Journal of Integrated
SCIENCE & TECHNOLOGY

Blockchain for decentralized malware detection on android devices

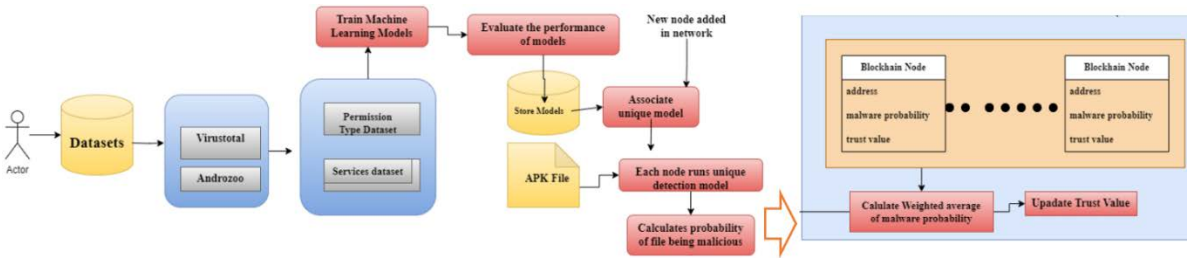
Praful R. Pardhi[1,2*], Jitendra Kumar Rout[3], Pratik K Agrawal[4*], Niranjan Kumar Ray[1]

[1]School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, India. [2]School of Computer Science & Engineering, Shri Ramdeobaba College of Engineering and Management, Ramdeobaba University, Nagpur. India. [3]Department of Computer Science & Engineering National Institute of Technology Raipur Chhattisgarh, India. [4]Symbiosis Institute of Technology, Nagpur Campus, Symbiosis International (Deemed University), Pune, Maharashtra, India.

Received on: 21-Oct-2024, Accepted and Published on: 04-Feb-2025

Article

ABSTRACT



Standard antivirus software falls short to address quickly the escalating Android smartphone malware threats emerging from their increased adoption. In light of this, the paper investigates the revolutionary idea of "Decentralized Malware Detection for Android Using Blockchain." A decentralized malware detection network powered by blockchain technology provides flexible protection while using the same framework originally intended for cryptocurrencies. The new approach reduces erroneous detection patterns markedly because current solutions still need regular software updates and have weaknesses against avoidance strategies. Our proposed work utilized Androzoo dataset which includes information on more than five million applications. We isolated data using Virus total VT scores to develop our own feature dataset which contained both permission-based and network service-based attributes. Random Forest along with Decision trees and Adaboost and K-NN classification methods are utilized by blockchain decentralized nodes to achieve superior malware detection accuracy. Analysis of sufficient Android application features enables a malware detection success rate of 90%. Results from the proposed work expose how blockchain technology improves Android malware detection to strengthen mobile apps security and raise mobile ecosystem resilience.

Keywords: Malware detection, permissions, cybersecurity, APK, machine learning, classification, blockchain, decentralized.

INTRODUCTION

Malware as defined by the abbreviation of malicious software is becoming a constant threat to computer systems hence the need to come up with strategies on analyzing, categorizing and controlling such program. The enhancement of the modern threats like viruses, worms, trojans, ransomware, spyware, and adware becomes rather sophisticated, and thus, requires the improvement of the methods to defend and secure the computer networks, as well as the data they contain. Since malware development is changing over time and adapting to new technologies and security solutions it is crucial to continuously defend ourselves and our organizations against the menace of malware. This defense involves strict security solutions; for example, anti-virus, frequent updates of the operating system and other software, safer web use, and security of computer networks. As a result, Android users have more complex malware detection and protection mechanisms than the original iOS users. All existing security solutions have many disadvantages, even if they are effective in the detection and minimization of malware threats: they can be easily evaded, and their effectiveness decreases with time and constant updates. To address these difficulties, the development of decentralized malware detection systems for Android based on the blockchain technology has received attraction.[18,19] A technology that was initially designed for use in cryptocurrencies such as Bitcoin, blockchain has evolved and can now be applied in a variety of industries. Some of the inherent properties including decentralization, transparency, and immutability make it as suitable means of enhancing security of Android devices. This technique significantly reduces cases of false positives and negatives since the detection is done in several nodes

*Corresponding Author: Praful Pardhi, Pratik K Agrawal, Email: pardhipr@rknec.edu, pratik.agrawaal@gmail.com

Cite as: *J. Integr. Sci. Technol., 2025, 13(4), 1084. DOI:106211/sciencein.jist.2025.v13.1084*

on the network, all checking the Android applications for their integrity. In this paper, it is described the aim of this new approach, as well as the steps that were followed in the context of this research, the methods used for example to collect substantial amounts of data from different sources, the process of extracting the features through the "Androzzo" tool, the construction of the services dataset for the machine learning, and the development of powerful approaches like the machine learning. The process involves storage and linking of the models with definite nodes on the blockchain dependable system offering a distributed detection procedure that lessens the vulnerability of a single susceptible point. The last two steps include probability computation and changes in the blockchain, which also helps in understanding how the utilization of blockchain enhances the identification of Android malware and strengthens security and reliability in the mobile environment

The solution is decentralised, there's no single control from one single authority over the network or over the MALWARE DETECTION process. Machine learning used in the system enables the system to learn new patterns of malware and be able to respond quicker to new threats than with the signature-based approach. Trust-based system: The solution employs the concept of trust to rank performance of each of the nodes in the network. This is critical in making sure that the system is always operating on the most correct, and up to date, predictions.

Malware is a colossal danger to people, corporations, and governments as it results in financial misfortunes, loss of important information, organizational interferences, and harm to the organization's reputation. Therefore, tremendous methods should be used for the detection and identification of different varieties of malwares.

