International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

AndroMal: A Blockchain-Based Machine Learning Framework for Android Malware Detection

Angelin Rosy M[1], Aswini Bai S[2]

[1]Assistant Professor, II MCA, Department of Master of Computer Applications, Er. Perumal Manimekalai College of Engineering, Hosur
[2]Department of Master of Computer Applications, Er. Perumal Manimekalai College of Engineering, Hosur

Abstract:
Due to many forms of malware, Android users are vulnerable to attacks that steal their data, block information until a ransom is paid or make it easier for scammers. More criminals are uploading malicious software into official storefronts, third-party markets and web guardians show that stronger malware detection systems are a necessity. In this project, machine learning, blockchain technology and web development tools work together to bring a new technique for detecting Android malware. An Android App Malware Detector API is part of the system, helping to identify and stop hazardous applications. The process begins by assembling datasets with a variety of Android apps, some good and some dangerous, to protect against many kinds of threats. Androguard is a useful tool that helps the system find permissions, API calls and intents for each app, exposing how each application functions. The established AndroMal malware detection model relies on CatBoost to foresee threats and the system stores its metadata through blockchain. The system is made transparent and reliable because of consensus mechanisms and smart contracts. The system we propose is equipped with privacy measures, ongoing monitoring and methods for accepting user input to make it more reliable. With this design, detecting Android malware becomes more efficient, easy for users and flexible, defending against ongoing changes in the mobile world.

Keyword: Android malware, Detection of malware, Blockchain, Machine learning, Cyber security, Security for Android, Threat intelligence

INTRODUCTION

Android App refers to software that runs through an Android device or emulator. The term is often used for an APK file which means Android package. The contents of this file are app code, resources and meta information in Zip format. Since most apps are available on app markets, you can get the APK from the site and set your device to install from USB. If you want to put APK in stores, ensure the package name (such as com.example.app) is set in the meta-information.

Types of Android Applications

Native Apps

Most native apps are designed to run on Android and IOS systems. Another thing to know is that OS

options for mobile apps include Blackberry and Windows. A nucleotide sequencer app is found on both Google Play Store and Apple App Store. These apps are designed to make use of every tool the phone provides, ensuring maximum performance and attractive looks because developers use the native user interface.

Web Applications

You can use web applications only when you use a browser. They depend largely on using HTML, CSS and Javascript together. This tool is accessed from Chrome, Firefox and other web browsers. Because web apps are so advanced, you might believe that they're native apps, since they share most characteristics and how quickly they respond. One main point that sets native mobile apps apart from web apps is that they can be used when offline, while web apps only work if an internet connection is present.

Problems Identified

There are many problems that impact Android users' experience and safety. One of the biggest concerns is security, because apps are at risk of being exploited by cybercriminals for stealing or accessing information illegally. There are privacy issues when apps gather a lot of individual data without the users' permission. Both programmers and scientists have come across several important issues in the current system for finding Android malware apps. A main problem is that signature-based detection methods find it hard to handle the fast transformation of different malware forms. Many traditional techniques do not spot zero-day and polymorphic malware, so devices are not fully protected from difficult threats. It is not possible to analyze all the available Android apps by hand because there are far too many, meaning it takes too long to spot and fix new threats. As a result, poor feature selection options make it hard for machine learning systems to achieve effective detection, resulting in many false positives. Also, since there is no standard and secure way to handle data, it is hard to ensure the privacy and quality of the data, so the current technology can only do so much. Solving these problems requires using both modern ways to check behavior and machine learning methods, combined with proper handling of information and the use of automatic feature selection to improve detection accuracy and efficiency. To solve these issues, the project is creating an efficient Android malware system that uses programs and algorithms, blockchain, proper data treatment and constant interventions to boost detection, scalability and ensure users' protection.

PROPOSED WORK

This system suggests making Android malware detection more innovative by using machine learning, blockchain and web development tools. Here's a summary of the proposed system:

You should focus on gathering and preparing your data.

Benign as well as malicious software for Android is included in the datasets collected.

At this stage, techniques are used to remove errors, eliminate extra or similar pieces of data and make sure the data is uniform in scale.

Using AndroGuard, I have extracted features from the app.

Using AndroGuard, developers can pull out important details from Android apps, including permissions, API usage and intents.

Thanks to these features, it becomes easier to recognize the behavior patterns of any application which supports malware detection.

Use CatBoost to train a Machine Learning model.

