

Remote Access Trojan Malware Detection Analysis on Android Operating System using Reverse Engineering Method

Dista Nurdiana^{1*}, Vera Suryani¹, Erwid M Jadied¹

¹Faculty of Informatics, Telkom University, Bandung, Indonesia

Received: December 01, 2023

Revised: July 27, 2026

Accepted: August 25, 2026

Published: August 31, 2026

Corresponding Author:

Dista Nurdiana

distanurdiana@student.telkomuniversity.ac.id

DOI: [10.29303/jppipa.v12i8.6334](https://doi.org/10.29303/jppipa.v12i8.6334)

 Open Access

© 2026 The Authors. This article is distributed under a (CC-BY License)



Abstract: This study detected malware on the Android operating system using a system called (syscall). Remote Access Trojan Malware Analysis on the Android operating system using Reverse Engineering Methods helps discover the technological principles of a device, object, or system by analyzing its structure, functions, and operation. This method employed dynamic analysis, namely running every malware on the Android operating system to get information on the system call (syscall) that is running. The results of the system call (syscall) information were selected using the Machine Learning selection feature using the Random Forest method. The purpose of this study is to determine the characteristics based on the system call (syscall) and the detection system's accuracy for the characteristics of Remote Access Trojan malware on the Android operating system. The average accuracy results in this study with four scenarios using the Random Forest method is 96%, and the f1-score value is 94%. This accuracy value is quite good and can be implemented in detecting remote access Trojan malware based on system call (syscall) on the Android Operating System

Keywords: Android Operating System; Malicious Software; Random Forest; Remote Access Trojan; Reverse Engineering

Introduction

Malware or Malicious Software is a software that can infiltrate the operating system so that it can damage the system and also steal important files on the system (Aslan & Samet, 2020; Gibert, Mateu, & Planes, 2020). Malware includes Computer Viruses, Trojan Horses, spyware, dishonest adware, crimeware, and other malicious and unwanted software. (Hemalatha, Roseline, Geetha, Kadry, & Damaševičius, 2021; Kunang & ..., 2022; Roseline, Geetha, Kadry, & Nam, 2020) New malware continues to emerge as technology evolves, both in terms of platform and operating system by exploiting security loopholes and user negligence. This has left more and more parties harmed. Malware can infect all types of operating systems (Alimardani & Nazeh, 2018; Aprilliansyah, Riadi, & Sunardi, 2022; Liu et al., 2019; M. Aranitasi, A. Daci, 2022).

The android system is an operating system based on linux for mobile phones, such as smartphones and tablet PCs (Adekotujo, Odumabo, Adedokun, & Aiyeniko, 2020; Li, Zhou, Yuan, Li, & Leung, 2020; Liu et al., 2019; Uttarwar, Tidke, Dandwate, & Tupe, 2021). The Android system has some advantages, such as an open source operating system, multitasking, and the ease of notifications to the number of applications or software that can be detected using the Android system (Kunang & ..., 2022). The evolution of Android devices and apps has rapidly changed us in our daily lives. Web browsing, social media, and internet banking are the examples of Android services using an internet connection and multiple access permissions (Castillo-Zúñiga et al., 2020). Therefore, Android devices have played an important role and become an important part of human life. Therefore, there are several negative impacts of using Android devices, one of which is cyber crime. Its crime is committed by using a computer or other means

How to Cite:

Nahta, D., Suryani, V., & Jadied, E. M. (2026). Remote Access Trojan Malware Detection Analysis on Android Operating System using Reverse Engineering Method. *Jurnal Penelitian Pendidikan IPA*, 12(8), 243–251. <https://doi.org/10.29303/jppipa.v12i8.6334>

of communication to inflict fear and anxiety on people or damage, harm, and destroy property. Cybercrime has two categories: computer-aided and computer-focused cybercrime. Some examples of computer-aided cybercrimes are child pornography, fraud, money laundering, and cyberbullying. Meanwhile, the examples of computer-focused cybercrime are hacking, phishing, and website destruction (Dennis, 2019; Donalds, Barclay, & Osei-Bryson, 2022; Joseph, 2017; Techopedia, 2018).

The topics and limitations of the problems used in the final project research included how to implement the Reverse Engineering methods to obtain information (System Call) contained in the Remote Access Trojan malware. The operating system used only the Android operating system with a dataset in the form of 40 types of Android Remote Access Trojan malware and 40 Android applications. Summary of related research can be seen in Table 1.

