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

Android Trojan Identification Using Machine Learning Techniques: A Review of the Literature

Dr. M.C.Bhanu Prasad[1], Mangala Tejaswini[2], Yerasi Sainath Reddy[3], Bulagondla Sai Divya Sree[4], Sugali Sailaja[5], Dayyala Praveen Kumar[6]

[1]Guide,[2,3,4,5,6]Student
[1,2,3,4,5,6]Department of Cse
Tadipatri Engineering College,
Tadipatri

Abstract

Users can search for any app on the Play Store using the interface. It if available, automatically retrieves the app's permissions list and privacy policy. To view relevant sentences from the privacy settings, users can select a specific permission strategy, alongside a reasonable clarification of what the consent involves. This makes a difference by highlighting important aspects, users quickly assess the privacy risks of Android apps sections of the privacy policy and a description of sensitive permissions. Our strategy introduces a new method for analyzing Android app privacy policies. Our apparatus makes it more obvious the security ramifications of introducing outsider apps and has already discovered issues in apps that are troubling. The tool is made to be expandable, allowing for seamless integration of future enhancements to improve efficiency and dependability. Additionally, if it is not too much work, make reference to the additional requirements in the section below under "Personal and Sensitive Information" if your app handles sensitive or personal user data. These standards for Play Google are on top of any privacy or data security laws that may be relevant. We suggested that when a client wants to introduce and use an external application, they may not understand the significance of the permissions that the application requests, so they grant all of them, allowing malicious applications to be installed and carry out their malicious activities covertly.

Keywords: Machine Learning, Detection, Processing, Data Collection, ML Algorithms.

1. INTRODUCTION

The utilization of cell phones has flooded over the course of the last ten years, offering a wide a wide range of applications for different purposes Cell phones are progressively supplanting customary PCs and are crucial for assignments like web perusing, banking and social networking They likewise support works like SMS informing, constant area sharing, and admittance to specific applications like those for health surveillance. In 2017, 208 million smartphones were sold worldwide. 2012, which was a big improvement from the previous year. This expansion has changed daily routines, affecting both social and business operations interactions. Markets for mobile apps like Apple's App Store and Google Play

also, Microsoft Windows Store give different applications, from games to route also, wellbeing instruments, taking special care of expansive client interests. Notwithstanding, the notoriety of cell phones has likewise made them focuses for malevolent exercises, as some applications compromise client protection. Market strategies change: Apples Application Store upholds rigorous app testing, whereas Google's Play Store gives users more freedom to developers, but when malicious apps are detected, removes them. Both platforms are able remotely erase hurtful applications from gadgets. This highlights the significance of early recognition frameworks for portable malware to keep up with client security.

2. RELATED WORK

In software development, the literature review is one of the most crucial processes. Prior to growing the device, the time component, cost savings, and commercial business resilience must be determined. After these are satisfied, the next stage is to identify the language and operating device that can be utilized to expand the device. Programmers require a lot of outside assistance once they begin building a device. You can get this help from veteran programmers, books, or websites. The system is built with consideration for the previously described problems in order to extend the proposed gadget.

Examining and reviewing all of the challenge improvement's needs is the core function of the assignment improvement department. Literature evaluation is the most crucial stage in the software development process for any task. Prior to expanding the equipment and associated layout, time considerations, resource requirements, labor, economics, and organizational electricity must be identified and evaluated. The next phase is to determine the operating system needed for the project, the software program specifications of the particular computer, and any software that needs to be carried on after those factors have been met and thoroughly investigated. a stage similar to expanding the tools and related capabilities.

The unabated global adoption of cell phones has prompted versatile malware to increase in complexity and scale. Since Android is quickly becoming the most popular mobile platform, malware targeting the platform has significantly increased. Additionally, Android malware is rapidly evolving to evade detection through traditional mark-based testing. Conveniently revealing new malware is still a fundamental problem, regardless of the current recognition estimations that have been established. This calls for creative zero-day Android malware techniques to address the growing danger. Consequently, the authors develop and analyze proactive machine learning techniques that employ Bayesian classification aimed at detecting hidden Android malware through static analysis. Based on a sizable malware sample set that comprises most of the current families, the virus demonstrates highly accurate locating capabilities. In order to develop persuasive static-scientific Bayesian strategies for spotting unknown Android malware that are based on categorization, comparative analysis and empirical data are presented [1].