A machine learning model for detecting malware is built using the CatBoost algorithm.

It studies the extracted properties and is refined to correctly assign applications to the benign or malicious classes.

Using Blockchain so that DeFi metadata is stored securely.

The system uses blockchain to securely manage metadata from analysis, permission mapping and prediction outcomes.

Using consensus mechanisms keeps the ledger clear and prevents it from being edited, thanks to smart contracts.

User Interface and Interaction Input.

We make the web interface simple so that users can operate the system easily.

Anyone can submit an Android app for study on the platform, examine results and receive warnings about possible malware problems.

By using advanced techniques, the newly proposed approach helps to effectively and scaleably detect Android malware and maintain security and reliability in play stores.

METHODS

1. Android Malware App Detector on the Web

To build the Android Malware App Detector Web App, we relied on Python's many features, using Flask to make the web app, MySQL to handle the database and Wampserver for running the local server. Basic data processing in Python is achieved with Pandas, NumPy and Scikit Learn and Matplotlib and Seaborn help in visual tasks. Furthermore, delicate metadata is stored using blockchain technology and JSON and Bootstrap guarantees a friendly and responsive user interface. The Web App for Android Malware App Detector includes different modules for better and safer malware analysis. It is simple for users to sign in and upload files, as these features make sure data is secure and in accordance with compliance rules. When pre-processing, analysis and result showing are complete, administrators handle management and continuous monitoring with the Admin Dashboard and Logging and Monitoring modules. Because of its strong security and compatibility, the app can offer useful and reliable information and protect all important information in the system.

2. Trust Chain Integration

Trust Chain Integration is crucial because it introduces blockchain to the system, ensuring data in the app is protected and clearly managed. It starts by picking a proper blockchain framework and guides the creation of a endurable blockchain setup adapted to the system's demands. All information about analyze apps, permissions and predictions is kept safely on the blockchain because of careful design and proper transaction handling. Transactions are validated by consensus mechanisms and smart contracts which also ensure that the agreed-on network rules are complied with, encouraging trust and reliability. Thanks to attention on privacy and access control, both the safety and confidentiality of users' information are secured by means of monitoring, auditing and an emphasis on getting users involved. In short, Trust Chain Integration increases safety, openness and reliability in the system by using blockchain technology to upgrade data security and assurance.

3. 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:

3.1. Import Dataset

This module helps get the dataset with both malware and legitimate Android apps. You might perform downloads from the internet or load data you have saved on your own system. After the dataset is obtained, this module checks that it is of the right quality and matches the correct format. It scans for any missed details or blunders which could negatively influence learning. For better control, the data is assigned to training, validation and testing groups. A correct model splitting ensures that neither overfitting nor under fitting occurs. Based on what AndroMal calls for, this module has the ability to turn some variables into codes or adjust their size to match the machine learning algorithm used.

3.2. Pre-processing

The Preprocessing module is crucial for preparing the dataset for effective training of the AndroMal model. Here's an outline of its components:

Managing records with dull values

Through this module, any empty fields within the data are identified and treated as needed. Specific approaches are applied to keep the data safe before any analysis in the dataset.

Get Rid of Data that Appears More than Once

Information where one instance has been included more than once is individually highlighted and automatically removed. As a result, the model's training improves and no bias occurs.

Delete instances that appear more than once: Xunique = unique(X)

When values are missing, we need a way to handle them.

Missing values also need to be identified and taken care of like null values. Either inserting estimations or removing data points with unknown values is used to improve data integrity.

Median missing values by replacing them with the median value of the rest.

If data is missing from a variable, put Ximputed = the value that shows up most often in the data.

Normalization

All numeric details in the data are set to a standard scale so they are all similar in scope. This allows important aspects of a model to be properly addressed during training.

The role of pre-processing is to ensure the dataset is free of errors, uniform in standards and suitable for better performance of AndroMal, increasing its accuracy and dependability.

3.3. Feature selection

Identifying the important features from the dataset using the *Filter Based Approach* module is essential for making the Android malware detection system more efficient. It includes a number of important parts. Feature Ranking helps determine the importance of each feature using correlation, mutual information or chi-squared as methods. The scores are used to rank each feature, giving useful guidance on which features matter most in telling malware from regular applications.

The following step for the module is to use feature ranking to set the best threshold for excluding unimportant features. Here, using statistical analysis or the experience related to the domain helps you decide which features are most significant and cost-effective.