Table 1. Summary of Related Research

Heading	Author / Year	Research Results	Excess	Deficiency
The Use of Machine Learning Techniques to Advance the Detection and Classification of Unknown Malware	Shhadat, Bataineh, Hayajneh, & Al-Sharif (2020)	Analyzed many machine learning algorithms in malware detection and classification. Implemented various models for malware detection and classification using the implementation provided by sklearn, namely: K-nearest Neighbors (KNN), Support Vector Machine (SVM), Bernoulli Naive Bayes (Bernoulli NB), J48 Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), and Hard Voting (HV). On certain classification algorithms used LG, SVM, Bernoulli NB and DT.	For very high accuracy results in all binaries and multi-classifiers.	The datasets used is less vary with unbalanced comparisons.
Android Malware Detection using Feature Selections and Random Forest	Eom, Kim, An, Park, & Kim (2018)	Used the Random Forest method to detect Android malware. The accuracy results obtained a high value. To demonstrate an approach to detection accuracy. This study calculated OOB (Out-of-Bag) errors using detection metrics, including false positives and false negatives.	In the experiments carried out, the results of accuracy were above average.	Uneven distribution of features to conduct research trials.
Malware Detection in Android-based on Dynamic Analysis	Bhatia & Kaushal(2017)	Dynamic analysis of malware used system call (syscall) analysis that could be used efficiently to classify unknown applications as malicious or not. This research serves to detect and monitor the behavior of malicious applications that use complex incognito techniques.	The dataset used in this study was balanced in number and had high accuracy results.	How to get system call (syscall) data is done one by one manually.
Mobile Malware Attacks: Review, Taxonomy & Future Directions	Qamar, Karim, & Chang (2019)	A review of smartphone malware attacks, vulnerabilities, detection techniques, and security solutions on Android.	There are guidelines to reduce or even avoid the harmful effects of intruders and hackers .	The data had a vulnerable time that is too past (2013-2019).
A Taxonomy on Recent Mobile Malware: Features, Analysis Methods, and Detection Techniques	Alimardani & Nazeh, (2018)	A review of mobile malware (features, methods of analysis and detection techniques).	Learn related data sets (features, analysis methods, and detection techniques) that are acceptable and reserved for researchers.	Dataset retrieval was only based on common parameters (trusted and often used).
Malware Detection: A Framework for Reverse Engineered Android Applications through Machine	Urooj, Shah, Maple, Abbasi, & Riasat (2022)	Using Android reverse engineering application features and machine learning algorithms to find vulnerabilities that exist in Android applications. Analysis used to detect malicious Android applications using static analysis .	Using a model that combines a more innovative static feature set with a malware sample dataset.	It is only limited to static analysis and has slow processing techniques that cause this

Heading	Author / Year	Research Results	Excess	Deficiency
Learning Algorithms			Using ensemble learning with Machine Learning algorithms namely, AdaBoost, Support Vector Machine (SVM), etc.	research to produce less accurate results.
Analysis, Detection, and Classification of Android Malware through System Calls	Shakya & Dave (2022)	The results of K-Nearest Neighbor and Decision Tree modeling provide the highest accuracy in malware detection and Family Classification, respectively.	The process for analyzing was carried out by static analysis methods.	The malware datasets used did not vary and did not have a numerous number or limited.
Mobile Malware Detection Techniques using System Calls	M. Aranitasi, A. Daci, (2022)	Recognized the abnormal behavior of the Android operating system due to malware attacks based on system call identification (syscall).	Using mathematical methods to perform system analysis call (syscall).	There was no validation between the method and the consumed dataset.
Malware Analysis and Detection using Reverse Engineering	Rashmitha, Alwina, Angelin, & Ramesh, (2022)	The reverse engineering used successfully found system vulnerabilities, researched malware, and analyzed the complexity of software algorithms serves to protect data hacking.	The characteristics of the resulting malware are quite complete and informative.	The analysis only used static analysis and time-consuming research which was quite a long time.
Detection and Robustness Evaluation of Android Malware Classifiers	Anupama et al.(2022)	Used Support Vector Machine (SVM) models for static analysis features and Random Forest models for dynamic analysis features. Models used for dynamic analysis features are vulnerable to attacks, but they showed higher resilience than classifiers created based on static analysis features.	The analysis features used were more than one, there were dynamic analysis and hybrid analysis	The malware datasets did not vary and did not have large or limited numbers.

