Detecting Illicit Data Leaks on Android Smartphones Using an Artificial Intelligence Model

Serge Lionel Nikiema, Aminata Sabané, Abdoul-Kader Kabore, Rodrique Kafando, Tégawendé F. Bissyandé

Detecting Illicit Data Leaks on Android Smartphones Using an Artificial Intelligence Model

Serge Lionel Nikiema[1], **Aminata Sabané**[2], **Abdoul-Kader Kabore**[1], **Rodrique Kafando**[1], **Tégawendé F. Bissyandé**[1 2]

[1]Centre d'Excellence CITADEL, Université Virtuelle du Burkina Faso
[2]Département d'Informatique, UFR/SEA, Université Joseph KI-ZERBO, Burkina Faso

*E-mail : lionelsergepha@gmail.com.com

Abstract

In today's world, hackers and espionage agents have become extremely interested at android – the most common mobile operating system in whole planet. We introduce DeepDetector – a system based on artificial intelligence to recognize data thefts in Android. This model is based upon a large dataset comprising of clean and tainted network traffic trained using a Random Forest Classifier. DeepDetector scores high in two main areas as it achieves 82.9% accuracy for connection anomaly detection and 89.9% recall in connection anomaly detection whereas it gets 78.9 percent accuracy and 81.6 recall in terms of detection of under the system mounted with Raspberry Pi, automatic data collection, preparing of a dataset, training and testing of the model, as well as leak detection are ensured. In this regard, DeepDetector offers a viable way of enhancing Android user security.

Keywords

Android, mobile security, data leak, machine learning, anomaly detection

I INTRODUCTION

Today, there are 72% of the world's mobile operating system market share on Android smartphones [1]. At the same time, this high degree of adoption renders Android as a potential prey for malware seeking to steal users' information. This has resulted from issues that still need to be solved, for example, despite propositions of Android malware detection techniques, only direct detection of data leaks still remains major challenge. This is due to the fact that it is not easy to detect network traffic on these devices. This implies that it becomes more difficult for us to secure our phones compared to the computers. In addition, even if there were some external network analyzers, it is not easy, neither quick, AI-based detector. This study has, therefore, been informed by this consideration. In addition, there are three main detection methods: these are static, dynamic and hybrid analyses.

Yet, sophisticated static detection approaches also fail against modern malware. Dynamic techniques based on machine learning are however the most advanced approaches being considered as an emerging alternative.

Multiple recent study involving different machine learning methodologies has shown promising result. By utilizing an inter-app information flow analysis, Lee et al. (2) were able to attain

95% on the accuracy rating with the help of RNNs. Song et al. (2015) utilized memory image features to develop and build a random forests model with 97% accuracy (3). One study in particular, by Zhang et. al, [4], built a communication monitoring platform using CNN's. With such a detection rate of 98%. Lastly, Zheng et al., [5] proposed hybrid framework which merges static, dynamic analysis along with an SVM that offers 93% accuracy rate. Actually, lots of studies have focused on malware and mobile phones.

The following describes DeepDetector that is a system for detecting illegal information leakage on Android phones. Instead, the proposed solution involves a machine learning algorithm that will be able to analyze in real time network data about other devices and detect any abnormalities suggesting unauthorized leakage of personal or confidential data. Although there are many studies already available pertaining to malware detection for smartphones; deep detector represents the first completely self-sufficient ai system. This system incorporates automated processes. The entire process of harvesting of network traffic, extraction of pertinent features and their processing using the pre-trained ML model as well as the visualization of results occur independently within the system. DeepDetector is an advancement because it offers the first solution for deep learning-based leakage detection.

It will be organized according to following arrangement. Then, describing the overall and specific structure of the system will be included in the next section. To begin with, the design of the dataset and training of this model with its rationalization of why the picked algorithm is appropriate will also be stated. Thirdly, a comparative study will be carried out and resultant findings highlighted. Thereafter, a conclusion with relevant recommendations/perspective shall be made.