The model uses the set threshold to choose and preserve only the top features from the dataset. As a result, the data contains the most important features which improves the model's performance and reduces the number of features.

By using a systematic method to select features, Android malware detection becomes easier and more accurate, saving resources too

3.4. Feature Extraction

AndroGuard's Feature Extraction module helps pull out necessary information from Android application files (APKs) that will guide the following steps. By examining the application very closely, it focuses on things such as permissions, API calls, intents and components. By examining both the bytecode and source code, the application tells about the steps in the program, allowing you to identify potential vulnerabilities. In addition, examining the AndroidManifest.xml file reveals important information about the app's main features. Optional dynamic tools can add to the information obtained through static analysis about behavior at runtime. Because of these well-extracted features, the AndroMal model can spot and categorize Android malware more accurately

3.5. AndroMal Model: Build and Train

By employing CatBoost, the AndroMal: Build and Train module is important for creating and teaching an Android malware detection system. This is a close look at what Go has to offer:

How the Model Was Built

At this stage in the AndroMal: Build and Train module, you configure the settings needed for the CatBoost algorithm to perform well. Tree depth, learning rate and regularization strongly affect how the model learns and makes its predictions. Decision trees in the ensemble can be as deep as the trees chosen in the forest, affecting the model's ability to detect detailed patterns in the data. Going deeper in the tree could unearth tougher relationships, although it can also cause overfitting if regulated poorly. During training, the learning rate controls how large the changes are made by gradient descent and affects its performance. Small learning rates allow the model to learn steadily, while using large rates can cause it to converge quickly but might land it too far from the optimal solution. L1 and L2 regularization settings try to stop overfitting by making sure the large values in the model do not have as big of an effect. The use of regularization makes it possible for models to fit data well and apply well to new, unknown data

Model Training

As soon as the configuration parameters are set, the model training starts. The prepared data is passed through CatBoost, allowing it to improve its performance by spotting the differences between malware and valid Android applications. During training, the model uses gradient descent to adjust its parameters in an effort to reduce a specific loss function. When they keep adjusting the parameters, the model gets better at telling apart applications from various types. Repeatedly, the training requires giving a batch of data to the model, finding losses and adjusting parameters to decrease the error. Thanks to this process, the model learns to recognize signs of bad behavior in Android applications and, moving forward, can tell whether a new application is malicious or safe.

3.6. Model Storage

It is the Model Storage module's job to protect the AndroMal model as it is stored in the Trust Chain. Next, the model is transformed into a storage-friendly and transmit-friendly format. After that, encryption is applied to guarantee that the data is confidential. The encrypted model data becomes part of a blockchain transaction that is checked and then logged in the blockchain record. Built-in rules and access limitations are taken care of by smart contracts. Metadata comprising the version and training settings is stored together with the model. The stored and locked model cannot be changed and control over it is regulated with smart contracts. Transparency and accountability are made possible by reviewing all of the transaction logs. This method guarantees that the AndroMal model remains safe and secure from its design to its use.

4. Android App Malware Predictor

It is critical for proactive identification of Android app malware by relying on malware indicators. Here we'll explain what makes up this structure:

4.1. How to Infect a Phone with Android Malware

With this component, users can choose to install applications directly through app stores or upload an APK file for scrutiny. They are free to either install apps from known developers or transfer them from their device's own storage.

4.2. Feature Extraction

When the module is installed or a file uploaded, it launches a process to pick out key attributes and behavior from the Android app. During feature extraction, people study the manifest file, break down the code to view its inner workings and look at aspects such as permissions, API calls, intents and other important features.

4.3. Prediction

This is what the Android App Malware Predictor does best. It uses the AndroMal model on Trust Chain to assess whether the application is safe from malware or not. The features are examined by sophisticated machine learning methods which classify the application as good or bad. The Trust Chain is also used to make a record of the predicted result as a kind of transaction. This maintains transparency and you can rely on it because each prediction result is protected on the blockchain ledger. Using the public blockchain to save predictions, users get to check all past assessments which makes the results clear and trustworthy.

