within the network. With a focus on secure access control and privacy considerations, users' data integrity and confidentiality are safeguarded, bolstered by monitoring, auditing, and educational initiatives to promote user adoption and participation. In essence, the Trust Chain Integration module fortifies the system's security, transparency, and reliability, leveraging blockchain technology to elevate data management and integrity assurance.

B. *AndroMal Model: Build and Train*

The AndroMal Model module encompasses a series of steps to build and train a machine learning model for Android malware detection. Here's an overview of each step:

1) *Import Dataset*

This module facilitates the retrieval of the dataset containing samples of both malware and legitimate Android applications. It may include methods for downloading datasets from online repositories or importing datasets stored locally. Once the dataset is obtained, this module validates its integrity and format to ensure it meets the required specifications. It checks for any inconsistencies or errors that could affect the training process. The dataset is divided into training, validation, and testing sets to facilitate model evaluation and performance assessment. This module ensures an appropriate split ratio to prevent overfitting or under fitting of the model. Depending on the requirements of the AndroMal model, this module may perform transformations on the dataset, such as encoding categorical variables or scaling numerical features, to ensure compatibility with the chosen machine learning algorithm.

2) *Pre-processing*

The Preprocessing module is crucial for preparing the dataset for effective training of the AndroMal model.the AndroidManifest.xml file unveils crucial metadata, offering a holistic view of the application's characteristics. Optional dynamic analysis techniques may complement static analysis to glean insights into runtime behavior. These meticulous

3) *Model Training and Storage*

Once the configuration parameters are specified, the model training process commences. Using the prepared dataset, the CatBoost model undergoes iterative optimization to learn the underlying patterns and characteristics that distinguish malware from legitimate Android applications. During training, the model updates its parameters based on the gradient descent optimization algorithm, aiming to minimize a specified loss function. By iteratively adjusting the parameters, the model gradually improves its ability to differentiate between different classes of applications. The training process involves repeatedly presenting batches of data to the model, computing gradients, and updating the model's parameters to minimize the loss.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Through this iterative process, the model learns to recognize patterns and features indicative of malicious behavior, ultimately achieving a state where it can accurately classify new Android applications as malware or legitimate. And

The Model Storage module is responsible for securely storing the trained AndroMal model into the Trust Chain. First, the model is serialized into a format suitable for storage and transmission. Then, it undergoes encryption to safeguard its confidentiality. Encrypted model data is embedded into a blockchain transaction, which is validated and added to the blockchain ledger. Smart contracts manage interactions and enforce access control policies. Metadata, including version information and training parameters, is recorded alongside the model data. Once stored, the model becomes immutable and tamper-proof, with access regulated through smart contracts. An audit trail of all storage transactions provides transparency and accountability. This approach ensures the integrity and security of the trained AndroMal model throughout its storage lifecycle.

IV. FEATURE EXTRACTION

The Feature Extraction module, powered by AndroGuard, is instrumental in distilling pertinent insights from Android

Download App: Users browse and download Android applications from the system, selecting from a variety of available options based on their preferences and requirements.

Try to Install: Users attempt to install downloaded applications on their devices, initiating the install.

These detailed modules collectively form the user interface of the AndroMal prediction system, providing a seamless and intuitive experience for administrators, app developers, and including parameter tuning and performance evaluatend users alike. Each module is designed to fulfill specific functionalities, ensuring efficient system operation and user satisfaction.

A. AndroMal Model: Build and Train

The AndroMal: Build and Train module, harnessing the power of the CatBoost algorithm, is instrumental in constructing and application files (APKs) to fuel subsequent analysis. Through meticulous scrutiny, it dissects various facets of the application, including permissions, API calls, intents, and components. By scrutinizing the application's bytecode and decompiled source code, it uncovers intricate details like method invocations and control flow structures. training the machine learning model for Android malware detection. Here's a detailed overview of its functionalities:

Model Configuration:The model configuration stage within the AndroMal: Build and Train module involves setting up the parameters that dictate the behavior and performance of the CatBoost algorithm. These parameters, including tree depth, learning rate, and regularization settings, profoundly influence how the model learns from the dataset and makes predictions. Tree depth determines the maximum depth of decision trees in the ensemble, controlling the model's complexity and capacity to capture intricate patterns in the data. A deeper tree can potentially capture more complex relationships but may lead to overfitting if not properly regulated. The learning rate parameter governs the step size taken during gradient descent optimization, affecting the speed and quality of convergence during training. A smaller learning rate results in slower but more stable convergence, while a larger rate may lead to faster convergence but risk overshooting the optimal solution. Regularization settings, such as L1 and L2 regularization, are used to prevent overfitting by penalizing large coefficients in the model. Regularization helps in promoting simpler models that generalize well to unseen data

