YAAMMLS: Yet Another Android Malware Machine Learning Survey

Patrick DeCoster
Department of Computer Science
Illinois Institute of Technology
Chicago, Illinois
pdecoster@hawk.iit.edu

Charles Melis
Department of Computer Science
Illinois Institute of Technology
Chicago, Illinois
cmelis@hawk.iit.edu

Mohammadreza Sediqin
Department of Computer Science
Illinois Institute of Technology
Chicago, Illinois
msediqin@hawk.iit.edu

***Abstract*—The Android operating system has benefited from rapid and expansive adoption. Its widespread use is attributed, in part, to its openness and customization. These attributes have also made Android a prominent target for malware as the platform endures additional complexity in securing its systems. The attackers also target the Android platform because it is based on a known kernel (Linux). Manufacturers offer various life cycles for system updates and security patches, resulting in devices across the entire ecosystem operating on different operating system (OS) versions and Application Protocol Interfaces (APIs). Additionally, users can install applications from alternative sources with differing degrees of scrutiny on the Application Package Kit (APK) installer. These factors complicate the security model for protecting Android devices from malware. Due to the substantial overhead required to properly analyze and investigate potential malware detection methodologies, researchers began leveraging Machine Learning (ML) and Deep Learning (DL) technologies. Several prior studies have demonstrated significant success in identifying malware with these techniques. Implementations of these techniques have differing algorithms, classifications, datasets, feature selections, and static and dynamic analysis for training ML and DL models. Each can have a significant impact on the performance of a particular model. Consequently, malware producers do not remain idle. Malware producers develop more complicated and sophisticated evasion techniques as detection methodologies improve. This survey aims to investigate the current ML and DL landscape, the importance of feature selection when training these models, the datasets representing benign applications and malware, the performance of various models, and identify future challenges researchers may encounter.**

***Index Terms*—android, android security, malware, machine learning, deep learning**

I. Introduction

The widespread use of Android-based mobile devices has led to a significant increase in the number of Android applications in the market. However, with the growing number of applications, the risks of malware attacks on these devices have also increased significantly [1]. In this context, detecting and preventing malware in Android applications has become a crucial concern in mobile security. This survey intends to provide a comprehensive review of the existing literature on malware detection in Android applications. Further, this study will permit the analysis of the different techniques proposed for malware detection, including static and dynamic analysis, machine learning, and behavior-based detection, and their strengths and limitations.

Moreover, this survey will provide recommendations for future research directions. Despite the advances in techniques, detecting malware in Android applications remains a significant challenge due to the evolving techniques used by malware developers to evade detection. Therefore, further research is needed to develop more effective and efficient techniques to detect and prevent malware in Android apps.

Malware poses significant data privacy and security threats for individuals and corporations alike. With companies implementing Bring Your Own Device (BOYD) policies allowing employees to access company networks, data shares, and email, malware can significantly impact a company financially, legally, and reputationally. Over 350,000 new malware samples are detected daily, with Android being the most targeted mobile platform [1]. Therefore, detecting and preventing malware within the Android ecosystem has become a crucial concern in mobile security.

Researchers continue to conduct extensive research on malware detection methodologies and prevention. Various techniques have been proposed, such as static and dynamic analysis, machine learning, and behavior-based detection [2] [3] [4]. These techniques aim to detect malicious behavior and patterns common to multiple malware apps to aid in the detection and prevent malware from infecting the device.

Despite the advances in malware detection techniques, detecting malware in Android applications remains a significant challenge. These arise when implementing malware detection with ML and DL as multiple algorithms, features, and training possibilities exist. Further, malware developers continually evolve their applications to evade detection, making it difficult for existing detection methods to keep up. Therefore, further research is needed to develop more effective and efficient techniques to detect and prevent malware in Android apps.

The paper aims to review the existing and state-of-the-art malware detection methodologies and assess performance, shortcomings, and challenges. This paper plans to provide recommendations for future research directions in this field.

II. Motivation

Malware on Android devices can lead to data theft, financial loss, identity theft, and other serious consequences. Numerous applications with access to confidential or sensitive information access and store data alongside other benign and potentially malicious applications. Users generally lack a robust understanding of application permissions and their implications for approving and allocating permissions appropriately. Further, user requests for identification and authorization are standardized (e.g., pin prompt or fingerprint prompt) on these devices, allowing users to be easily tricked into providing privileged permissions to the wrong applications. Due to this risk, effective measures to detect known, but importantly new and unknown variants, and malware constructed to evade traditional detection techniques, is critically important.

Traditional malware detection techniques include static analysis, dynamic analysis, package analysis, runtime observation, network traffic analysis, hashing and signature-based analysis, and permission analysis. To counter these methods, malware producers continue to evolve their programs to evade detection and obfuscate intention. Polymorphism evasion strategies include encryption of data, bytecode, and payloads, as well as package transformation in the form of repackaging and renaming. Metamorphism evasion strategies include sandbox or Virtual Machine (VM) awareness, user interaction detection, code obfuscation via code reordering, call indirection, or dead code insertion, and code transformation in the form of reflection and dynamic loading.