In this study, the researcher conducted an analysis of malware detection, particularly Remote Access Trojan malware on the Android operating system. The dataset used by the researcher is 40 types of Android Remote Access Trojans malware and 40 Android applications. The researcher used the Reverse Engineering method with dynamic testing. This method was used because it could work very detailed and was used to extract information data contained in malware. In addition, the use of the Reverse Engineering method is also useful for discovering the technological principles of a device, object, or system from the analysis of its structure, function, and operation that is commonly used for maintenance or creating new programs equally without copying anything from the original. The information data used is a type of System Call (Syscall) in every malware and application used as a dataset.

Method

Objective

The purpose of this research is to implement the Reverse Engineering Method used to analyze Remote

Access Trojan malware on the Android operating system. In addition, this final project research also aims to find out the characteristics based on the system call (syscall) and the accuracy of the detection system for the characteristics of Remote Access Trojan malware on the Android operating system.

Built System

The following is a system design flow from the analysis and detection of Remote Access Trojan Malware (Setia, Aldya, & Widiyasono, 2019; Sharma, 2019). The system built after having a dataset is to start running each dataset into the Virtual Device Manager on Android Studio. If it is successful, then continued by analyzing each dataset dynamically by running System Call (syscall). After successfully in each dynamic analysis process, the results of the syscall that have been obtained are selected using Machine Learning with the Random Forest method and the selection results will be analyzed. So, they will get the analysis results contained in this research report.

Result and Discussion

Dataset

The dataset used is 40 types of Android malware Remote Access Trojan and 40 Android applications. The malware dataset in this study was downloaded on [the https://bazaar.abuse.ch/browse/](https://bazaar.abuse.ch/browse/) web page. For the Android application dataset in this study was downloaded on the https://apkpure.com/web_page. The list of datasets in this study can be seen in <https://bit.ly/3ZVFMuE> while the datasets used for the process of collection features using Machine Learning. It can be seen in <https://bit.ly/3wik0nf>.

Dataset Sharing

In this study, the distribution of the number of datasets used in each scenario was different. The comparison of the datasets number used in each scenario is 1:1, 1:2, 4:1, and 2:1. The total of the entire dataset is 80 data with the system call (syscall) feature which was found to be 54 *syscall*.

Retrieval of System Call (Syscall) Feature on Dataset

In the existing dataset and the installation process has been carried out on the Android operating system. Each dataset will be processed to find out the system call (syscall) feature. In this process, one by one process was carried out to find out the system call (syscall) that is running on each dataset. It can be seen at the flow of retrieving the system call (syscall) feature for each dataset in Figure 3. How to find out the system call (syscall) based on the PID number on each dataset that has been run ("Strace - Trace System Calls and Signals," 2021). The results of the system call (syscall) will be combined and recorded into one unified file with excel format and then will be used as a dataset for selecting important system call (syscall) features using Machine Learning Feature Selection with the Random Forest method.

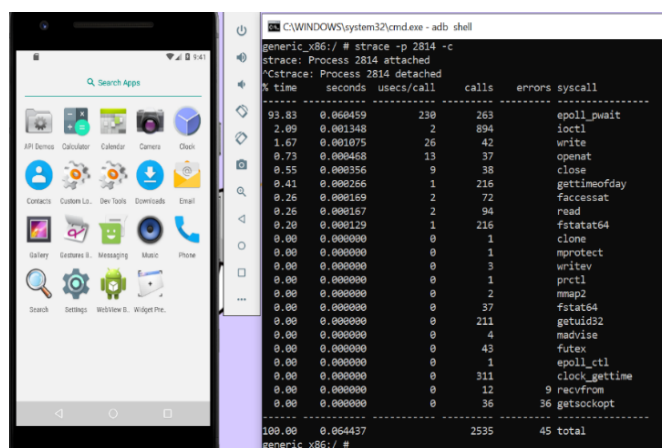


Figure 2. Strace Output Results from Malware or Applications (Datasets) Running

openat	read	ioctl	mmap2	prctl
mmap2	madvise	futex	fstatat64	epoll_pwait
getuid32	setrlimit	setpriority	clone	writew
ugetrlimit	gettid	mknodat	renameat	faccessat
sched_yield	ftruncate64	fdatasync	getuid32	getpriority
close	write	sigaction	mprotect	pread64
fstat64	getdents64	clock_gettime	readlinkat	recvfrom
dup	gettimeofday	fsync	_llseek	rt_sigreturn
fcntl64	epoll_ctl	unlinkat	fchmodat	getsockopt
flock	getpid	pwrite64	tgkill	connect
sigaltstack	sendto	rt_sigprocmask	lseek	