II MATERIALS AND METHODS

2.1 Hardware Architecture

The DeepDetector system was implemented on a nano computer called Raspberry PI 3 Model B with a powerful internal hardware configuration that includes 1.4 GHz 64 bit quad core ARM Cortex-A53 processor, 1 GB RAM, 802. Raspian is the operating system that powers this device. It has a 10.1-inch touch screen attached for interacting with the users.

2.2 DeepDetector General Software Architecture

The approach used by DeepDetector follows a 5-step main architecture as shown in figure 1:

1. Network traffic collection: Tshark detects Android's network traffic in real-time. The data sets include TCP connection logs and DNS queries. This data is in the native/.pcap format that cannot be used directly in its current form).
2. Data pre-processing: Unprocessed data is converted into the Zeek log format and then undergoes change to get the training dataset. Zeek program organises data, which is contained in more than two logs that are specifically optimised for a better dataset preparation, such as conn.log and dns.log. Thereafter, they organize and prepare the connection and DNS query datasets then clean them, select features, vectorize and normalize.
3. Machine Learning model training and evaluation: Random Forest Classifier model is trained supervisely on developed data set.
4. Anomaly detection: In production, trained model analyzes Android network traffic for possible data leaks anomalies.

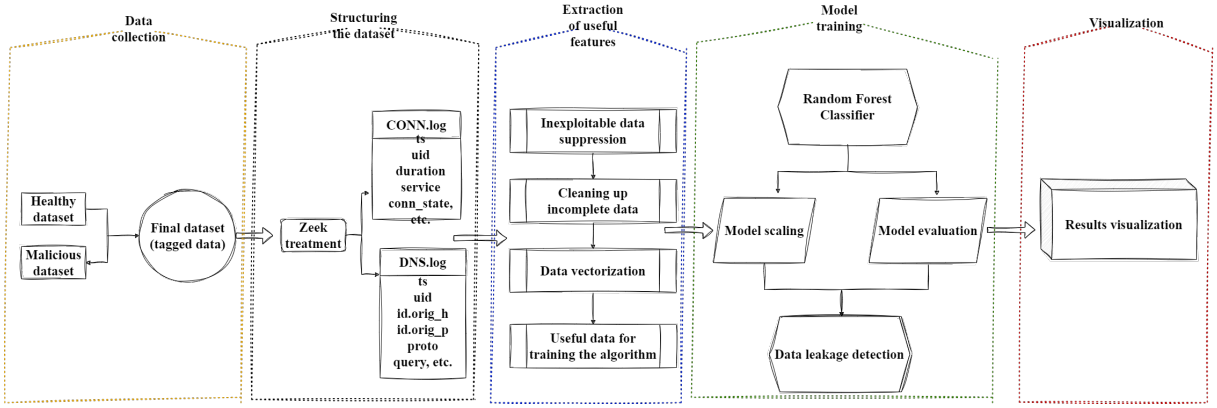

Figure 1: System architecture.

5. Visualization of results: in this step, you will observe the detection outcomes and draw conclusions from them.

2.3 Dataset

DeepDetector's training dataset consists of network captures from:

- Healthy Android devices: Smartphone usage leads to normal traffic as well. To derive this "healthy" dataset, we had to go by exclusion involving using, emulating or virtualing a smartphone. Therefore, the most suitable method was the virtualization that offered minimal probability of a corrupted data.
- Devices infected with malware: traffic from one out of eight common Android RATs. A particular type of malware known as RAT (remote access Trojan) allows for the attacker to assume command and control the victim's device from afar. This takeover is primarily done for stealing of data. The Stratosphere Laboratory of the Czech Technical University, in Prague, created the Android Mischierf Dataset [6].

The Android Mischief is a collection that comprises of network captures from different malware infected android apps. The current version of the dataset includes 8 packet captures from 8 Android RATs executed: RAT01 - ANDROID TESTER V.6.4.6, RAT02 - DROIDJACK V4.4 , RAT03 - HAWKSHAW, RAT04 - SPYMAX V2.0, RAT05 - ANDRORAT, RAT06 - SAEFKO ATTACK SYSTEMS V4.9, RAT07 - AHMYTH, RAT08 - ANDRORAT COMMAND LINE.