The key objectives of the proposed work are as follows:The key objectives of the proposed work are as follows:

- Enhancing Android Security: Therefore, the main aim of this research is to design a decentralized Anti-Malware framework to improve Android device security through the integration of the blockchain technology. It has the intended strategy of addressing a better shield of preventing and defeating malware threats resulting in less possibility of an attack.
- Leveraging Blockchain Technology: In so doing, the following study will seek to examine how the use of block chain technological innovation may be accomplished. The principles of efficiency consist of the aspects related to blockchain, including decentralization, transparency, and the presence of an unchangeable ledger.
- Eliminating Central Points of Failure: In so doing, the research will seek to eliminate the centralized point of vulnerability that is well known to plague normal antivirus facilities. This gives the workload division which makes the systems difficult to penetrate and reduce the possibility of breaches.
- Privacy-Preserving Solutions: The research also seeks to address the issue of privacy and security by designing the system in such a way that does not compel people to provide some information that may be considered as private. There is another advantage of enhancing the detection of malware, and that is the user does not have to compromise privacy.
- Minimizing False Positives and Negatives: Decentralized malware detection employing blockchain technology may be able to minimize false positives and false negatives by employing a distributed consensus mechanism. This objective is crucial for improving the accuracy of malware detection.
- Real-Time Threat Detection and Scalability: This study looks into how scalable blockchain- based malware detection is and how it can give Android users real-time threat detection, making it a dependable and practical solution.

LITERATURE REVIEW

The various work on detecting android malware to tackle real life cyber-threats are discussed in this section.

New dataset which is Malware dataset was introduced to researchers through the works of Hreirati and Iqbal.[1] The dataset is composed of 5,000 applications belonging to ulterior malware categories, as taken from a pool of more than 5 million apps. Namely, according to the authors, the goal of this dataset is to assist the research community by offering them examples of malware in the real world.

Dogru and kiraz[14] This research study proposed a new system, Web-Based Android Malicious Software Detection and Classification System for detecting and categorising the malicious soft ware. This system employs static analysis, web scraping technique and client server architecture as its components. The performance analysis of the developed system was done and it recorded 97% success level. 62 % is achieved in identifying both benign and malicious datasets. Analytical methods used in the key performance indicators include the static, dynamic and hybrid statistical methods.

Iqbal et al.[3] present a solution in the form of a security tool that would help in avoiding the misuse of application permissions thereby ensuring that the information of the users is well protected. As for the tool, it is concerned with tracking application sharing the same user ID and offers ideas on how this can be protected.

Catak and Yazi[4] have done research work where feat for known malware which collected data by executing them in a sandbox. The researchers then analyzed these recorded actions and categorized the malware into eight main families: Trojan, Backdoor, Downloader, Worms, Spyware and Adware, Dropper and Virus.

D'Angelo, et al.[5] the main goal is to introduce a new dataset called the Unisa Malware Dataset (UMD), which focuses on capturing different characteristics of malware programs through static and dynamic features. The authors conducted experiments using various machine learning and deep learning techniques to demonstrate the effectiveness of the proposed dataset in training AI-based models for malware classification.

Ehsan , et al.[6] review and explores different methods used to detect malicious software (malware) on Android devices by analysing the permissions requested by apps.

Shehata, et al.[7] proposed approach that aims to identify malware and benign applications in Android by analyzing the permissions extracted from the AndroidManifest.xml file.

HarshaLatha and Mohanasundaram[8] suggest a way to detect malicious software on Android devices. The authors focused on

improving the accuracy of malware detection by analyzing the behavior of these harmful programs while they run on devices.

Singhal and Raul[9] proposes a new and advanced antivirus system that can not only scan files for viruses, but also learn and identify files that may be potential threats. The system achieves this by analyzing the actions performed by different types of safe and harmful programs, and using machine learning techniques to determine the security risk level of files.

A survey by Gibert, et al.[10] aims to give a comprehensive overview of machine learning techniques used for detecting and classifying malware, with a specific focus on deep learning methods. It provides a step-by-step explanation of the methods and features typically used in traditional machine-learning workflows for malware detection.

The article by Chen et.al.[11] introduces an innovative deep neural network-based approach for Windows malware detection, emphasizing full process information. It overcomes limitations of traditional security solutions by leveraging inner-process and inter-process behavior awareness, constructing process graphs, and implementing deep learning. The method outperforms naive models, demonstrating robustness against adversarial attacks and concept drift. In summary, the paper advances malware detection using deep learning and comprehensive process-aware behaviors.

The research by R. Vinayakumar[12] addresses malware-related security breaches by introducing the "ScaleMalNet" framework, combining deep learning with static, dynamic, and image-based analyses for malware detection. Extensive experiments validate the framework's effectiveness compared to traditional methods. The paper identifies future research areas and offers a hybrid deep learning solution for real-time malware detection.

Akhtar and Feng [13] present a research on dynamic malware detection, which underscores behaviour-based detection that includes machine learning algorithms; the latter automates the process. This research calls for the use of machine learning and the behavior based analysis in the detection of advanced malware which may even reduce false positives and the speed in which the detection is done.

Shantanu, Janet B and Joshua Arukl Kumar R et al devised proposal [14] that highlights the challenge of detecting frowned upon websites and URLs. It considers URL malicious classification as a binary classification problem and evaluates a number of AI based approaches like machine learning by using 0. 45 million URLs dataset with each different model performance. The accuracy and F1-score best websites show the Random Forest classifier to be one of the best in this case. The research notes how do we do that, suggesting that the better balance of the model's training data and a feasible method for predicting and fighting malicious URLs can be briefly mentioned.

The research tries to tackle the increasing malware problem by training an ML-based model to classify down-loaded files as malicious or clean. Here, some notable features like MD5 hash, the Optional Header size, and the Load Configuration Size are particularly chosen as descriptors. The Random Forest Classifier scores high accuracy, close to 100% as much as to say so. 99% on the test dataset, followed by XGBoost at 99. Among them, the majority is 68%. This leads to information about the analysis comparing with previous researches, indicating that the accuracy is enhanced, providing an effective method for detecting malware types that can destroy user systems and so on.