Figure 3. System Call (syscall) Names Found in Each Dataset

Feature Selection using Random Forest

At this stage, a feature selection was carried out on the results of system call analysis (syscall) for each dataset that has been run in the Virtual Device Manager in Android Studio. The selection feature itself used the help of Machine Learning to speed up the process, as for the method used, namely Random Forest. The result of this feature selection is a feature score. This algorithm can be used to rank the importance of the system call (syscall) in each dataset. This feature selection used system call (syscall) results from 80 datasets (40 types of Android malware Remote Access Trojans and 40 Android applications). Features that produced large values for this score were classified as more important than features that produced small values. Based on this score, the researcher selected the most important features and dropped the least important ones for model creation. The source code used was based on the tutorial (Banerjee, 2019; Scikit-learn, 2022).

In model Random Forest Classifier is this stage, a Random Forest Classifier modeling was built with default parameters ($n_estimator = 10$). This can also be called as the decision-tree or modeling of the method used. In this study, 10 decision-trees was used to build the model. Evaluation Metrics is a brief explanation of the metrics used to obtain classification results [16] is True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN) and Confusion Matrix.

True Positiver (TP) is defined as the number of events that are considered positive and actually positive. True Negative (TN) is defined as the number of events that are considered negative and actually negative. False Positive (FP) is defined as the number of events that are considered positive but actually negative. False Negative (FN) is defined as the number of events that are considered negative but are actually positive. Confusion Matrix is defined as a two-dimensional square matrix with rows and columns for each class. Each matrix element indicates the number of instances where the actual class is a row and the predicted class is a column. The illustration of the confusion matrix can be seen in Figure 4.

		Predicted class		
		yes	no	Total
Actual class	yes	TP	FN	P
	no	FP	TN	N
Total		P'	N'	P + N

Figure 4. Confusion Matrix Illustration

Accuracy is defined as the sum of the sum of true positives and true negatives divided by the total number of events. The formula can be seen in Formula 1.

$$Accuracy = \frac{TP + FP}{TP + FP + TN + FN} \quad (1)$$

Precision is the ratio of true positive events to the total number of positive events that may include events which are considered falsely positive. The formula can be seen in Formula 2.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall is the ratio of the number of true positive examples to the total number of actually positive instances. The formula can be seen in Formula 3.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Parameters Used

The following is a brief explanation of the parameters used to obtain classification results [24] is *n_estimator* can also be referred as the number of trees in Random Forest modeling. It has a default value of =100, but in some versions, it can also have a value between 10 and 100. *Random_state* is used to control the randomness of the sample bootstrap used when building the tree (if *bootstrap=True*) and feature sampling to consider when looking for the best split on each node (if *max_features < n_features*). *Test_size*, this

parameter is used to define the size of the testing data (what percentage of the overall data). In this study, in order to maximize accuracy results and find out in detail related to the characteristics of malware on the Android operating system, several scenarios are required. This study builds 4 (four) scenarios on different comparisons between malware datasets and applications. But, it has the same parameter numbers for *test_size*, *random_state*, and Random Forest (*n_estimator*) modeling parameters. The scenario can be seen in Table 2.

Table 2. Composition Division of Sum & Comparison Datasets in Each Scenario.

Scenario	Composition of the Number of Datasets	Dataset Comparison
1	40 Malware 40 Applications	1:1
2	40 Malware 20 Applications	1:2
3	10 Malware 40 Applications	4:1
4	20 Malware 40 Applications	2:1

Evaluation

In the research that was built, there were several test scenarios that explained in point 3.4. The test scenario aims to determine the accuracy, precision, recall, f1-score, confusion matrix, and feature score of the selection feature using the Random Forest machine learning method. From the results of this test scenario, it makes easier to find out the characteristics of malware based on system calls on the Android operating system.

Test Results

The test results in this study were different in each scenario. The average accuracy result in the four scenarios that have been created is 96% with a confusion matrix balanced between predictions and authenticity in the results of the dataset being tested. The accuracy results of the selection of system call features using the Random Forest method is presented in Table 3.