The dataset contains both positive (malicious traffic) and negative (normal traffic) samples as shown in figure 2. In total, over 500,000 samples were collected.

2.4 Model Training

In this study, the data was trained on a single machine learning algorithm: Random Forest Classifier. First, it was chosen based on the following criteria:

- High performance: Random Forest usually has very high accuracy in malware detection and it has been reported that in a number of studies, the rate goes beyond 90 percent.
- Robustness to overfitting: Aggregation of multiple decision trees to form the Random Forest algorithm as well as shuffling the data during tree construction make Random Forests especially resistant to the overfitting issue. This helps in avoiding over-fitting with respect to the input information.
- Handling of imbalanced data: Malware is normally represented by the minority class in malware classification, while benign software constitutes the majority class. This is a class imbalance that is however handled by the Random Forests.

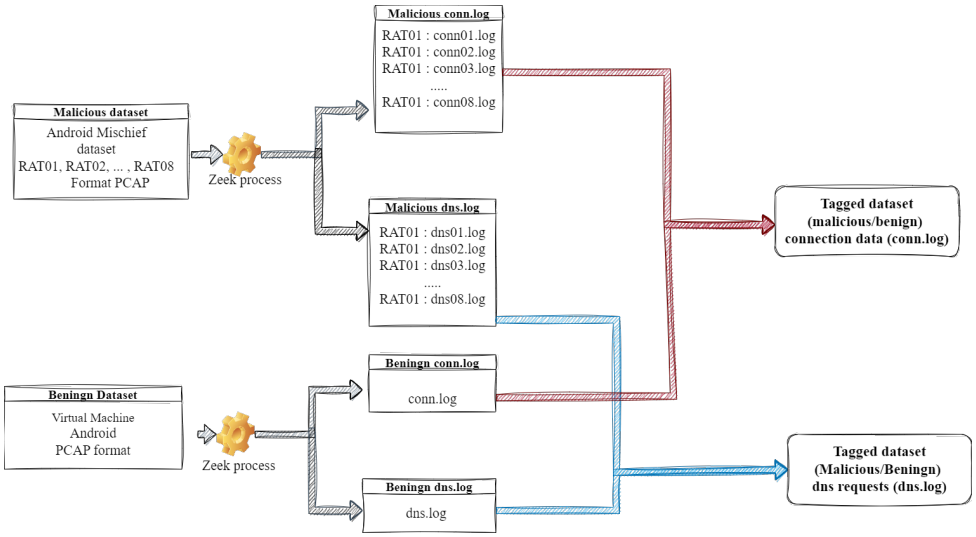

Figure 2: Dataset Creation

- Interpretability: In a sense Random Forest has some measure of interpretation through the tree rules as compared to pure neural network approaches. It enables one to determine the vital elements.
- Efficiency on large datasets: This makes Random Forest, an ideal tool for handling vast volumes of data collected from several Android apps, some carrying tens of thousands of attributes.

Afterwards, a general review showed it [7] suggested that random forest was best performing compared to the other methods depicted in figure 3.

The primary goal consisted in putting on footings to an independent, fully automated, real time leakage trace detection system. Therefore, greater importance focused on automation. Nonetheless, as a matter of fact, new algorithms will be used for comparison against their respective performances.

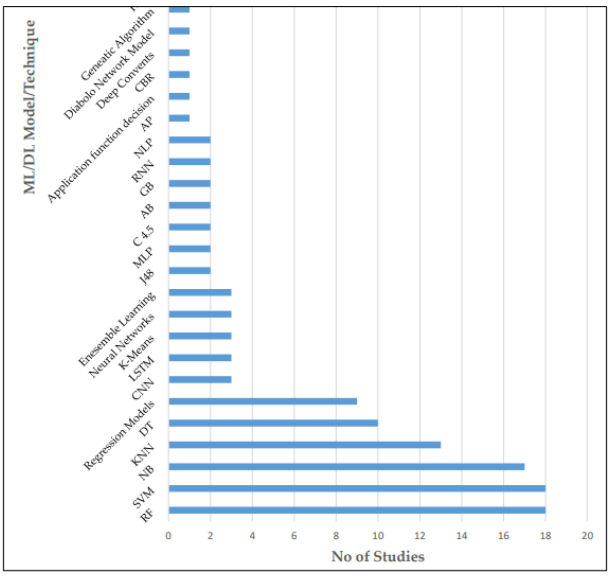

Figure 3: Performance comparison of different algorithms.