B. End User Interface

The End User Interface comprises distinct modules tailored to cater to the specific needs and functionalities of administrators, app developers, and regular users within the AndroMal prediction system:

1) Admin Interface:

- Login: Administrators access the system securely via unique credentials, ensuring authorized access to administrative functionalities.
- Upload Dataset: This feature empowers administrators to upload diverse datasets, crucial for model training and evaluation, ensuring the system's predictive accuracy and effectiveness.
- Train the Model: Administrators utilize this functionality to initiate model training sessions, including parameter tuning and performance evaluated.
- User Management**:** Administrators oversee user accounts, managing roles, permissions, and access rights within the system, ensuring efficient and secure user management.

2) App Developer Interface:

- Register**:** App developers create accounts within the system, providing requisite information such as contact details and credentials.
- Login: Registered developers securely access the system, utilizing their unique credentials to log in and access developer-specific functionalities.
- Upload Developed App: This feature allows developers to upload their Android applications for analysis and certification, facilitating the integration of their apps into the system.
- Receive App Certification: Upon uploading an application, developers receive timely feedback regarding its certification status, indicating whether it is classified as legitimate or malwaredevices from potential security threats.

3) User Interface:

- Register: Users create accounts within the system, furnishing necessary information to establish their profiles.
- Login: Registered users log in securely to access system functionalities, ensuring secure and personalized access to available features

To evaluate the performance of the Android Malware App Detector Web App project, various metrics including confusion matrix, accuracy, precision, recall, and F1-score can be computed based on the model's predictions. Here's how each metric is calculated:

V. PERFORMANCE ANALYSIS

A. Confusion Matrix

A confusion matrix is a table that summarizes the performance of a classification algorithm. It consists of four main components: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

- TP: The number of correctly predicted positive instances (malware).
  These metrics can be computed using the predictions generated by the Android Malware App Detector Web App model and comparing them with the ground truth labels. The confusion matrix provides a detailed breakdown of the model's performance across different classes (malware and legitimate applications), while accuracy, precision, recall, and F1-score offer summarized measures of its effectiveness in classification tasks. High values of these metrics indicate better performance of the model in detecting malware applications while minimizing false positives and false negatives.
- FP: The number of incorrectly predicted positive instances (legitimate applications classified as malware).
- TN: The number of correctly predicted negative instances (legitimate applications).
- FN: The number of incorrectly predicted negative instances (malware classified as legitimate applications).

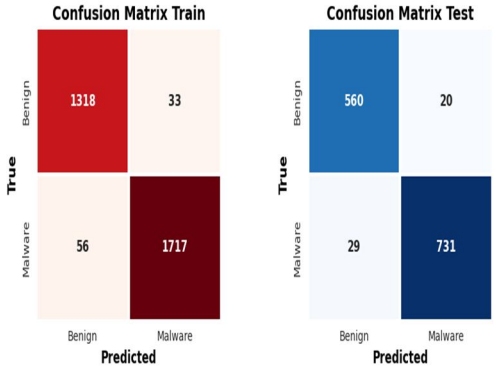


Fig .5. Confusion Matrix

B. Accuracy

Accuracy measures the proportion of correctly classified instances out of the total instances.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

C. Precision

Precision measures the proportion of correctly predicted positive instances (malware) out of all instances predicted as positive.

Precision = TP / (TP + FP)

D. Recall (Sensitivity)

Recall measures the proportion of correctly predicted positive instances (malware) out of all actual positive instances.
Recall = TP / (TP + FN)

E. F1-Score

F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics.
F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

VI. RESULTS

The performance evaluation of the Android Malware App Detector Web App project yields promising outcomes in the realm of malware detection and classification. Leveraging machine learning algorithms and robust feature selection techniques, the model demonstrates commendable accuracy in discerning between malicious and legitimate applications. With high precision, recall, and F1-score metrics, the system effectively identifies potential security threats within the Android ecosystem. The results, as evidenced by the confusion matrix, provide a comprehensive overview of the model's performance across different classes, highlighting its ability to accurately classify malware instances while minimizing false positives and negatives.

VII. DISSCUSSION

The achieved results underscore the significance of the Android Malware App Detector Web App as a proactive cybersecurity solution for mobile devices. By harnessing advanced machine learning capabilities and feature selection methodologies, the system exhibits robustness in detecting and mitigating malware risks. The high accuracy, precision, and recall rates validate the efficacy of the model in distinguishing between malicious and benign applications, thereby empowering users and administrators with actionable insights to safeguard against evolving security threats. Moreover, the comprehensive analysis facilitated by the confusion matrix enables deeper insights into the model's strengths and areas for improvement, informing future optimization efforts and algorithmic enhancements. Overall, the results affirm the system's role in bolstering mobile security and underline the importance of ongoing refinement to stay ahead of emerging threats in the dynamic landscape of Android malware.