Table 3. Results of a Curated Selection of Syscall Features using Random Forest

Scenario	Precision	Recall	F1-Score	Accuracy	Confusion Matrix
1 Malware	93%	100%	96%	96%	[[13 0]
Application	100%	90%	95%		[1 10]]
2 Malware	100%	91%	95%	94%	[[10 1]
Application	88%	100%	93%		[0 7]]
3 Malware	100%	67%	80%	93%	[[2 1]
Application	92%	100%	96%		[0 12]]
4 Malware	100%	100%	100%	100%	[[8 0]
Application	100%	100%	100%		[0 10]]

Based on the results of the feature selection, it means that the results of the modeling are successful enough to prove that system call (syscall) is able to

classify the characteristics of malware and applications on the Android operating system. The results of this system call accuracy can classify the most important

features of the system call (syscall) based on the feature score obtained. The list of important system call (syscall) features that are often used in every malware and application based on the feature score can be seen in Table 4.

Table 4. The Most Important List System Call based on Feature Score

Scenario	System Call (syscall)	Feature Score
1	1 renameat	0.178393
	2 unlinkat	0.140947
	3 getdents64	0.124363
	4 rt_sigprocmask	0.115806
	5 fstatat64	0.063041
2	1 getdents64	0.153235
	2 _llseek	0.137450
	3 Gettid	0.133223
	4 readlinkat	0.111314
	5 fsync	0.110866
3	1 unlinkat	0.248327
	2 rt_sigprocmask	0.163544
	3 GetSockopt	0.124211
	4 setpriority	0.086638
	5 getdents64	0.081909
4	1 rt_sigprocmask	0.165577
	2 getdents64	0.145609
	3 mprotect	0.103270
	4 GetSockopt	0.101758
	5 unlinkat	0.085557

List System Call Terpenting

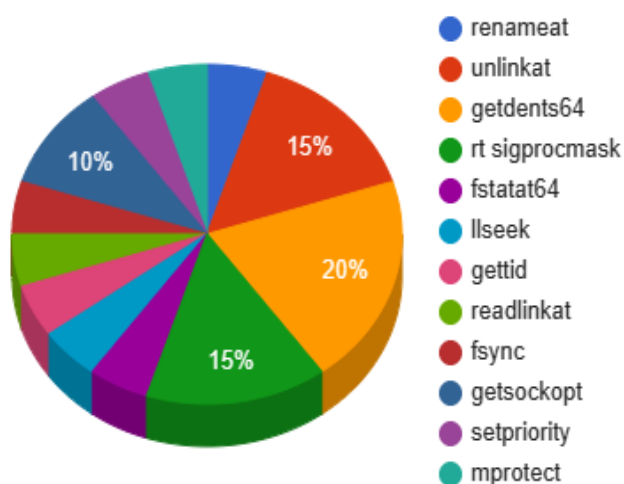


Figure 5. Lists of Most Important System Call based on Feature Score

Based on the data in Figure 8, there are four System Call (Syscall) that are most important and often used in malware and tested applications (datasets). These four system calls (syscall) are *getdents64*, *unlinkat*, *rt_sigprocmask*, and *getsockopt*.

Analysis of Test Results

Android is a Linux-based operating system designed for mobile devices specifically for touch screen devices, such as smartphones and tablets. Started from the phones that can still be accessed in black and white display up to the latest smartphones or minicomputers. The operating system has changed a lot in the last 15 years. One of the most widely used mobile operating systems today is Android. It is a software founded in Palo Alto, California in 2003. The version history of the Android OS was started with the release of the Android 1.0 beta in November 2007. Since April 2009, each Android version has been developed codenamed in favor of dessert items. These versions were released in alphabetical order: Alpha, Beta, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo, and Pie. Since Android 10 Google does not use fruit names for naming Android versions. The latest Android version is Android 12 which launched in May 2021 with SDK Build-Tools version 31 (Aprilliansyah et al., 2022).

The android application also contains a Manifest file where this file is the most critical file that includes all the information needed to run the application, such as metadata, components application, and permissions. The metadata contains some standard application information, such as package name, application ID, SDK version, package version, etc. Permissions are the important information in the manifest file where all the resources used by the application should be declared here. These permissions can be anything from APIs, resources to permissions for hardware access. Access these permissions are used when installing an app or when using those resources (Shakya & Dave, 2022). The data Can be seen in Figure 1. which is a Taxonomy of the android manifest (Urooj et al., 2022).

Android has a wide reach for its users and great data. Existing data is easily hacked by irresponsible hackers and abuses the use of Android malware (Mat, Razak, Kahar, Arif, & Firdaus, 2022; Pan, Ge, Fang, & Fan, 2020; Qiu et al., 2021). It is a malicious software category created specifically to attack the Android system. The type of malware that can attack the Android system will usually take advantage of the operating system where the main target is the Personal Digital Assistant (PDA). This causes the users to easily experience leakage of data and confidential information (Isohara, Takemori, & Kubota, 2011). Android malware is divided into several categories based on the characterization of the installation, activation, and way of operating its own malware (Y. Zhou & Jiang, 2012).