The Android platform's impressive growth over the last few years has made it a lucrative goal for developers of unpleasant applications. Examples of malicious apps include those that monitor private user information, send premium rate SMS messages, or even if not described as malware, direct sketchy activities influencing the user security or costing them cash. In this paper, we look into whether or not using both the app's category, the permissions it asks for, and what permissions are required by other apps in the same category to provide users with better information whether the potential benefits of installing an app outweigh the potential drawbacks benefit. Only the dangers posed by permissions are considered by current methods mentioned by an application and overlook both the advantages and what consents are

prompted by other apps, resulting in a limited impact. We propose a few risk signals using two datasets, one of which consists of 158,062 Android applications from the Android Market, and 121 additional malicious software we exhibit the adequacy of our proposition through broad information examination [2].

The use and popularity of in recent years has skyrocketed cell phones and cell phones, which is joined by huge sum and wide assortment of component rich cell phone applications. These cell phone applications (or apps), which are typically organized into distinct Mobile users can easily browse marketplaces and then simply clicked to introduce on various cell phones. By and by, other than the official marketplaces operated by platform vendors (like Apple and Google, for example) In order to maintain and advance a clean smartphone app ecosystem, several alternative marketplaces have also been established by third parties to house thousands of applications (for instance, to satisfy localization or regional requirements); each of these marketplaces must be an outside commercial center that provides high-quality applications to a diverse clientele. We thoroughly examine six well-known third-party markets for Android in this post. Among these, we consider it a common "in nature" practice to repackage and distribute authentic programs from the official Android Market through third-party marketplaces in order to enhance Understanding the magnitude of this type of activity, we employ the fuzzy hashing approach in the app similarity Droid MOSS assessment system to effectively find and identify app-behavior that is being repackaged. According to the Droid MOSS trials, between 5 and 13 percent of the apps hosted on the markets under study underwent repackaging. According to further manual research, they were repackaged. Applications are mostly employed to replace current in-application advertisements or introduce new ones in order to "take" or re-course advertising incomes. We also identify a few instances when repackaged programs have harmful payloads or backdoors that have been placed. According to the findings, a comprehensive verification cycle is necessary to improve marketplace guidelines for third-party smartphone applications [3].

Outsider applications (applications) drive the engaging quality of web and portable platforms for applications the control structure of many of these platforms is decentralized procedure, depending on express client assent for giving authorizations that the Applications ask. As the primary source of information for users, community ratings signs to recognize the possibly unsafe and unseemly applications even despite the fact that community ratings typically reflect opinions regarding usefulness or execution instead of about chances. With the onset of User-consent permission systems, like those found in HTML5 web apps, will become increasingly widespread. We leverage the extensive data gathering from Facebook apps, Chrome extensions, and Android apps to study the operation of user-consent authorization systems. Our investigation indicates that the local area evaluations currently used in application showcases are not reliable indicators of an application's protection risks. We find various indications of efforts to lure or lull people into granting permissions: Free apps, apps containing mature material, lookalike apps, and apps with names similar to well-known apps also ask for more permissions than they often do. Additionally, we find that well-known applications require more consents at each of the three stages [4].

Personal information is frequently stored on Android phones, attracting malicious designers to implant code in Android applications to take delicate information. With known methods from the literature, it is simple to determine whether sensitive data is being sent from an Android smartphone. Nonetheless, transmission of delicate information in itself doesn't be guaranteed to show protection spillage; a superior marker may depend on whether the user intended the transmission or not. At the point

when transmission isn't expected by the client, it is more probable a protection spillage. The issue is instructions to decide whether transmission is client expected as a starting point for this space, we present App Intent, a brand-new analysis framework. Each data set transmission, App Intent is able to effectively provide a series of GUIs manipulations that correspond to the order of the occurrences that result in the data transmission, accordingly assisting an investigator with deciding whether the information transmission is client expected or not. The fundamental thought is to utilize emblematic execution to produce the above-mentioned sequence of events, but simple symbolic execution demonstrates to take too much time to be useful. App Intent's major innovation is to make use of the distinctive Android execution model to cut down on search space without forfeiting code inclusion. We likewise present an assessment of App Intent with a collection of 750 malicious apps and 1,000 of Google's best free apps Play. The outcomes demonstrate the way that App Intent can successfully help separate the applications that really spill client security from those that don't [5].