Bilot[16] his state-of-the-art survey paper addresses the dynamic nature of malware detection by highlighting the role of ML and DL in evaluating how existing solutions can be increased by considering state-of-the-art concepts. It highlights the relatedness of graphs and the usage of Graph Neural Networks (GNNs) , which are known as graph guaranteed network. This paper discusses and presents the works that have been utilized, showing the role of GNNs and word embedding in decoding of source code and structuralizing malware. It also describes attacks on graph-oriented detection methods, and it gives an outlook to the possible role of graph ML in malware detection research endeavors.Selamat[17] This paper conducts a comprehensive literature review and taxonomy of machine learning methods for malware detection, addressing issues of accuracy and malware identification based on 77 research works. It emphasizes the growing role of machine learning in cybersecurity, highlighting the impact of larger datasets and the effectiveness of behavior-based classification and dynamic or hybrid analysis types in malware detection. The study suggests future research directions related to feature extraction, classification methods, and analysis types.

Raje et al.[18] This paper describes on peer-to-peer blockchain based firewall model for classification of Portable Executable (PE) files as either being malicious or benign in nature through implementation of deep belief neural network (DBN) employing a dataset containing 10,000 files. The paper demonstrates the opportunity of automated heuristic malware detection and supports the paradigm of combining various technologies in cybersecurity field. Future work seeks to improve the system scalability and analysis of the dimensionality of data.

Sheela S et al. [19] With this research, the authors present a new model on malware detection employing the blockchain technique for detecting new malware files which is a combination of both signature and behavioral based systems. It positively impacts security by providing high detection rates and minimal false positives and it uses blockchain. Some of the possible future work include the use of artificial intelligence, machine learning, real-time monitoring of the systems as well as automating the process of updating to new signatures for more effective detection of threats and enhancing the accuracy of the system.

ARCHITECTURAL FRAMEWORK IMPLEMENTATION

The proposed work provides a new and clear strategy of malware detection and prevention from the blockchain and machine learning perspective. These are system architecture are illustrated in the figure 1 below. Malware detection process is therefore a multi-step process.

It involved leveraging Androzoo's extensive dataset of over five million Android apps, obtained from various sources including Google Play and F-Droid. To manage the data, the 'latest.csv' file was streamlined to 2000 applications by categorizing 1000 with a VT detection value over 40 as malware and 1000 with a value of 0 as benign. Following this, we meticulously crafted permission type and services datasets by extracting sha256 values and conducting

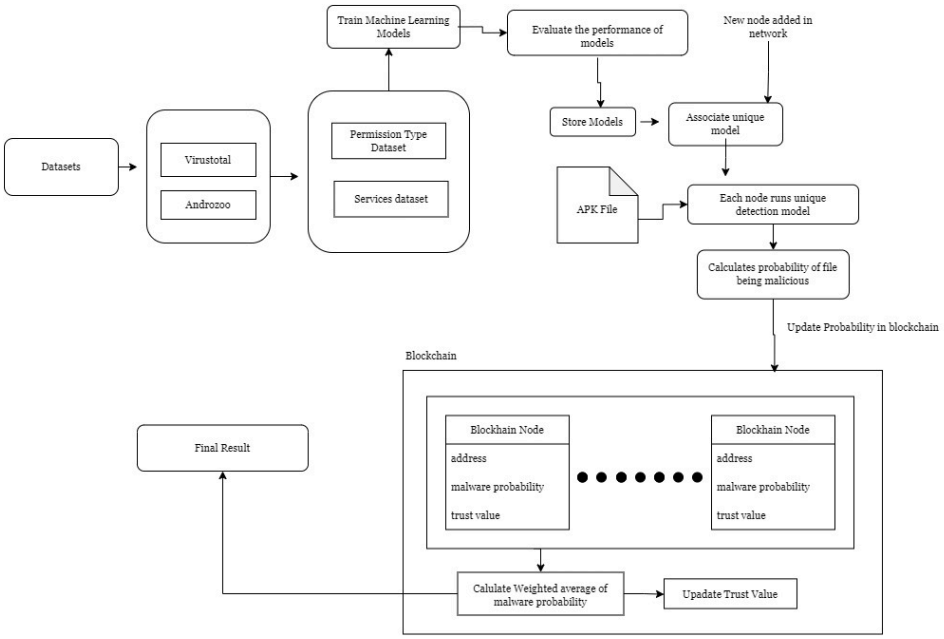


Figure 1. Proposed Methodology

static analysis through VirusTotal, storing the results in separate CSV files. Subsequently, our method delved into the quest for the most accurate classification algorithm, employing various supervised techniques on these datasets and documenting resulting accuracies along with model IDs in the database.

Within the blockchain network, our approach dictated that each new node integrates with a uniquely trained ML model boasting the highest accuracy, featuring a node address, percent malware probability, and an initialized trust value. During APK file analysis, all nodes independently execute their detection models, updating their malware probability values in the blockchain. The resultant probability presented to the user is determined through a weighted average. Post-detection, our method involves adaptive adjustments to node trust values based on deviations from the resultant probability. Ensuring the blockchain's security, our implementation strategically utilizes Ethereum's smart contract capabilities, nullifying any unauthorized alterations within a node automatically. This approach guarantees the integrity of the information disseminated by the nodes.[18] In the next subsection we have discuss the functionality of each step in details.

3.1 Dataset Creation

Androzoo's Dataset: The official Google Play market, some open-source repositories like F-Droid, and other markets are all represented in the expanding library of Android apps known as AndroZoo. AndroZoo currently has more than five million apps available. [20] The writers additionally give a variety of details about each application in addition to the APK files. For instance, Androzoo utilizes VirusTotal to examine every application and include the number of antivirus programs that identify it as malware.[23]

Weekly database updates are also made by Androzoo. The Androzoo database is available as a CSV file download. The document's name is latest.csv. Numerous helpful properties, like the sha256 checksum values, package names, VirusTotal ratings, source marketplaces, and many more. [20,21,22] Figure 2 shows the overview of the dataset with the different features.