5. Malware Level Indicator

Once malware is discovered, The Malware Level Indicator module fully assesses the risk, with a focus on the type of threat, possible damage to the device and user privacy and recognized security issues. Relying on set rules and formulas, the module gives each malware a number to represent how risky it is. Users can learn from the score whether the threat is minor or serious. Furthermore, the module organizes the detected malware into various well-known classifications, including adware, spyware, ransomware and trojans which tells users about the kind of threat each one is. To make things clear for users, the module displays the malware level indicator using colors or short descriptions. Thanks to this display, users can understand how serious the malware is and select the best response. Because of these features, the Malware Level Indicator module gives users the chance to handle risks linked to malware threats in Android apps.

6. Malware App Installation Blocker

The Malware App Installation Blocker module serves as a proactive defense mechanism within the AndroMal prediction system, preventing the installation of Android applications identified as malware. Here's a detailed description of its functionality: Upon detecting an attempt to download or install an Android application flagged as malware by the AndroMal prediction system, the module intervenes to block the installation process. It analyzes the application metadata and compares it against the prediction results to verify the malware classification. If the application is confirmed as malware, the module promptly blocks the installation attempt, preventing the application from being installed on the user's device. This module operates seamlessly in the background, intercepting installation requests in real-time and providing immediate protection against potentially harmful applications. By proactively blocking the installation of malware apps, it safeguards users' devices from security threats and ensures a secure and trustworthy app environment. Additionally, the module may log and report blocked installation attempts,

providing valuable insights into the prevalence and distribution of malware across app ecosystems. Through its proactive intervention, the Malware App Installation Blocker module contributes to enhancing overall device security and protecting users from the risks associated with malware-infected applications.

7. Integrity Verification

This module is critical for the system because it checks for Android application integrity every step of the way. The service carefully studies an application and its resources by using checksums, signatures and hashes to confirm its integrity. Besides protecting users, it watches over repositories and distribution sources to catch any unauthorized activities. Run all the time, it handles suspicious activity and alerts you instantly so actions can be taken straight away. Together with other safeguards, it helps secure and protect the app market for everyone.

8. End User Interface

The End User Interface is designed with different tools to respond to the particular interests and roles of administrators, app developers and regular users of the AndroMal prediction system.

Name: 8.1 Admin Interface

Administrators must log into the platform with individual credentials to safely perform administration duties.

Administrators can use this function to upload a range of important datasets, required for training and testing machine learning models.

- Administrators can start training modeling sessions by selecting this option, improving model performance and checking its accuracy.
- Organizations rely on admins to supervise user accounts, assign roles, set up permissions and keep user management both efficient and secure within the system.

8.2. App Developer Interface is designed to make building apps easy.

Some app developers log in by registering new accounts which requires them to provide their contact and credentials.

To log in, registered users use their credentials to securely access the system and reach the functions designed for developers.

8.3. User Interface:

Users are required to sign up and fill in information to set up their profiles.

Registered users enter a password to make sure they are safe and only can use the features meant for them.

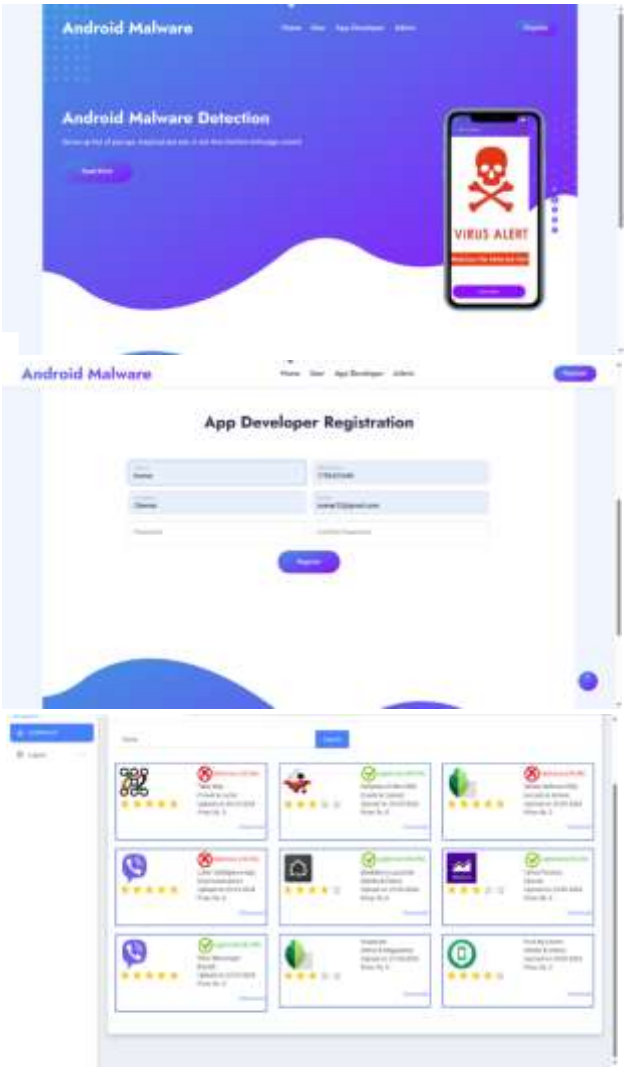
- Users Have the Option to Download: From the system, they are able to find and install Android applications based on what they want.