The training set of DeepDetector's Random Forest Classifier consists of 80 % of the available

	Recall	Precision	F-score
Connection data	89.98%	82.90%	86.29%
DNS irregularities	81.69%	78.91%	80.27%

Table 1: Model evaluation results.

dataset. This stage involves optimization of different parameters like number of decision trees. For this purpose, the rest of 20The Random Forest Classifier model was trained by optimizing the following hyperparameters via cross validation on the training dataset:

- Number of decision trees: Optimal number of trees to be tested between 10 and 500 is 100.
- Node split function: A comparison between Gini and Entropy has indicated that Gini performs better.
- Maximum tree depth: optimally with no specified depth limit, tested at least five and not more than twenty levels.
- Minimum number of samples per node: optimized between 1-10; it is best at 2
- Evaluation criterion: accuracy vs. Optimization of F1 score, better performance.

Finally, it should be noted that the model was trained on two datasets (so there are two models): conn.log and dns.log detect connection data and DNS queries respectively. Thus, the reasons behind having results concerning anomalous connections traces as well as anomalous DNS requests in the visualization of the results is that they are presented in table 1.

Some key features required for effective detection of data leaks.

In the detection of data leaks using machine learning algorithms, it is possible to leverage a number of relevant network traffic parameters for anomaly identification and leak pattern recognition.

For our random forest model, we utilized some of these features collected from Zeek network logs in training our DeepDetector. As such, the model is designed to identify atypical combinations of these attributes in order to expose potential data compromise. Some of these features are:

- Traffic statistics per connection: Any volume of bytes sent or received that appears abnormal is suspect.
- Connection duration: Any unusual duration of these connections is deemed suspicious.
- DNS query sizes: Unusually large lengths of DNS queries are deemed uncommon and regarded as suspect ones.
- Source and destination IP addresses and ports: Other types of suspicious connections include normally closed ports.
- Timestamp of events: Timestamp consistency is important for leak detection.
- Geolocation: Such connections might be abnormal since they are not coming from their usual geographical origins.

The categorical information of our data set was converted into digital and uploaded to the algorithm. This is what informed the Random Forest classifier on the important aspects of consideration it would focus on. Proper selection of features and a selected appropriate algorithm can help in effective anomaly detection. DeepDetector was indeed developed with special focus on such aspects.

2.5 Source Code

The complete DeepDetector source code is available on GitHub at: https://github.com/Beninwende/Dataleak_detector_by_Machine_Learning. All Python scripts that collect data, pre-process it, train ML models, and detect anomalies are located in this directory, shown in Figure 4. Additionally, it encompasses Jupyter notebooks utilized for first prototyping and descriptive data analytics. These notebooks describe in detail each of the key development steps: data preprocessing, training, tuning, testing and validation. The code in comment is all structured so that anyone can understand it, rewrite something for the betterment of the system or improve what already exists. Lastly, the steps of running this system on Raspberry Pi are explained.