Pre-process the 'latest.csv' file: The size of the latest.csv is too large. Therefore, it was reduced to contain details of around 2000 applications only. This was done by selecting the 1000 rows with VT detection value greater than 40 and categorizing them as malware and 1000 rows with VT detection value 0 and categorizing them as benign.[20,23] Figure 3 shows the preprocessed dataset.

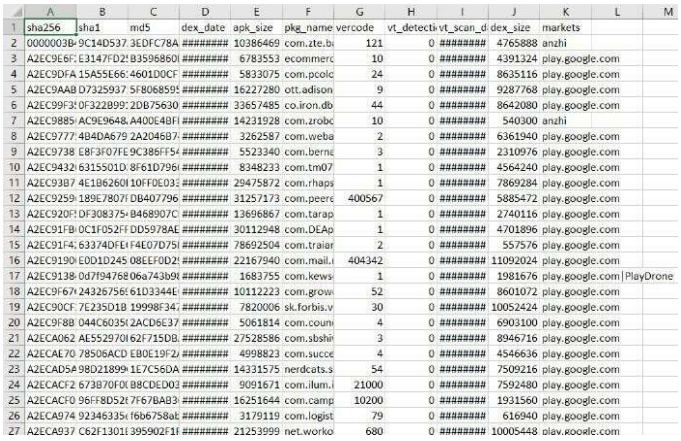
	A	B	C	D	E	F	G	H	I	J	K	L
1		sha256	sha1	md5	dex_date	apk_size	pkg_name	vercode	vt_detecti	vt_scan_d	dex_size	markets
2	3404	000A067D	991078B8	AC926221	########	3366203	com.depo	1	45	########	30768	unknown
3	8880	0019EDF7	C7A1B284	C4A08B89	########	1089531	com.keji.d	32	44	########	294484	praguard
4	16799	0030C868	00BB548D	06B536B2	########	3705113	pj.ishuaji	11	43	########	792808	VirusShare
5	30814	0059C516	BE21BEEC	71807353	########	3425085	pj.ishuaji	16	44	########	730908	VirusShare
6	46697	00882D7C	F41CAB38	813F42D3	########	334513	kyrgyzcha	1	43	########	150796	VirusShare
7	48116	008C3C11	0B91706E	CD807291	########	884724	com.keji.d	27	45	########	235996	praguard
8	108700	013E055F	E9FF24F5	0D135CB2	########	2142662	com.sffa.r	20405	44	########	1180240	VirusShare
9	114097	014E1753	7689495D	727B985E	########	3704626	com.lololc	1	55	########	9140	VirusShare
10	127239	017466F0	65443C2D	D1A55BE2	########	84033	nang.dv	1	43	########	14924	VirusShare
11	140737	019B34A0	CA1B727F	8542DEE6	########	295193	superman	81	44	########	177400	appchina
12	141258	019CBDBF	AE03FFD8	F9F7284A	########	24357	skb.az.gan	1	44	########	8888	VirusShare
13	147596	01AFCF5F	95966BF1	724338D7	########	208430	yhonvqb.c	1	43	########	82040	VirusShare
14	166947	01E90826	2305505F	94538DC5	########	219259	available.	70	43	########	178916	appchina
15	186179	0221511D	7B2B615A	8222FD66	########	218883	wdbpiewh	1	43	########	81708	VirusShare
16	227977	029BCA69	559D8F49	D64B482C	########	429699	available.a	98	43	########	174436	appchina
17	229671	02A0A48A	45D17878	B97874F4	########	64725	nang.dv	1	45	########	14924	VirusShare

Figure 2. Overview of Dataset collection

	A	B	C	D	E	F	G	H	I	J	K	L
1	sha256	developer	privacy	dangerous	normal	sms	system	money	signature	internet	gps	Category
2	5416F3C2I	0	3	3	2	2	0	1	0	1	1	Malware
3	6BFD5B82	0	2	2	2	0	0	0	0	1	1	Malware
4	8019767A	1	1	3	2	0	1	0	1	1	0	Malware
5	8C374ED0	0	1	5	3	0	0	0	0	1	0	Malware
6	6FD54D1D	1	2	4	2	0	1	0	1	1	1	Malware
7	9B619783	0	2	2	2	0	0	0	0	1	1	Malware
8	6BFD5B82	0	2	2	2	0	0	0	0	1	1	Malware
9	9B619783	0	2	2	2	0	0	0	0	1	1	Malware
10	8019767A	1	1	3	2	0	1	0	1	1	0	Malware
11	8C374ED0	0	1	5	3	0	0	0	0	1	0	Malware
12	8019767A	1	1	3	2	0	1	0	1	1	0	Malware
13	6BFD5B82	0	2	2	2	0	0	0	0	1	1	Malware
14	6FD54D1D	1	2	4	2	0	1	0	1	1	1	Malware
15	6FD54D1D	1	2	4	2	0	1	0	1	1	1	Malware
16	5416F3C2I	0	3	3	2	2	0	1	0	1	1	Malware
17	5416F3C2I	0	3	3	2	2	0	1	0	1	1	Malware
18	5416F3C2I	0	3	3	2	2	0	1	0	1	1	Malware
19	8C374ED0	0	1	5	3	0	0	0	0	1	0	Malware
20	6FD54D1D	1	2	4	2	0	1	0	1	1	1	Malware
21	6BFD5B82	0	2	2	2	0	0	0	0	1	1	Malware
22	8019767A	1	1	3	2	0	1	0	1	1	0	Malware
23	8C374ED0	0	1	5	3	0	0	0	0	1	0	Malware
24	6FD54D1D	1	2	4	2	0	1	0	1	1	1	Malware
25	9B619783	0	2	2	2	0	0	0	0	1	1	Malware
26	6BFD5B82	0	2	2	2	0	0	0	0	1	1	Malware
27	8C374ED0	0	1	5	3	0	0	0	0	1	0	Malware

Figure 3. Pre-Processed dataset

Creation of permission type dataset: From the dataset created in above step, the sha256 values were extracted and given to VirusTotal for static analysis. From the analysis result, permission types along with their count were selected and stored in the csv file.[23] Figure 4 shows the permission type dataset.