3. EXISTING SYSTEM

The GP-PP model, which stands for "Generic Permissions-Privacy Invasive Permissions," is helpful. App permissions are divided into two categories by users: generic and privacy invasive. It makes it simple for users to choose which applications could be dangerous when installed. By looking at the permissions that an app, the model determines that it violates privacy if the majority of its requests Permissions are an example of this. This grouping permits clients to make informed choices about which authorizations might actually think twice about privacy. We approve the GP-PP model to guarantee it precisely arranges applications based on the permission sets they requested, ensuring its usefulness in assisting before installing an app, users evaluate its security.

Disadvantages

1. Low Security While the GP-PP model group’s application authorizations to help it lacks robust security, and users identify potential privacy risks measures to stop bad things from happening. The model dependence on authorization Analysis may not fully capture the app developers' intentions, allowing some unsafe applications to dodge location.

2. The model only considers the requested permissions without examining the behavior of the app or the context in which permissions are used. These False positives can occur when benign applications are flagged as having limited scope privacy-invading or false negatives, when apps that are actually harmful are overlooked.

3. Client Dependence The viability of the GP-PP model generally relies upon users' comprehension and interpretation of the results of the classification. Users might lack the technical know-how to base their decisions solely on the model's output, which could cause an unnecessary alarm or a false sense of security.

4. REQUIREMENT ANALYSIS

Evaluation of the Rationale and Feasibility of the Proposed System

This system's main objective is to allow any user to install and utilize the third-party application without understanding the importance of the rights that an application requests, therefore it grants all permissions, which allow malicious applications to be installed and operate their malicious behavior in the shadows. The need of this approach lies in its capacity to proactively distinguish possibly destructive

applications from outsider sources by examining their permissions, improving user security and privacy. By assigning permissions "Normal" or "Dangerous" labels and employing classification algorithms to identify applications that are harmful, this method offers a systematic approach to protecting devices from malicious apps. The potential exists supported by the Android app's availability of extensive datasets consents and the adequacy of AI classifiers in determining whether apps are safe or harmful. Furthermore, the computational assets expected for this investigation are sensible, making it useful no doubt worldwide implementation for device security assurance.

5. PROPOSED SYSTEM

Random Forest is a powerful and adaptable machine learning algorithm that wins arrangement and relapse tasks by combining the power of several decision trees. The middle idea behind Sporadic Boondocks is to build a 'backwoods' of choice trees, every one of which was prepared on unmistakable arbitrary subsets of the planning data and using unpredictable subsets of features. This randomness makes it easier to deal with the problem of overfitting and reduce the model's variance, both of which are common issues for individual decision trees. During the readiness stage, a bootstrap test of the information is utilized to construct every choice tree, and at each split in the tree, an unpredictable subset of components is considered. This communication guarantees that the woods' trees are different and mirror an assortment of the construction of the information. The Irregular Timberland arises after the tree woodland is laid out calculation makes forecasts by consolidating these singular's outcomes trees.

The last forecast in grouping assignments is made by greater part vote casting a ballot, in which each tree chooses a class name and decisions in favor of the mark with the most votes. Votes transform into the last estimate. The last expectation for relapse assignments is the average of each tree's forecasts. Since the combined expectations of many trees are less likely to be influenced by the commotion or particularities of each tree, this group approach improves the model's overall accuracy and robustness. Additionally, practitioners can identify which aspects have the greatest impact on predictions thanks to Random Forest's inherent measure of feature importance. This Arbitrary join power, interpretability, and exactness. For many purposes, backwoods is a well-known decision.

Advantages

- Digital weather forecasting Statistical
- Predict the climate
- Synoptic climate forecast
- Security is high
- Measures accuracy and execution time, ensuring reliable and efficient classification
- Provides a detailed analysis of classifier performance, aiding in selecting the best model for spam detection

6. SELECTED METHODODLOGIES

Decision Tree Algorithm

One supervised machine learning method that can be used for regression and classification tasks is the decision tree. Based on input features, it is a simple yet efficient decision-making method. The method

creates a structure that looks like a tree, where each internal node represents a feature or characteristic, each branch represents a decision rule based on that property, and each leaf node represents a class label or a numerical value (for regression).