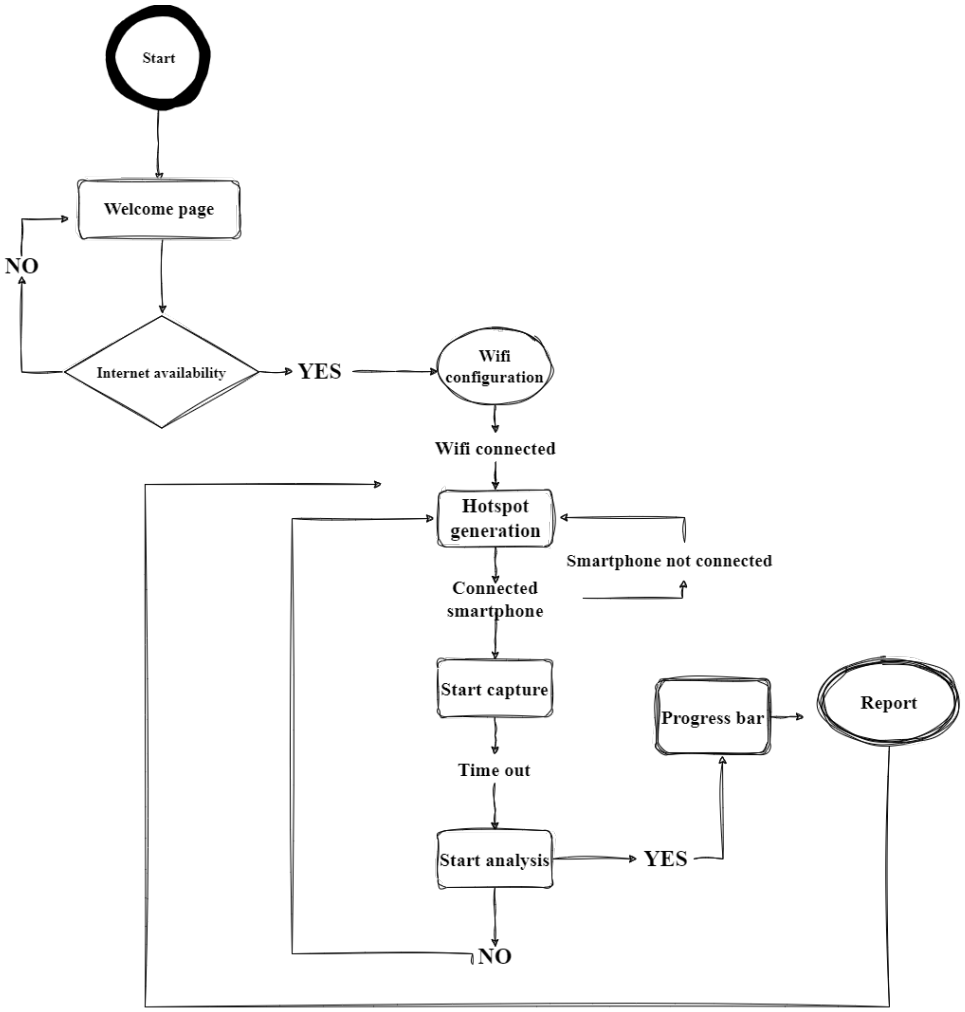

Figure 4: System operation.

III COMPARATIVE STUDY

This is part of the comparative study for various methods of android malwar detection in dynamic approach AI for 2017-2023. Several criteria are compared: using the detection process applied, choosing the appropriate algorithm, model's accuracy, as well as the pros and cons of each study. Datasets consist of DREBIN, Andro360, AMD, ML-Android, and Android Mischief dataset with up to 500,000 samples. Detectability depends on employed methods, with

detection rates ranging between 83 - 99%. Their performance in high performance is evident. Nevertheless, there are drawbacks associated with each method. Refer to Table 2 in appendix.

Its strengths include DeepDetector, among other things. First of all, Deep Detlector uses a dynamic technique for detecting leaks through the analysis of live traffic. This applies machine learning algorithms (Random Forest trained on healthy and infected network data).

Its main strengths are:

- Dynamic real-time approach, Modifiable and extendible architecture involving the integration of new ML models.
- Supervised training

IV RESULTS

The trained Random Forest model achieves 82.9% precision and 89.9% recall in classifying network connections, and 78.9% precision with 81.6% recall in detecting abnormal DNS requests.

These results were obtained using the following criteria in building the decision trees:

- Number of trees: optimal at 100
- Split hyperparameter (data split): GINI (measure of impurity)
- Number of features considered at each split: optimal with sqrt
- Maximum tree depth: optimal with no depth limit
- Minimum number of samples per node: optimal at 2

These performances are obtained on the 20% test dataset of samples not used for training. They demonstrate the good generalization capabilities of the model to detect anomalies in the network traffic of unknown Android devices.