Analysis on Android Malware is divided into two, namely dynamic analysis and static analysis. Dynamic analysis is the process of analyzing program code that includes calculations and testing applications

by implementing direct secfigs. This analysis aims to find errors or defects in the program during operation (finding malware). The advantage of this dynamic analysis is easy to analyze the behaviors of the application during run-time and analyze important data or target data. At the time of conducting dynamic analysis, it requires a lot of resources and a fairly long time. Meanwhile, static analysis is the process of analyzing program code by not running the program first. This analysis is used to facilitate the observation of the entire program before the running of the program during run-time (Christodorescu & Jha, 2003). The advantages of this static analysis only require less time and a small amount of resources.

Reverse Engineering is the process of discovering the technological principles of an object or component through the analysis of its structure and function (Asher et al., 2021; Zhang, 2003). The results of the analysis can be used to redesign the same object to produce a number of small components that will be adjusted to the user's need. The way the reverse engineering method works, it has benefits in before method running, especially in terms of flexibility and speed in analyzing the structure and function of an object (Z. Zhou, Chen, & Xie, 2007).

Based on the test scenarios and results, the analysis was obtained from the four most important system calls (syscall) and is often used in malware and applications tested (datasets). The analysis obtained is the *getsockopt*, *getdents64*, *unlinkat*, and *rt_sigprocmask*. *Getsockopt* is System Call (syscall) that must be watched. The system called (syscall) is one of the characteristics of malware on the Android operating system. This is because the *getsockopt* system call (syscall) functions to manipulate, set, and get options related to the Android operating system socket. In addition, system calls (syscall) of *getdents64*, *unlinkat*, and *rt_sigprocmask* are not dangerous and are not characteristic of malware on the Android operating system. This is because the function of the system call (syscall) does not harm or even try to hack the Android operating system. Therefore, the characteristics of the *Remote Access Trojan malware* on the Android operating system are the presence of a *getsockopt system call (syscall)*.

Conclusion

Based on the results of tests and analyses that have been carried out, reverse engineering methods have been implemented to detect Remote Access Trojan malware with dynamic analysis. Dataset collection carried out manually and tested one by one is a unique feature of dynamic analysis in this study. Feature selection using machine learning of the Random Forest method obtained an average accuracy of 96%, precision of 97%, recall of 93.5%, and f1-score of 94%. Thus,

machine learning selection features are important because they produce the most important list system call (syscall) and are often used in malware and tested applications (datasets). The list of the *getsockopt* system call (syscall) characterizes the existence of Remote Access Trojan malware on the Android operating system. Suggestions for further research are to develop how to find out the system call (syscall) automatically, as well as perform many test scenarios, and optimize the accuracy results of the random forest method to produce better accuracy and performance to make it easier when analyzing the existence of malware on the Android operating system.

Acknowledgments

The authors would like to thanks to University for give occasion for this research

Author Contributions

Investigation, D.N, V.S and E.M.J; formal analysis, D.N, V.S and E.M.J; investigation, D.N, V.S and E.M.J; resources D.N, V.S and E.M.J; data curation, D.N, V.S and E.M.J ; writing—original draft preparation, D.N, V.S and E.M.J ; writing—review and editing, D.N, V.S and E.M.J; visualization D.N, V.S and E.M.J; supervision, D.N, V.S and E.M.J; project administration, D.N, V.S and E.M.J; funding acquisition, D.N, V.S and E.M.J. All authors have read and agreed to the published version of the manuscript.

Funding

This research is fully supported by the author's funds without any external funding sources

Conflicts of Interest

We certify that there is no conflict of interest with any financial, personal and other relationships with other peoples or organization related to the material discussed in the manuscript.