sha256	sha1	md5	dex_date	apk_size	pkg_name	vercode	vt_detecti	vt_scan_d	dex_size	markets	
0000003B	9C14D537	3EDFC78A	########	10386469	com.zte.bi	121	0	########	4765888	anzhi	
A2EC9E6F	E3147FD2	B3596860	########	6783553	ecommerc	10	0	########	4391324	play.google.com	
A2EC9DFA	15A55E66	4601D0CF	########	5833075	com.pcolc	24	0	########	8635116	play.google.com	
A2EC9AAB	D7325937	5F8068595	########	16227280	ott.adison	9	0	########	9287768	play.google.com	
A2EC99F3	0F322B99	2DB75630	########	33657485	co.iron.db	44	0	########	8642080	play.google.com	
A2EC9885	AC9E9648	A400E4BF	########	14231928	com.zrobc	10	0	########	540300	anzhi	
A2EC9777	4B4DA679	2A204587	########	3262587	com.weba	2	0	########	6361940	play.google.com	
A2EC9738	E8F3F07F	E9C386FF	########	5523340	com.berne	3	0	########	2310976	play.google.com	
A2EC9432	63155D1D	8F61D796	########	8348233	com.tm07	1	0	########	4564240	play.google.com	
A2EC9387	4E1862601	0FF0E033	########	29475872	com.rhaps	1	0	########	7869284	play.google.com	
A2EC9259	189E7807	DB407796	########	31257173	com.peere	400567	0	########	5885472	play.google.com	
A2EC920F	DF308375	B468907C	########	13696867	com.tarap	1	0	########	2740116	play.google.com	
A2EC91FB	0C1F052F	DD5978AE	########	30112948	com.DEAp	1	0	########	4701896	play.google.com	
A2EC91F4	63374DFE	F4E07D75	########	78692504	com.traiar	2	0	########	557576	play.google.com	
A2EC9190	E0D1D245	08EEF0D2	########	22167940	com.mail.i	404342	0	########	11092024	play.google.com	
A2EC9138	0d7f94768	06a743b9	########	1683755	com.kews	1	0	########	1981676	play.google.com	PlayDrone
A2EC9F67	24326756	61D3344E	########	10112223	com.grow	52	0	########	8601072	play.google.com	
A2EC90CF	7E235D1B	19998F34	########	7820006	sk.forbis.v	30	0	########	10052424	play.google.com	
A2EC9F8B	044C6035	2ACD6E37	########	5061814	com.couru	4	0	########	6903100	play.google.com	
A2ECA062	AE552970	62F715DB	########	27528586	com.sbshir	3	0	########	8946716	play.google.com	
A2ECAE70	78506ACD	EB0E19F2	########	4998823	com.succe	4	0	########	4546636	play.google.com	
A2ECAD5A	98D21899	1E7C56DA	########	14331575	nerdcats.s	54	0	########	7509216	play.google.com	
A2ECACF2	673B70F0	B8CDED03	########	9091671	com.ilum.i	21000	0	########	7592480	play.google.com	
A2ECACF0	96FF8D52	7F67BAB3	########	16251644	com.camp	10200	0	########	1931560	play.google.com	
A2ECA974	92346335	f6b6758a	########	3179119	com.logist	79	0	########	616940	play.google.com	
A2ECA937	C62F1301	395902F1	########	21253999	net.worko	680	0	########	10005448	play.google.com	

Figure 4. Permission type dataset

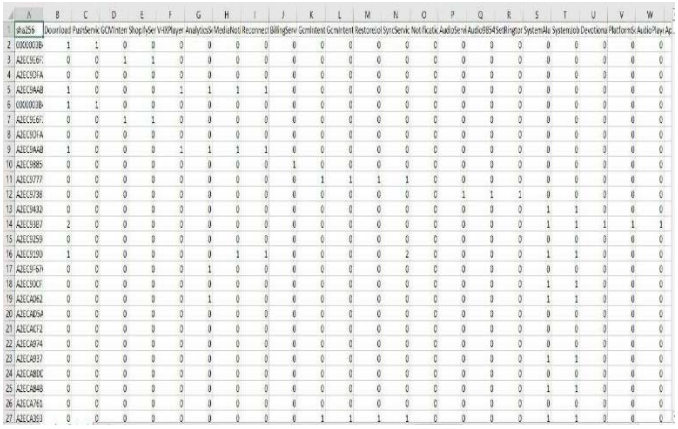


Figure 5. Services dataset

Creation of services dataset: From the dataset created in step (3), the sha256 values were extracted and given to VirusTotal for static analysis. From the analysis result, services along with their count were selected and stored in the csv file.[23] Figure 5. shows the Services dataset.

The next step is finding the classification algorithm that is more accurate than others.[13] Various supervised algorithm techniques like Decision Trees, Random Forests, AdaBoost, and K-Nearest Neighbor were used on the permission type dataset and services dataset. Accuracies of these models are stored in the database along with the model ID used by the blockchain network node for malware detection.[8,10,22,23]

Modeling Approach

The modeling approach for Android malware detection using machine learning involves the following steps:

• Data Collection: Collect a dataset of Android applications, consisting of both benign and malicious samples. The Androzoo dataset is in the form of CSV file (latest.csv).

• Data Preprocessing: Preprocess the dataset to extract features from the Android applications. Features could include permissions requested by the application, API calls made by the application, services utilized by the application or system calls made by the application.

• Feature Selection: Select the most relevant features from the preprocessed dataset. This step helps to reduce the dimensionality of the dataset, making it easier to train the machine-learning models.

• Model Selection: Choose an appropriate machine learning algorithm for the problem of Android malware detection. Popular machine learning algorithms used for this task include Decision Trees, Random Forests, AdaBoost, and K-Nearest Neighbor.

• Model Training: Finally, apply the selected machine learning algorithm on the dataset which has been preprocessed and feature selected. This step requires having to separate the data into training and testing set after which the machine learning model is trained on the training data.