Algorithm:

Step 1: Choose the relevant attributes that may influence the categorization and identify the target variable.

Step 2: Standardize the data to ensure that each attribute is on the same scale.

Step 3: Randomly set the weights and biases of the model to zero or a minimal initial value.

Step 4: Explain how the sigmoid and cost functions change any real number into a value between 0 and 1.

Step 5: Determine measures such as F1-score, recall, accuracy, and precision to assess the model's performance on the testing set.

Step 6: To improve the model's performance, adjust the hyper parameters, such as the regularization strength, learning rate, and number of iterations.

Step 7: Utilize the Decision Tree Algorithm model that has been trained to forecast fresh data and

Naïve Bayes

The Naïve Bayes algorithm is a supervised learning method for data classification that is based on the Bayes theorem. Text classification comprising a high-dimensional training data set is its main use case. One of the simplest and most effective classification techniques for building fast machine learning models with quick prediction capabilities is the Naïve Bayes Classifier. The probabilistic classifier uses an item's likelihood to create predictions. Articles about spam filtering, sentiment analysis, and categorization are a few well-known applications of Naïve Bayes algorithms.

Algorithm:

Step 1: Choose the relevant attributes that may have an impact on the categorization and identify the target variable.

Step 2: Standardize the data to ensure that each attribute is on the same scale.

Step 3: Initialize the model's parameters' weights and bias to zero or small, arbitrary values.

Step 4: Explain the sigmoid and cost function that convert any real-valued number to a number between 0 and 1.

Step 5: Determine parameters like F1-score, recall, accuracy, and precision to assess the model's performance on the testing set.

Step 6: To improve the model's performance, adjust the hyper parameters, such as the number of iterations and the intensity of regularization and learning rate.

Step 7: Make predictions on new data using the gradient boosting model that has been the trained data and categorize them as authentic or fraudulent job advertisements.

7. SYSTEM ARCHITECTURE

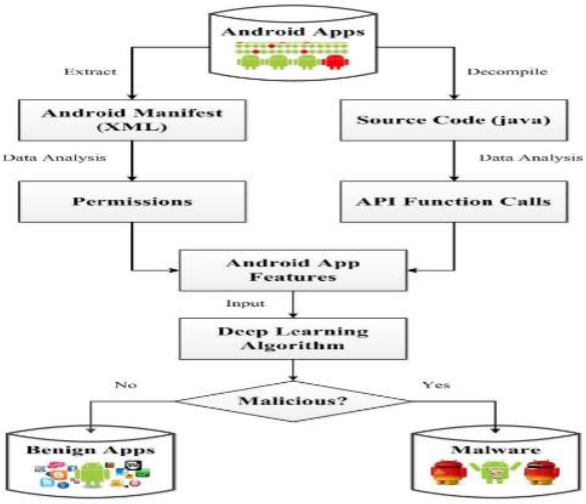


Fig 1: System Architecture

The significance of the requirements and the stated request for a serious degree of the device are related to how the product's general features are portrayed. Numerous web pages and their connections are described and designed during architectural design. Key software components are defined, broken down into processing modules and conceptual records systems, and the connections that exist between them are explained. The proposed framework characterizes the accompanying modules.

8. SYSTEM MODULES

Application Download

This session involves pre-downloading a number of apps from the Google Play Store onto a mobile device. Finding and analyzing malware in different applications is the main objective after installation. Applications that seem authentic may conceal malware, but upon deeper examination, they may display detrimental features like network manipulation, battery waste, or unauthorized data access. To find harmful activities that might not be immediately obvious to the user, the method starts by scanning these applications.

Although a lot of security solutions concentrate on static or dynamic analysis methods, they frequently have trouble identifying sophisticated malware varieties that employ evasive tactics or conceal their activities. At this point, the emphasis switches to a different strategy: identifying malware by looking at the application's network traffic patterns. Because malware frequently communicates with external servers or conducts odd data transfers that diverge from the app's typical behavior, such techniques are effective.