All of these detailed modules come together to serve as the user interface of AndroMal, allowing administrators, app developers and end users to work easily and smoothly. Every module is set up to handle specific tasks which leads to better system performance and greater user enjoyment.

RESULTS

The Web App Android Malware App Detector has demonstrated good performance in detecting and dividing malware into types. Thanks to machine learning and the right set of handpicked features, the model is able to easily tell apart malicious and legitimate applications. With high precision and recall and the use of the F1-score, the system can discover dangers to security on Android devices. The results

revealed by the confusion matrix show that the model accurately finds malware instances and rarely makes mistakes in labeling or ignoring them.



CONCLUSION

In short, the project marks an important improvement in protecting mobile devices from threats. The project uses the latest technologies and approaches such as machine learning, blockchain and web tools, to successfully manage and handle malware threats on Android. Great care has been taken during development to maintain the system's reliability, efficiency and ease of use. In every stage, starting with processing data, choosing the best features and using models, careful attention has been paid to ensure reliable malware detection results. In addition, the use of blockchain technology ensures the safe and clear storage and management of crucial data about malware analysis. Because its navigation is easy and the interface is straightforward, the project is accessible to both administrators and users. Using tools like user authentication, file upload and alerts in real time, users can take care of their data and devices without fear. According to testing and evaluation, the system's performance has improved. The effectiveness of the system in detecting malware using accurate, precise, remembered and F1-score metrics is highlighted. Besides, confusion matrix results show the system's performance on different kinds of malware which supports making it better. In the future, the industry will continue improving and innovating, mainly by increasing ability to detect threats, widening the range of intelligence sources and adjusting to new trends

in mobile security. Teamwork with industry players and continuous research will help the system provide advanced protection for Android users.

REFERENCES

1. [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. A. Albakri, F. Alhayan, N. Alturki, S. Ahamed and S. Shamsudheen, "Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification", *Appl. Sci.*, vol. 13, no. 4, pp. 2172, Feb. 2023.
4. 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.
5. H. Alamro, W. Mtouaa, S. Aljameel, A. S. Salama, M. A. Hamza and A. Y. Othman, "Automated Android malware detection using optimal ensemble learning approach for cybersecurity", *IEEE Access*, vol. 11, pp. 72509-72517, 2023.
6. 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.
7. J. Samhi, J. Gao, N. Daoudi, P. Graux, H. Hoyez, X. Sun, et al., "JuCify: A step towards Android code unification for enhanced static analysis", *Proc. IEEE/ACM 44th Int. Conf. Softw. Eng. (ICSE)*, pp. 1232-1244, May 2022.
8. 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.
9. P. Bhat and K. Dutta, "A multi-tiered feature selection model for Android malware detection based on feature discrimination and information gain", *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9464-9477, Nov. 2022.
10. H. Zhou, S. Wu, X. Luo, T. Wang, Y. Zhou, C. Zhang, et al., "NCScope: Hardware-assisted analyzer for native code in Android apps", *Proc. 31st ACM SIGSOFT Int. Symp. Softw. Test. Anal. (ISSTA)*, pp. 629-641, Jul. 2022.
11. K Bakour and H M Unver, "DeepVisDroid: Android malware detection by hybridizing image-based features with deep learning techniques", *Neural Computing and Applications*, vol. 33, pp. 11499-11516, 2021.
12. K Khariwal, R Gupta, J Singh et al., "R-M Fdroid: Android malware detection using ranked manifest file components", *International Journal of Innovative Technology and Exploring Engineering*, vol. 10, no. 7, pp. 55-64, 2021.
13. A K Singh, G Wadhwa, M Ahuja et al., "Android malware detection using LSI-based reduced opcode feature vector", *Procedia Computer Science*, vol. 173, pp. 291-298, 2020.