• Model Evaluation: Check how well the ML model that has been trained from the training set to predict the testing set. The usual evaluation criterion which can be applied to assess the performance of the model is the measure that defines accuracy, precision, recall, F1-Score, etc.

In general, the main challenge arising while applying modeling approach for Android malware detection using machine learning is that the data set should go through certain preprocessing steps, relevant features should be selected and an optimum machine learning algorithm should be chosen to balance the accuracy and the time complexity of the model.[13]

IMPLEMENTATION

The proposed work starts with reducing the size of Androzoo dataset by creating two separate dataset called as permission & service type dataset. This multi process approach is discussed in details as follows.

A. Reducing the size of latest.csv:

Androzoo contains information about more than 22 lakh applications and the size of the latest.csv file is more than 5 GB.[1] The dataset size was reduced by selecting 1000 rows with vt_detection value > 40 and 1000 rows with vt_detection value = 0. Dask library of python was used to increase the speed of the execution. The new csv file now contains information of about 2000 applications. This dataset will serve as a base dataset that contains application permission features called permission type and other dataset contains information related to network services called services dataset. [6]

B. Creation of permission type dataset and services dataset:

To extract the permissions and category from VirusTotal all the sha256 values from the new csv file created in (A) were selected and stored in a list. Then by using a loop all the sha256 values were given to the virustotal for static analysis. This was done by using the virustotal API key. But virustotal allows only 500 requests per day and we require analysis of about 2000 applications. So multiple API keys were used. As soon as one API key gets exhausted, the next API key will be used to get the analysis of the next sha256 value. The data from VirusTotal was received in the form of JSON i.e key-value pair.[23]

VirusTotal includes in its static analysis all the types of permissions used and thier count. This information was extracted and saved in the permission type dataset csv file. In the same way, services information was extracted and saved in the services dataset csv file. Figure 2 shows the creation of dataset.

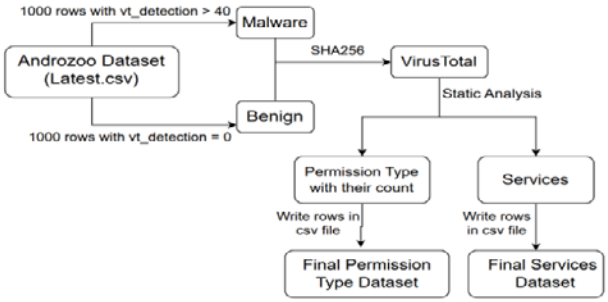


Figure 6. Creation of two datasets

After the data are pre-processed, it is fed to the models for training and testing. A dataset is split into 70% training and 30% testing. The training data is given to the model and features are recognized and classified based on labels. Different AI based approaches like machine learning are trained on two datasets: (i) permission type dataset and (ii) services dataset and the performance of models is then evaluated. Models used were: (i) Decision tree (ii) Random Forest, (iii) AdaBoost and (iv) K nearest neighbors.[8,13]

Resultant probability calculation in blockchain:

When an apk file is submitted for analysis, all the nodes will run their unique detection model and update the probability in blockchain. The weighted average of all the probabilities with the trust value is calculated [18] using the Equation 1:

$$\mu = \frac{\Sigma probability[i].trustValue[i]}{\Sigma trustValue[i]} \quad (1)$$

where, μ - weighted average
∑ - Summation

This resultant probability is the final result which gives the probability of the file being malicious. The trust value of each node is updated after each detection[18] by using the Equation 2 & 3 respectively:

deviation = final_probability – node_probability (2)

new_trust = (old_trustValue + 100 - deviation)/2 (3)

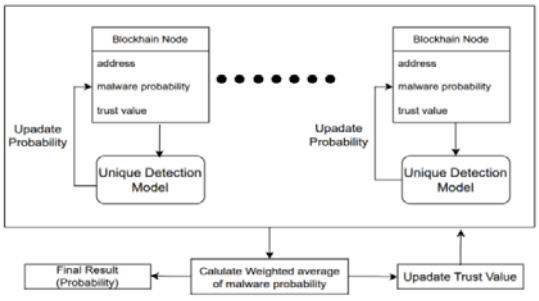


Figure 7. Shows the resultant probability calculation

Ethereum's innovative smart contracts are executed through the Ethereum Virtual Machine (EVM), providing a robust mechanism for self-executing and self-enforcing protocols. These smart contracts play a pivotal role in enforcing predefined agreements, including event responses, within the blockchain ecosystem. To implement these smart contracts on the Ethereum network, we have employed the Solidity programming language, thereby ensuring the integrity and reliability of our system.[19]

The implementation of our system was carried out on a testnet which used ganache, a nodeJS-based client developed to mimic the ethereum blockchain network. Ganache does a good job of organizing a localized p2p network with ten specific accounts, with each node of the network being responsible for one of the accounts. Using accounts, these smart contracts work together to determine the characteristics of a particular file.

Every action and event occurring within this network, such as external file downloads, trust value updates, and more, undergoes a rigorous process of hashing and is then securely stored in the blockchain as transactions, often referred to as ledgers. This meticulous record-keeping mechanism ensures the integrity and transparency of the entire system

RESULTS AND DISCUSSION

The modeling step was done with Decision Trees, Random Forest, AdaBoost, and K-nearest neighbors training and testing on two datasets. The datasets were split in the ratio of 70:The K-Folds method, as in iiii;, we shall have, for instance, 70% training and 30% testing[10]

The following evaluation metrics were used to assess the performance of the four ML models:

- Accuracy: The percentage of data points that were correctly classified.
- Precision: The percentage of predicted positive cases that are actually positive.
- Recall: The percentage of actual positive cases that are predicted correctly.
- F1 Score: A harmonic mean of precision and recall.