Android Application Scan

Many of the programs in this module have already been downloaded to a mobile device via the Play Store. The objective is to find any possible malware that may be present in these programs after they are installed. It can be especially difficult to detect malware on mobile devices using conventional techniques like signature-based detection or standard static or dynamic analysis techniques. Malware that employs

polymorphic approaches or behaves differently depending on environmental factors may go undetected by these conventional methods.

This method's main innovation is its emphasis on network traffic analysis as the main detection method. It is feasible to create a pattern of the app's typical activity by keeping an eye on its network interactions, including data transmitted to and received from external services. Then, using this pattern as a baseline, any anomalies that might point to malicious activity—like odd network communication or attempts to connect to untrusted servers—are identified.

Malware Detection

One approach is put forth in this proposed system to assess the security of mobile applications using data mining and cloud computing platforms in order to increase app security. The assignment of Malware identification will be divided into three categories: analysis, classification, and detection and eventual malware containment.

Application of the Classification Algorithm:

Using a dataset that lists this module makes use of a machine to obtain permissions from various Android apps developing classification algorithms for identifying dangers applications. It processes the categorized training and testing permissions various classifiers

Precision and Execution Appraisal:

This module assesses the precision with which the classification algorithms differentiate between destructive and safe applications. It also measures how long it takes to complete the examination, guaranteeing the strategies are productive and functional for genuine world use.

Classifier Comparative Analysis:

The final module examines and compares how well various classification algorithms work. It gives experiences into which classifiers are best at recognizing destructive applications, directing future enhancements and executions.

Android Permission Check

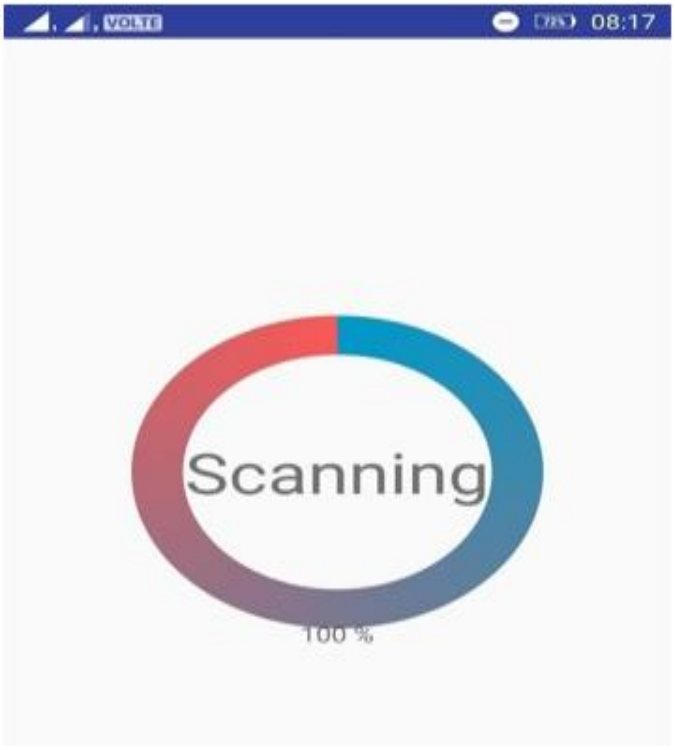
When a user wants to install and use a third-party app, they often do not know the importance or meaning of the permissions that the app asks for, so they just grant them all. This allows malicious apps to be installed and carry out their malicious activities in the background. An application must obtain the necessary authorization before using resources or data outside of its own sandbox. Listing the permission in the app manifest and asking the user to authorize each one at runtime is how you indicate that your app requires a permission. Every time you carry out an action that calls for a risky permission—which your app may require—you must verify that you have it.