VIII. CONCLUSION

In conclusion, the project represents a significant advancement in mobile security technology. By leveraging a combination of state-of-the-art tools and methodologies, including machine learning, blockchain, and web development frameworks, the project has successfully created a robust system for detecting and mitigating malware threats on Android devices. Throughout the development process, careful attention has been paid to ensuring the system's reliability, efficiency, and user-friendliness. From data preprocessing and feature selection to model training and deployment, each stage has been meticulously crafted to deliver optimal performance and accuracy in malware detection. Moreover, the integration of blockchain technology adds an extra layer of security and transparency, enabling secure storage and management of critical metadata related to malware analysis.

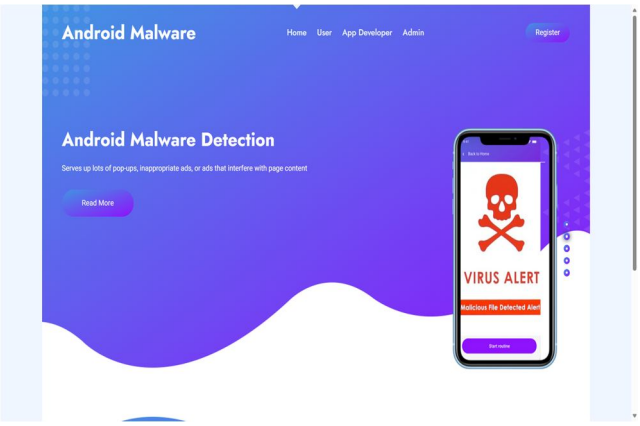


Fig .6. Home Page of Android Malware

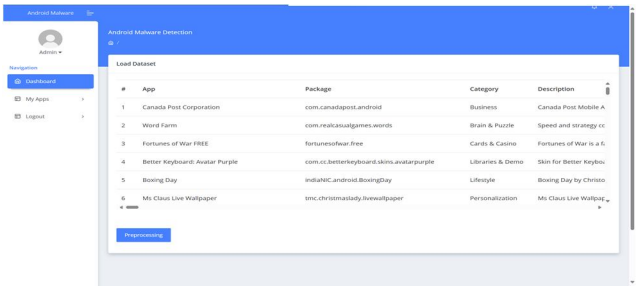


Fig .7. Admin Dashboard Data

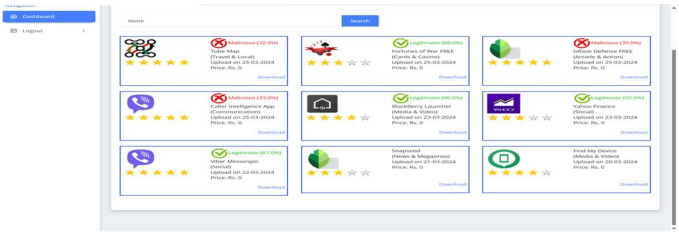
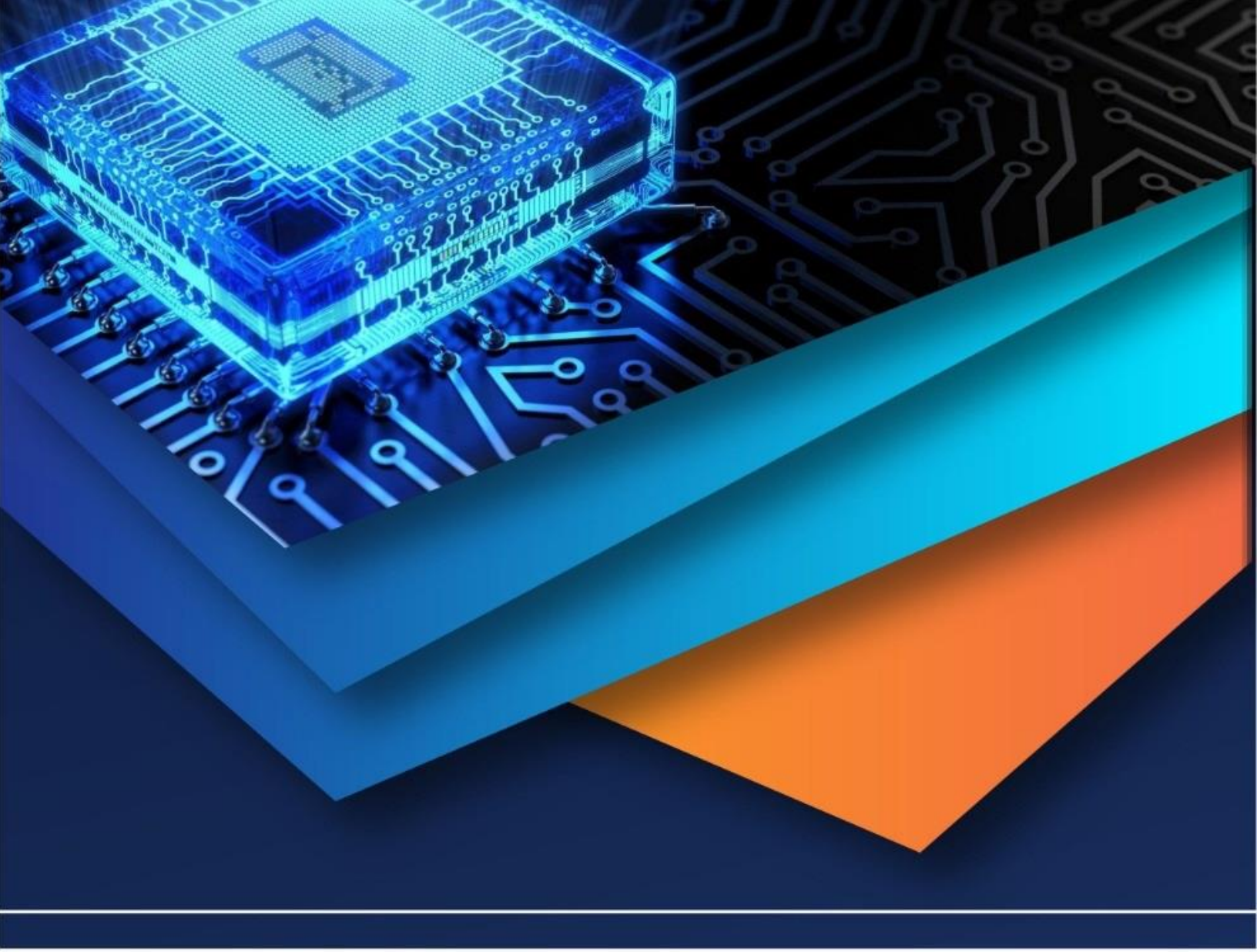


Fig .8. App Detection

REFERENCES

[1] A.Gómez and A. Muñoz, "Deep learning-based attack detection and classification in Android devices", Electronics, vol. 12, no. 15, pp. 3253, Jul. 2023.
[2] H. Rathore, A. Nandanwar, S. K. Sahay and M. Sewak, "Adversarial superiority in Android malware detection: Lessons from reinforcement learning based evasion attacks and defenses", Forensic Sci. Int. Digit. Invest., vol. 44, Mar. 2023.