References

- Adekotujo, A., Odumabo, A., Adedokun, A., & Aiyeniko, O. (2020). A Comparative Study of Operating Systems: Case of Windows, UNIX, Linux, Mac, Android and iOS. *International Journal of Computer Applications*, 176(39). <https://doi.org/10.5120/ijca2020920494>
- Alimardani, H., & Nazeh, M. (2018). A taxonomy on recent mobile malware: Features, analysis methods, and detection techniques. *ACM International Conference Proceeding Series*, 44–49. <https://doi.org/10.1145/3230467.3230478>
- Anupama, M. L., Vinod, P., Visaggio, C. A., Arya, M. A., Philomina, J., Raphael, R., ... Mathiyalagan, P. (2022). Detection and robustness evaluation of android malware classifiers. *Journal of Computer Virology and Hacking Techniques*, 18(3), 147–170. <https://doi.org/10.1007/s11416-021-00390-2>
- Aprilliansyah, D., Riadi, I., & Sunardi. (2022). Analysis

- of Remote Access Trojan Attack using Android Debug Bridge. *IJID (International Journal on Informatics for Development)*, 10(2), 102–111. <https://doi.org/10.14421/ijid.2021.2839>
- Asher, S. W., Jan, S., Tsaramirsis, G., Khan, F. Q., Khalil, A., & Obaidullah, M. (2021). Reverse engineering of mobile banking applications. *Computer Systems Science and Engineering*, 38(3). <https://doi.org/10.32604/CSSE.2021.016787>
- Aslan, O., & Samet, R. (2020). A Comprehensive Review on Malware Detection Approaches. *IEEE Access*, Vol. 8. <https://doi.org/10.1109/ACCESS.2019.2963724>
- Banerjee, P. (2019). *Random Forest Classifier Tutorial | Kaggle*. Retrieved from <https://www.kaggle.com/prashant111/random-forest-classifier-tutorial>
- Bhatia, T., & Kaushal, R. (2017). Malware detection in android based on dynamic analysis. *2017 International Conference on Cyber Security And Protection Of Digital Services, Cyber Security 2017*. <https://doi.org/10.1109/CyberSecPODS.2017.8074847>
- Castillo-Zúñiga, I., Luna-Rosas, F. J., Rodríguez-Martínez, L. C., Muñoz-Arteaga, J., López-Veyna, J. I., & Rodríguez-Díaz, M. A. (2020). Internet data analysis methodology for cyberterrorism vocabulary detection, combining techniques of big data analytics, NLP and semantic web. *International Journal on Semantic Web and Information Systems*, 16(1), 69–86. <https://doi.org/10.4018/IJSWIS.2020010104>
- Christodorescu, M., & Jha, S. (2003). Static analysis of executables to detect malicious patterns. *Proceedings of the 12th USENIX Security Symposium*, 169–186.
- Dennis, M. A. (2019). cybercrime | Definition, Statistics, & Examples | Britannica.
- Donalds, C., Barclay, C., & Osei-Bryson, K.-M. (2022). Towards a Cybercrime Classification Ontology. In *Cybercrime and Cybersecurity in the Global South*. <https://doi.org/10.1201/9781003028710-15>
- Eom, T., Kim, H., An, S. M., Park, J. S., & Kim, D. S. (2018). Android malware detection using feature selections and random forest. *Proceedings - 2018 4th International Conference on Software Security and Assurance, ICSSA 2018*, 55–61. <https://doi.org/10.1109/ICSSA45270.2018.00023>
- Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*, Vol. 153. <https://doi.org/10.1016/j.jnca.2019.102526>
- Hemalatha, J., Roseline, S. A., Geetha, S., Kadry, S., & Damaševičius, R. (2021). An efficient densenet-based deep learning model for Malware detection. *Entropy*, 23(3). <https://doi.org/10.3390/e23030344>
- Isohara, T., Takemori, K., & Kubota, A. (2011). Kernel-based behavior analysis for android malware detection. *Proceedings - 2011 7th International Conference on Computational Intelligence and Security, CIS 2011*, 1011–1015. <https://doi.org/10.1109/CIS.2011.226>
- Joseph, A. E. (2017). Cybercrime definition. *Computer Crime Research Center*, (June 2017).
- Kunang, Y. N. K. Y. N., & ... (2022). Analisis Forensik Malware Pada Platform Android. *Analisis Forensik ...*. Retrieved from http://eprints.binadarma.ac.id/10591/%0Ahttp://eprints.binadarma.ac.id/10591/1/yesinovaria_kunang_analisis_forensik_malware_android_ubd.2.pdf
- Li, H., Zhou, S., Yuan, W., Li, J., & Leung, H. (2020). Adversarial-Example Attacks Toward Android Malware Detection System. *IEEE Systems Journal*, 14(1). <https://doi.org/10.1109/JSYST.2019.2906120>
- Liu, X., Du, X., Zhang, X., Zhu, Q., Wang, H., & Guizani, M. (2019). Adversarial samples on android malware detection systems for IoT systems. *Sensors (Switzerland)*, 19(4). <https://doi.org/10.3390/s19040974>
- M. Aranitasi, A. Daci, and A. G. (2022). Mobile malware detection techniques using system calls. *International Journal of Engineering Research and Applications Wwww.Ijera.Com*, 12, 54–57.
- Mat, S. R. T., Razak, M. F. A., Kahar, M. N. M., Arif, J. M., & Firdaus, A. (2022). A Bayesian probability model for Android malware detection. *ICT Express*, 8(3). <https://doi.org/10.1016/j.icte.2021.09.003>
- Pan, Y., Ge, X., Fang, C., & Fan, Y. (2020). A Systematic Literature Review of Android Malware Detection Using Static Analysis. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.3002842>
- Qamar, A., Karim, A., & Chang, V. (2019). Mobile malware attacks: Review, taxonomy & future directions. *Future Generation Computer Systems*, 97, 887–909. <https://doi.org/10.1016/j.future.2019.03.007>
- Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S., & Xiang, Y. (2021). A Survey of Android Malware Detection with Deep Neural Models. *ACM Computing Surveys*, Vol. 53. <https://doi.org/10.1145/3417978>
- Rashmitha, B., Alwina, J., Angelin, B., & Ramesh, E. R. (n.d.). Malware analysis and detection using reverse Engineering. *International Journal of Computer Science and Information Technology Research*, 10(4). Retrieved from www.researchpublish.com,
- Roseline, S. A., Geetha, S., Kadry, S., & Nam, Y. (2020). Intelligent Vision-Based Malware Detection and