9. RESULTS

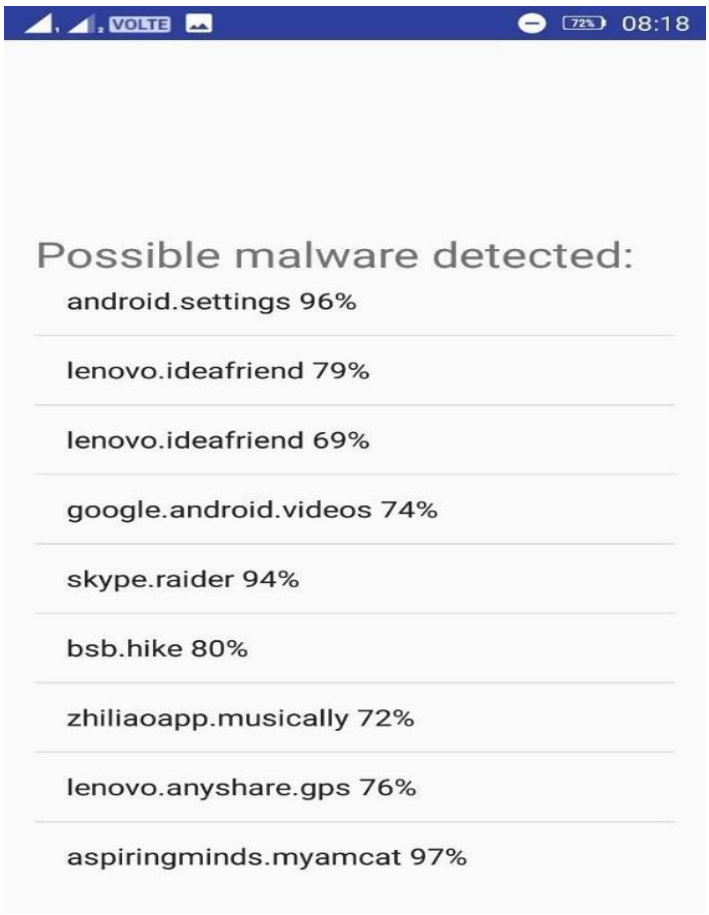
Home Page

Initial Process

Scanning Process



Malware Detection Process



Result Page

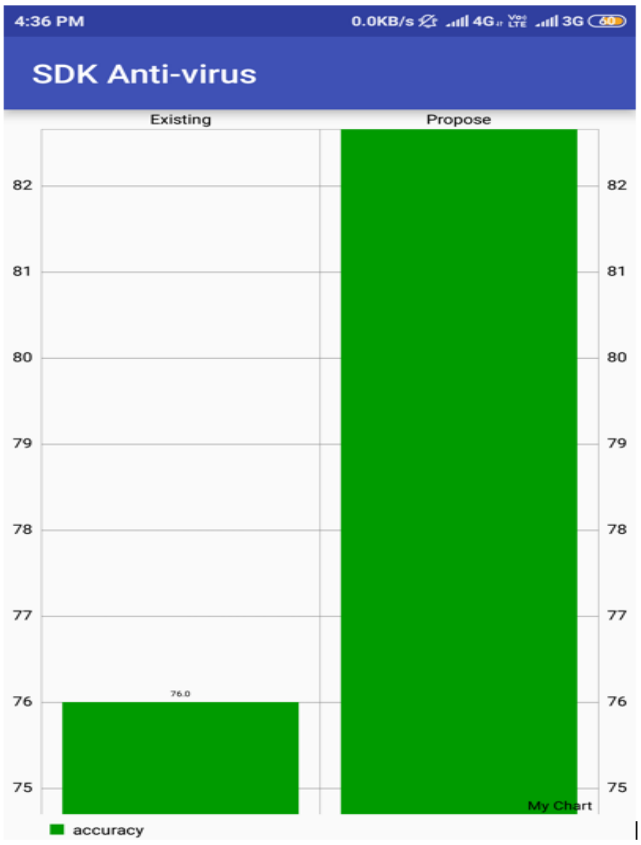


Fig 2: Comparison between Existing and Proposed System

The task involves identifying a list of third-party installed applications on an Android device and extracting their permissions, including the android: protection level (Normal or Dangerous) for each permission. Using an Android app permissions dataset, classification algorithms are applied to identify harmful applications. Both the execution duration and the malware classification accuracy are measured. The suggested classifiers and the current ones are compared, analyzing their performance in terms of accuracy for identifying harmful and normal apps. This helps assess the efficiency and reliability of the classification models used for malware detection