[3] L. Hammood, I. A. Dogru and K. Kiliç, "Machine learning-based adaptive genetic algorithm for Android malware detection in auto-driving vehicles", Appl. Sci., vol. 13, no. 9, pp. 5403, Apr. 2023.
[4] M. N. AlJarrah, Q. M. Yaseen and A. M. Mustafa, "A context-aware Android malware detection approach using machine learning", Information, vol. 13, no. 12, pp. 563, Nov. 2022
[5] H. Wang, W. Zhang and H. He, "You are what the permissions told me! Android malware detection based on hybrid tactics", J. Inf. Secur. Appl., vol. 66, May 2022.
[6] A Roy, D S Jas, G Jaggi et al., "Android malware detection based on vulnerable feature aggregation", Procedia Computer Science, vol. 173, pp. 345-353, 2020.
[7] S Y Yerima and S Sezer, "Droidfusion: a novel multilevel classifier fusion approach for Android malware detection", IEEE Trans. on Systems Man and Cybernetics, vol. 49, no. 2, pp. 453-466, 2019.
[8] G. Xu, C. Zhang, B. Sun, X. Yang, Y. Guo, C. Li, et al., "AppAuth: Authorship attribution for Android app clones", IEEE Access, vol. 7, pp. 141850-141867, 2019.
[9] H. Wang, J. Si, H. Li and Y. Guo, "RmvDroid: Towards a reliable Android malware dataset with app metadata", Proc. IEEE/ACM 16th Int. Conf. Mining Softw. Repositories (MSR), pp. 404-408, May 2019.
[10] X Y Liu, J Weng, Y Zhang et al., "Android malicious application detection based on APK signature information feedback", Journal of Communications, vol. 38, no. 5, pp. 190-198, 2017.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)