In addition, the DeepDetector system operates in a fully automated manner to:

- Collect Android device traffic in real time via the Raspberry Pi configured as a WiFi access point.
- Generate the training dataset from the Zeek logs.
- Train the Random Forest model with hyperparameter optimization.
- Evaluate performance on the test dataset.
- Analyze traffic with the trained model and report detection results to the user in graphical interactive form.

This complete automation of data collection, processing and analysis steps is a key advantage of DeepDetector for practical field use.

V DISCUSSION

This achieved shows promise for application of artificial intelligence methods in particular supervised machine learning in the detection of data leakage from Android smart phones.

A solution DeepDetector – free, easy to use and efficient for increased users' security and privacy against the growing risk of malware.

Several improvements can still be made to the system:

- Increase the size and variety of the training dataset to reinforce model robustness.
- Optimize model parameters and architecture to improve performance.
- Add new software sensors to enrich the data collected.

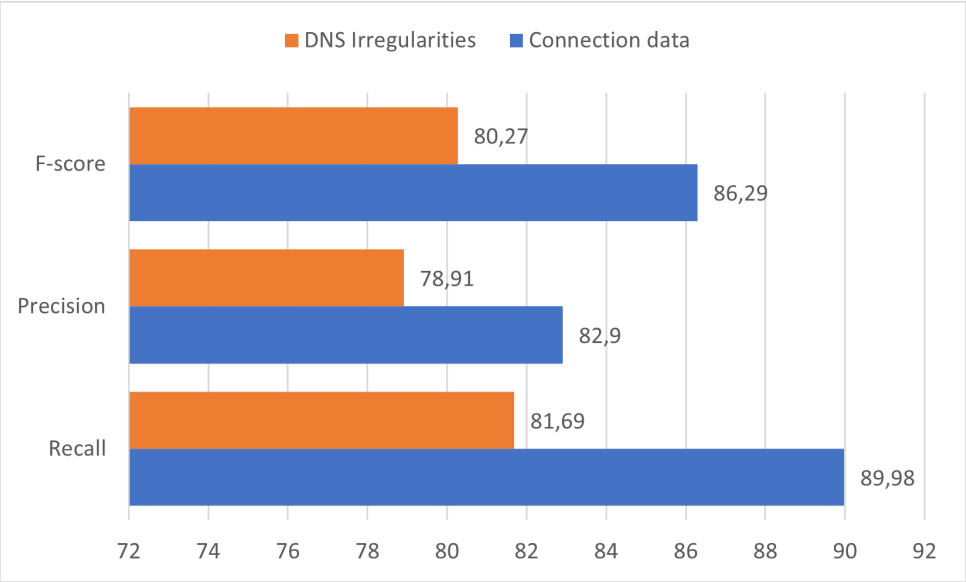

Figure 5: Model evaluation results.

- Implement advanced techniques like deep learning.
- Deploy the system on a cloud platform for scaling.

This work demonstrates the potential of artificial intelligence for protecting mobile data and paves the way for developing ever higher performing future data leak detection systems.

VI CONCLUSION

Here, we introduced Deepdetector – an artificial intelligent-based detection system of data leakage that can be run purely on Raspberry Pi by itself in order to avoid such incident on mobile phones. These findings showed promise for detecting anomalies in network traffic (80% precision) and DNS queries (90% recall).

In this regard, DeepDetector proves the capability of supervised machine learning methods in securing people's sensitive information from the increasing danger posed by mobile malware.

Adding more and different information will further enhance the results in future implementations. Another way of improving this collection is by adding new software sensors for additional data enhancement.

In essence, DeepDetector might eventually be considered universally available efficient antimobilism measure. Automatic data leak detection would further enhance users' privacy protection.

Deep detector's contribution is described in the conclusion that summarises the main results. This as well creates avenues for future work to further enhance the system. uitgen.

ACKNOWLEDGEMENT

This work was conducted as part of the Artificial Intelligence for Development in Africa (AI4D Africa) program, with the financial support of Canada's International Development Research Centre (IDRC) and the Swedish International Development Cooperation Agency (Sida).

VII ANNEX 1

REFERENCES

[1] "Mobile Operating System Market Share Worldwide | Statcounter Global Stats," Statcounter Global Stats, [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide. Accessed on: Feb. 14, 2023.