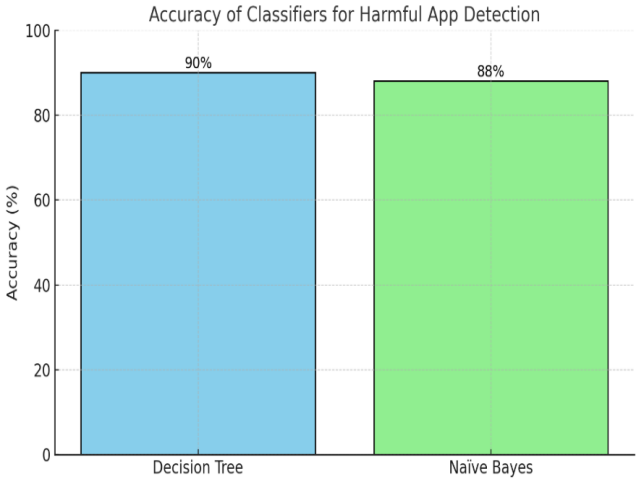


Fig 3: Comparison of Accuracy of Decision Tree and Naive Bayes

Decision Tree (90%) This algorithm performs the best among the two, achieving a high accuracy of 90%. Decision Trees are effective in classifying based on a series of decision rules, which is suitable for distinguishing between legitimate and malicious behaviors in Android apps. Naïve Bayes (88%) Slightly lower than the Decision Tree, Naïve Bayes performs well, reaching 88% accuracy. It's based on probability and works well with smaller datasets or when features are independent, but may not capture complex patterns as effectively as the Decision Tree.

10. CONCLUSION

The promise that Android Trojans can be identified using machine learning. Nonetheless, persistent exploration is vital for tackle current difficulties and improve detection capabilities in the mobile industry, which is always changing cybersecurity. As portable dangers develop, there a requirement for further developed strategies that can adjust and remain in front of new types of malware. By improving models for machine learning and evaluating new data sources, the future our ability to identify and eliminate Android Trojans can be improved through efforts actually, shielding cell phones and client information from arising security chances.

Given the shortcomings of conventional signature-based approaches, machine learning (ML) techniques for Android Trojan detection show great potential for the future. Based on behavioral and structural patterns in Android apps, machine learning (ML), specifically supervised learning (e.g., DT, SVM), unsupervised learning (e.g., clustering, auto encoders), and DL (e.g., CNNs, RNNs), is being used more and more to identify Trojans that have not yet been discovered. By analyzing both dynamic and static features such as runtime API calls and app code—these techniques make it possible to identify malicious activity more precisely. As malware develops, machine learning techniques such as reinforcement learning for dynamic adaptation will be essential for enhancing Trojan detection and thwarting new threats.

REFERENCES

1. L. Gang and Y. Wen, “Research on Clue Mining in Criminal Cases of Smart Phone Trojan Horse under the Background of Information Security,” Journal of Robotics, vol. 2022, pp. 1–11, Feb. 2022, doi: 10.1155/2022/7568110.
2. F. Zareen and R. Karam, “A Framework for Detecting Hardware Trojans in RTL Using Artificial Immune Systems,” Behavioral Synthesis for Hardware Security, pp. 265–287, 2022, doi: 10.1007/978-3-030-78841-4_12.
3. A. Attkan and · Virender Ranga, “Cyber-physical security for IoT networks: a comprehensive review on traditional, blockchain and artificial intelligence based key-security,” Complex & Intelligent Systems 2022, pp. 1–33, Feb. 2022, doi: 10.1007/S40747-022-00667-Z.
4. R. Sánchez-Salmerón et al., “Machine learning methods applied to triage in emergency services: A systematic review,” International Emergency Nursing, vol. 60, p. 101109, Jan. 2022, doi: 10.1016/J.IENJ.2021.101109.
5. I. Tiddi and S. Schlobach, “Knowledge graphs as tools for explainable machine learning: A survey,” Artificial Intelligence, vol. 302, p. 103627, Jan. 2022, doi: 10.1016/J.ARTINT.2021.103627.
6. K. Dushyant, G. Muskan, Annu, A. Gupta, and S. Pramanik, “Utilizing Machine Learning and Deep Learning in Cybesecurity: An Innovative Approach,” Cyber Security and Digital Forensics, pp. 271–293, Feb. 2022, doi: 10.1002/9781119795667.CH12.