It is important to note that the performance of ML models can vary depending on the specific dataset and the hyperparameters used to train the model. Therefore, it is important to evaluate multiple ML models on different datasets before selecting a model for deployment. It is also important to note that ML models can be improved over time by retraining them on new data or by using techniques such as transfer learning.[15] The implementation results on the permission & service dataset for 627 & 655 android APK file for the respective dataset are analyzed and the performance metrics are calculated.

1) Permission Type Dataset: This dataset consists of the type of all the possible permission along with their count for each sha256 value. The testing was done on total 627 apks. Figure 8 shows the confusion matrix of all the algorithms.

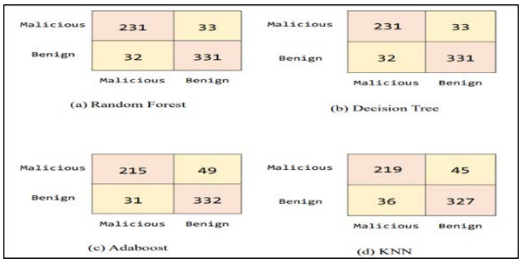


Figure 8. Confusion matrix for all the algorithms (Permission Type Dataset)

F1_score, accuracy, recall, precision, and specificity are calculated based on the above confusion matrix. Table 1 shows the result for all the algorithms for permission type dataset.

Table 1. Metric Of Ml Models On Permission Type Dataset

Metric	Algorithms			
	Random Forest	***Decision Tree***	***AdaBoost***	***KNN***
F1_score	87.666034	87.666034	87.666034	87.666034
Accuracy	89.633174	89.633174	87.240829	87.081340
Recall	87.832700	87.832700	87.398374	85.882353
Precision	87.500000	87.500000	81.439394	82.954545

2) Services dataset: This dataset consists of all the possible services along with their count for each sha256 value. The testing was done on total 653 apks. Figure 9 shows the confusion matrix of all the algorithms.

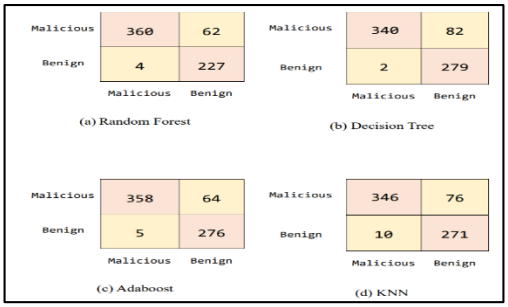


Figure 9. Confusion matrix for all the algorithms (Services Dataset)

F1_score, accuracy, recall, precision, and specificity are calculated based on the above confusion matrix. Table 2 shows the result for all the algorithms for services dataset.

Table 2. Metric of ml models on Services Dataset

Metric	Algorithms			
	Random Forest	**Decision Tree**	**AdaBoost**	**KNN**
F1_score	91.603053	91.603053	91.603053	91.603053
accuracy	90.611664	88.051209	90.184922	87.766714
recall	98.901099	99.415205	98.622590	97.191011
precision	85.308057	80.568720	84.834123	81.990521

APK Analysis has two ways for users to select files: a direct path or the apk file sha256 hash value. Then, they can go to “Next” option to click on “Upload” for the file to be extracted. The feature extraction process involves extracting relevant features from the APK file, such as: (i) request type accompanied with the number of them and (ii) aps management.

Extracted features which are then used to determine the taxonomy of the APK file from the blockchain with an increment of the probability of APK files being malicious. That can be accomplished by means of machine learning algorithms irrespective of which technique you opt. For getting the relevant probability, users just use the main button that reads “Find Probability” on the GUI and perform virus detection. The chance that the file is bad will be shown on the machine with a percent probability.Apart from that users can also add or remove a node from the network by providing the node address at any point of time. The example of resultant probability detection is shown in Figure 10.

Figure 10. Example of Finding the probability of an apk file being malicious

CONCLUSION

One main reason is the fact that these existing problems, which are partly because of Android’s proprietary architecture and its overwhelming popularity among mobile users leading to a wide number of users, could be imputed to that. This mounting security fallout is overall just a tip of the iceberg as there is still a lot of room to enhance security. We have shown how the hidden malware code is readily concealed in the package files of mobile devices for Android applications. Research clearly shows that Android device protection has been enforced with decentralized malware detection through Android blockchain, which is a new and unimaginable proposition to strengthen this security level. There are at least several examples of directions where research innovation of this field can be driven.

Android device and app progression have the biggest worth in scalability, yet this remains a challenge. This is something which is feared by faces with the prospect of understanding the application in decentralized detection system that could smoothly manage to grow to a large user base without sacrificing the operational efficiencies. These raise to consider the preservation of privacy as well. This has led to the creation of these platforms capable of safeguarding and preserving privacy and security of data. Privacy assurance technologies improvement will increase security system and user confidence. Subsequently, the experience of AI-based techniques' performance involving machine learning with varying permission and service levels business speak could has the invalueable lesson. It highlights, in particular, the fact that rich feature sets should be fully utilized in the process. To sum up the research, it actually reflects the first shape of the Android that is more strong and safe. Tomorrow's progressions in precision, compliance and privacy shuould guarantee a favourable opposition against the mobile changing threats through androids malware. It must be a continual process as the landscape of mobile technology only continues to evolve. The steps proposed here serve as a starting point for further investigation in order to create more effective mobile security techniques in the future. The future of the mobile

platform security landscape will be tethered on integrating these new generation of security architectures into Android and other mobile platforms in order to protect our data and guarantee user privacy.

One of the most recent malware dataset formation techniques for malware evaluation and grouping has already shown the prospect of future trends and is a promising area for cybersecurity research and development. Some potential future directions in this domain include:Some potential future directions in this domain include:

Enhanced Realism: It is vital that data would be created taking into account the fact that malware is undergoing modifications and being upgraded every day, and thus the datasets need to be made in such a way that they would provide a reflection of the current threat environment. Irrespective of next generation datasets including advanced persistent threats, fileless malware and polymorphic malware and so on can meet future challenges. Besides that, fake traffic along with system logs, user data, and so forth can provide a fully realistic representation of malware as it appears in the real world.