[2] H. Zhang, P. P. Chan, and N. M. Cheung, "Android Malware Detection Based on Generative Adversarial Network," *Neural Computing and Applications*, 2023.

[3] M. Zheng, J. Lee, and J. Jeong, "Android Malware Detection Using Convolutional Neural Network," *Symmetry*, vol. 13, no. 11, p. 2110, 2021.

[4] "Android Mischief Dataset," Stratosphere IPS, [Online]. Available: https://www.stratosphereips.org/android-mischief-dataset. Accessed on: Oct. 29, 2023.

[5] "Deep Learning for Android Malware Defenses: a Systematic Literature Review," [Online]. Available: https://www.researchgate.net/publication/349943796 Deep Learning for Android Malware Defenses a Systematic Literature Review.

[6] S. Garg, S. K. Peddoju, and A. K. Sarje, "Network-based detection of Android malicious apps," *Int. J. Inf. Secur.*, vol. 16, pp. 385–400, 2017.

[7] A. K. Sikder, H. Aksu, and A. S. Uluagac, "6thSense: A Context-Aware Sensor-Based Attack Detector for Smart Devices," in *Proc. 26th USENIX Secur. Symp.*, Vancouver, BC, Canada, 2017, pp. 397–414.

[8] M. Salehi, M. Amini, and B. Crispo, "Detecting malicious applications using system services request behavior," in *Proc. 16th EAI Int. Conf. Mobile Ubiquitous Syst. Computing, Networking Services*, Houston, TX, USA, 2019, pp. 200–209.

[9] R. Thangavelooa, W. W. Jinga, C. K. Lenga, and J. Abdullaha, "DATDroid: Dynamic Analysis Technique in Android Malware Detection," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 10, pp. 536–541, 2020.

[10] J. Lee, S. Park, and J. Jung, "Detecting malicious behavior in Android apps through analyzing inter-app information flows," *Expert Systems with Applications*, vol. 189, p. 116124, 2022.

[11] Y. Song, C. Liu, and C. Zhang, "Android Malware Detection System Using Isolation Forest," *IEEE Access*, vol. 10, pp. 53372-53379, 2022.

Year	Study	Detection approach	Selected ML Algorithm	Model Accuracy	Strengths	Limitations/Drawbacks
2017	[7]	Extracting DNS, HTTP, TCP, Origin based network features used by apps	RF	98%	Works with different OS versions, Detects unknown malware, infected apps	If malware apps use encryption, not possible to properly detect malware
2017	[8]	Using Dynamic permission analysis with runtime and detect malware using ML calculate accuracy	Simple Logistic	99.70%	High Accuracy	Need to address app crashing issue in selected emulators in dynamic analysis
2019	[9]	Dynamically tracking execution behaviors of applications using ServiceMonitor framework	RF	96.70%	High accuracy and efficiency	Not detecting difference in some system calls of malware and benign apps since signature based verification was not applied
2020	[10]	Extracting features and permissions from Android app. Performing feature selection and classification with DATDroid	RF	91.70%	High efficiency	Impact from features like HTTP, DNS, TCP/IP patterns not considered
2021	[11]	Android Malware Detection Using Random Forest Based on API Call Sequences	RF	97.40%	High Accuracy	DEREBIN dataset size limited to 120000 samples
2022	[12]	Android Malware Detection with Deep CNN-LSTM Model	CNN-LSTM	99.10%	High Accuracy	Andro360 dataset size limited to 360000 samples
2022	[13]	Android Malware Detection System Using Isolation Forest	Isolation Forest	94.20%	High Accuracy	AMD dataset size limited to 25000 samples
2023	[14]	Android Malware Detection Based on Generative Adversarial Network	GAN	96.80%	High Accuracy	ML-android dataset size limited to 25000 samples
2023		Detecting illicit data leaks on Android smartphones using an AI model (DeepDetector)	RF	83.34%	*Modular architecture *Expandable training and test dataset *Fully automatic and autonomous system	*Does not identify leak source Trained on single ML algorithm

Table 2: Comparative study of Android malware detection techniques.