7. H. Salmani, “Gradual-N-Justification (GNJ) to Reduce False-Positive Hardware Trojan Detection in Gate-Level Netlist,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2022, doi: 10.1109/TVLSI.2022.3143349.
8. N. M. Chayal and N. P. Patel, “Review of Machine Learning and Data Mining Methods to Predict Different Cyberattacks,” Lecture Notes on Data Engineering and Communications Technologies, vol. 52, pp. 43–51, 2021, doi: 10.1007/978-981-15-4474-3_5.
9. Mahindru, A., Sangal, A.L. MLDroid—framework for Android malware detection using machine learning techniques. Neural Comput & Applic 33, 5183–5240 (2021). Android Malware Detection using Machine learning: A Review 19.
10. Al-Ofeishat, H. A. (2024). Enhancing Android Security: Network-Driven Machine Learning Approach For Malware Detection. Journal of Theoretical and Applied Information Technology, 102(2), 737-750.
11. Jyothish, A., Mathew, A., & Vinod, P. (2024). Effectiveness of machine learning based android malware detectors against adversarial attacks. Cluster Computing, 27(3), 2549-2569.
12. Alhamri, R. Z., Cinderatama, T. A., Eliyen, K., & Izzah, A. (2024). Supervised Learning Methods Comparison for Android Malware Detection Based on System Calls Referring to ARM (32-bit/EABI) Table. Journal of Information Technology and Cyber Security.
13. Hareram Kumar and Prof. Sarwesh Site. "Android Malware Prediction using Machine Learning Techniques: A Review." International Journal of Recent Development in Engineering and Technology, vol. 11, no. 2, Feb. 2022.
14. Neamat Al Sarah, Fahmida Yasmin Rifat, Md. Shohrab Hossain, Husnu S. Narman. An Efficient Android Malware Prediction Using Ensemble machine learning algorithms. Procedia Computer Science, Volume 191, 2021, Pages 184-191. doi:10.1016/j.procs.2021.07.023.
15. R. Feng, S. Chen, X. Xie, G. Meng, S. -W. Lin and Y. Liu, "A Performance-Sensitive Malware Detection System Using Deep Learning on Mobile Devices," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 1563-1578, 2021, doi: 10.1109/TIFS.2020.3025436.
16. Nasri, Nuren Natasha Maulat, Mohd Faizal Ab Razak, RD Rohmat Saedudin, Salwana Mohamad, and Ahmad Firdaus Asmara. "Android malware detection system using machine learning." International Journal 9, no. 1.5 (2020).
17. Kouliaridis V, Kambourakis G. A Comprehensive Survey on Machine Learning Techniques for Android Malware Detection. Information. 2021; 12(5):185. https://doi.org/10.3390/info12050185.
18. Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2024). Android Malware Detection and Identification Frameworks by Leveraging the Machine and Deep Learning Techniques: A Comprehensive Review. Telematics and Informatics Reports, 100130.
19. Pathak, A., Barman, U., & Kumar, T. S. (2024). Machine learning approach to detect android malware using feature-selection based on feature importance score. Journal of Engineering Research.
20. Misalkar, H., & Harshavardhanan, P. (2024). Assessing the efficacy of Machine learning classifier for Android malware detection. Journal of Integrated Science and Technology, 12(4), 788-788.
21. Xie, W., & Zhang, X. (2024, January). The Application of Machine Learning in Android Malware Detection. In 2024 4th International Conference on Neural Networks, Information and Communication (NNICE) (pp. 1-4). IEEE.

22. Fatima, A.; Kumar, S.; Dutta, M.K. Host-Server-Based Malware Detection System for Android Platforms Using Machine Learning. In Advances in Computational Intelligence and Communication Technology; Springer: Singapore, 2021; pp. 195–205.
23. Guerra-Manzanares, A.; Bahsi, H.; Nõmm, S. KronoDroid: Time-based Hybrid-featured Dataset for Effective Android Malware Detection and Characterization. Comput. Secur. 2021, 110, 102399.
24. Jannath Nisha, O.S.; Mary Saira Bhanu, S. Detection of malicious Android applications using Ontology-based intelligent model in mobile cloud environment. J. Inf. Secur. Appl. 2021, 58, 102751.
25. Mathur, A.; Podila, L.M.; Kulkarni, K.; Niyaz, Q.; Javaid, A.Y. NATICUSdroid: A malware detection framework for Android using native and custom permissions. J. Inf. Secur. Appl. 2021, 58, 102696.