- Classification Using Deep Random Forest Paradigm. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.3036491>
- Scikit-learn. (2022). sklearn ensemble Random Forest Classifier. *Scikit-Learn*. Retrieved from 11/13/22, 9:13 AM <https://sklearn.ensemble.RandomForestClassifier> – scikit-learn 1.1.3 documentation <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier>
- Setia, T. P., Aldya, A. P., & Widiyasono, N. (2019). Reverse Engineering untuk Analisis Malware Remote Access Trojan. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, 5(1). <https://doi.org/10.26418/jp.v5i1.28214>
- Shakya, S., & Dave, M. (2022). *Analysis, Detection, and Classification of Android Malware using System Calls*. Retrieved from <https://arxiv.org/abs/2208.06130v1> <https://arxiv.org/ftp/arxiv/papers/2208/2208.06130.pdf>
- Sharma, M. (2019). A Study on RAT (Remote Access Trojan). *Academic Journal of Forensic Sciences*, 02(02).
- Shhadat, I., Bataineh, B., Hayajneh, A., & Al-Sharif, Z. A. (2020). The Use of Machine Learning Techniques to Advance the Detection and Classification of Unknown Malware. *Procedia Computer Science*, 170, 917–922. <https://doi.org/10.1016/j.procs.2020.03.110>
- Strace - Trace System Calls and Signals. (2021). Retrieved January 25, 2023, from <http://linux.die.net/man/1/strace>
- Techopedia. (2018). What is Cybercrime? - Definition from Techopedia.
- Urooj, B., Shah, M. A., Maple, C., Abbasi, M. K., & Riasat, S. (2022). Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms. *IEEE Access*, 10, 89031–89050. <https://doi.org/10.1109/ACCESS.2022.3149053>
- Uttarwar, P. S., Tidke, R. P., Dandwate, D. S., & Tupe, U. J. (2021). A Literature Review on Android-A Mobile Operating system. *International Research Journal of Engineering and Technology*, (September).
- Zhang, Y. (2003). Research into the engineering application of reverse engineering technology. *Journal of Materials Processing Technology*, 139(1-3 SPEC). [https://doi.org/10.1016/S0924-0136\(03\)00513-2](https://doi.org/10.1016/S0924-0136(03)00513-2)
- Zhou, Y., & Jiang, X. (2012). Dissecting Android malware: Characterization and evolution. *Proceedings - IEEE Symposium on Security and Privacy*, 95–109. <https://doi.org/10.1109/SP.2012.16>
- Zhou, Z., Chen, D., & Xie, S. (Shengquan). (2007). Springer Series in Advanced Manufacturing. *Thermoplastics and Thermoplastic Composites*, 863–866. Retrieved from <http://www.springer.com/series/7113> <http://linkinghub.elsevier.com/retrieve/pii/B9781856174787500118>