- Cross-Domain and Cross-Platform Analysis: The more cyber threats multi-platform malware enter into the whole network system, the big problem may get worse if we don't look after it very seriously and cover up upcoming risks. Hence, when aggregating data from sources that cover newer types of devices (mobile, IoT), future datasets can become valid also for other platforms and operating systems. Therefore, malware classifiers that are able to work on a number of platforms will be developed and the infection ways and roles of malware will be better comprehended in different environments.
- Behavioral Analysis: IT is not just about the code itself but also the dynamic behavioral aspects of the malware need to be featured in the next datasets. This would encompass the process of monitoring the malicious software with the environment of host system, network or various other system componentsFurthermore, behavior-based logs can effectively drive the generation of cutting-edge detection and classification mechanisms by analyzing the execution features and actions against malware.
- Blockchain Consensus: To ensure the true randomness of decision makers outcomes, the researcher should investigate blockchain consensus algorithms in details.

CONFLICT OF INTEREST STATEMENT

The author declared no conflict of interest for the publication of this work.

REFERENCES

1. O. Hreirati, S. Iqbal, M. Zulkernine. An adaptive dataset for the evaluation of android malware detection techniques. In *Proceedings - 2018 4th International Conference on Software Security and Assurance, ICSSA 2018*; ICSSA, **2018**; pp 62–66.
2. I.A. Doğru, Ö. Kiraz. Web-based android malicious software detection and classification system. *Appl. Sci.* **2018**, 8 (9).
3. S. Iqbal, A. Yasin, T. Naqash. Android (Nougat) Security Issues and Solutions. Proc. IEEE Int. Conf. Appl. Syst. Innov. (ICASI). **2020**.
4. F.O. Catak, A.F. Yazı. A Benchmark API Call Dataset for Windows PE Malware Classification; Cybersecurity Eng. Dept. Istanbul Sehir Univ, Turkey, **2019**.
5. G. D'Angelo, F. Palmieri, A. Robustelli, A. Castiglione. Effective classification of android malware families through dynamic features and neural networks; Taylor & Francis Group, **2021**; Vol. 33.
6. A. Ehsan, C. Catal, A. Mishra. Detecting Malware by Analyzing App Permissions on Android Platform: A Systematic Literature Review; Molde Univ. College, Norway, **2022**; Vol. 22.
7. S.M. Shehata, A.H. El Fiky, M.S. Torky, T.H. Farag, N.A. Abbas. Android Malware Prevention on Permission-Based Analysis. *Int J Appl Eng Res* 15 (1), 5–11.
8. P. Harshalatha, R. Mohanasundaram. Classification of malware detection using machine learning algorithms: A survey. *Int. J. Sci. Technol. Res.* **2020**, 9 (2), 1796–1802.
9. P. Singhal. Malware Detection Module using Machine Learning Algorithms to Assist in Centralized Security in Enterprise Networks. *Int. J. Netw. Secur. Its Appl.* **2012**, 4 (1), 61–67.
10. D. Gibert, C. Mateu, J. Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *J. Netw. Comput. Appl.* **2020**, 153 (102526).
11. C. Du, Y. Tong, X. Chen, et al. Toward Detecting Malware Based on Process-Aware Behaviors; Hindawi Secur Commun Netw, **2023**; Vol. 2023.
12. R. Vinayakumar, M. Alazab, K.P. Soman, P. Poornachandran, S. Venkatraman. Robust Intelligent Malware Detection Using Deep Learning. *IEEE Access* **2019**, 7, 46717–46738.
13. M.S. Akhtar, T. Feng. Evaluation of Machine Learning Algorithms for Malware Detection. *Sensors* **2023**, 23 (2).
14. Shantanu, B. Janet, R. Joshua Arul Kumar. Malicious URL Detection: A Comparative Study. In *Proceedings - International Conference on Artificial Intelligence and Smart Systems, ICAIS 2021*; **2021**; pp 1147–1151.
15. A. Kamboj, P. Kumar, A.K. Bairwa, S. Joshi. Detection of malware in downloaded files using various machine learning models. *Egypt. Informatics J.* **2023**, 24 (1), 81–94.
16. T. Bilot, N. El Madhoun, K. Al Agha, A. Zouaoui. A Survey on Malware Detection with Graph Representation Learning. *ACM Comput. Surv.* **2024**, 56 (11).
17. N.Z. Gorment, A. Selamat, L.K. Cheng, O. Krejcar. Machine Learning Algorithm for Malware Detection: Taxonomy, Current Challenges, and Future Directions. *IEEE Access* **2023**, 11, 141045–141089.
18. S. Raje, S. Vaderia, N. Wilson, R. Panigrahi. Decentralised firewall for malware detection. *Int. Conf. Adv. Comput. Commun. Control 2017, ICAC3 2017* **2017**, 2018-January, 1–5.
19. S. Sheela, S. Shalini, D. Harsha, V.T. Chandrashekar, A. Goyal. Decentralized Malware Attacks Detection using Blockchain. *ITM Web Conf.* **2023**, 53, 03002.
20. A. Nazir, J. He, N. Zhu, et al. Collaborative threat intelligence: Enhancing IoT security through blockchain and machine learning integration. *J. King Saud Univ. - Comput. Inf. Sci.* **2024**, 36 (2).
21. A. Nazir, J. He, N. Zhu, et al. Advancing IoT security: A systematic review of machine learning approaches for the detection of IoT botnets. *J. King Saud Univ. - Comput. Inf. Sci.* **2023**, 35 (10).
22. P.K. Agrawal, H. Gehani, P. Dubey, S. Bura. Natural language based smart garbage management system using Artificial Intelligence. *J. Integr. Sci. Technol.* **2024**, 12 (4), 790.
23. Y.N. Thakare, R. Kadam, U. Wankhade, C. Rawarkar, P.K. Agrawal. Development and design approach of an sEMG-based Eye movement control system for paralyzed individuals. *J. Integr. Sci. Technol.* **2024**, 12 (5), 811.