ML and DL have emerged as powerful tools in the fight against malware, providing unique abilities to detect features and behaviors often difficult to replicate by other detection methods. With the proper feature selection, training, and algorithm selection, ML and DL have the potential to effectively detect complexly obfuscated known malware, new unknown variants of malware, and zero-day malware, as they share similar properties that neural networks can extract.

Researchers have implemented ML and DL algorithms to combat these evasion strategies from malware authors. The effectiveness of these techniques depends on the quality of the datasets and selected features. Large datasets of benign apps and known malware are required to train these models. These models rely upon extracted features from the APKs through the detection techniques mentioned above. Applying ML and DL to these comprehensive datasets allows for robust pattern recognition between various types of malware, allowing for advanced detection.

Due to the complexity of feature selection and model creation, many potential avenues exist to explore. Various ML and DL algorithms are in use, with potentially new algorithms to be utilized. Models can be trained with varying techniques, including supervised, unsupervised learning, and ensemble methods. Additionally, a variety of datasets have been used to train models. Each of these models has differing performance characteristics and accuracies. Our motivation is to evaluate these techniques' performance using various metrics, such as accuracy, precision, and recall.

Ultimately, there is a need to stay ahead of the constantly evolving threats posed by malware and to provide effective solutions to combat them. The analysis of existing research and new and emerging research has the potential to provide insights when working with these technologies. This survey's findings hope to contribute to developing more robust and effective ML and DL processes for detecting Android malware, enhancing the security of the Android ecosystem, and protecting sensitive data.

III. Methodology

With the objective of providing an up-to-date Android malware detection survey using ML and DL. We will first provide an overview of the types of malware in the Android ecosystem. Next, we will provide a synopsis of traditional malware detection and the methods used by malware designers to evade and circumvent detection. Then, we will review the various ML and DL methods of malware detection, their algorithms, feature selection, and data sets.

A. Discovery

Through the review of existing research papers, journal articles, and conference proceedings, this survey aims to provide a comprehensive analysis of the current state-of-the-art techniques used for detecting malware using ML and DL algorithms. Sources for the relevant articles and papers include the *Association for Computing Machinery*, *Google Scholar*, *Institute of Electrical and Electronics Engineers Xplore*, and *Network and Distributed System Security (NDSS) Symposium*. The search utilized keywords, phrases, and combinations thereof, such as *Android malware*, *malware dataset*, *deep learning*, *machine learning*, and *malware detection*.

B. Selection

All articles and papers reviewed were written in English or translated into English. The relevant time period for these articles was from 2010 through March 2023.

C. Contributions

Our contributions are a overall analysis of the current state of malware detection with on the Android platform, a review of the performance of these models, a review of the datasets commonly used by researchers, and an analysis of areas that may impact future growth with current implementation and design of these models.

IV. Results

A. Malware

Malware in the Android ecosystem falls into several categories, including but not limited to adware, backdoor, file infector, malware, ransomware, riskware, scareware, spyware, and trojan [5]. While this list is not exhaustive, it represents a significant portion of the malware that targets users, degrades users' experiences, compromises privacy and security, and targets financial accounts.

1) Adware pushes unwanted, misleading, and malicious advertisements on the user's screen. Intentionally or unintentionally clicking an advertisement can lead to unwanted or potentially harmful software installed on the user's device.
2) Backdoors allow bad actors to access a user's devices usurping required authentication or granting itself privileged permissions on the device.
3) File infectors are attached to APKs and are executed at the time of the installation of the APK. These can have a range of effects on a device, from accessing to deleting private information, installing or removing apps or malware, or seeking elevated privileges.
4) Generic malware is any software that causes unwanted or damaging behavior to the device. An example would be cryptojacking malware that uses the device to mine cryptocurrency for an unknown party.
5) Ransomware encrypts a user's files with a key that can only be retrieved by paying a ransom to an unknown actor. The ransom is typically paid in a cryptocurrency. If the ransom goes unpaid, the user's files can be unrecoverable.
6) Riskware are apps installed at the insistence of the user but still pose risks to the user's privacy or security. While the user may desire the app, riskware may attempt to steal information, be adware itself, or perform unwanted actions on the phone.
7) Scareware are typically applications that trick users into installing them by using fear. These applications may pretend to act as antivirus or other security software, warning that the device is compromised.
8) Spyware compromises the user's privacy by stealing information from all sources on the phone. This information could include the geo-positional location, phone call or text message information, and other user-identifying information. This information is then sold to advertising firms.
9) Trojans are types of malware that can purport to be legitimate applications but steal information from the device. Several subclasses of trojans on the Android ecosystem include banking, sms, and spy.

B. Malware Obfuscation Techniques

Signature-based detection is a highly-effective and widely-implemented method for uncovering malware and viruses. Fingerprints based upon binary or textual patterns are created by scanning known malware. Antivirus or anti-malware software examines files on a system and compares the signatures of the files it encounters to the signatures of known malware from the database. This method is hampered as it requires having a known piece of malware to create signatures and is subject to code obfuscation techniques implemented by malware designers to hide or scramble signatures. Fingerprinting is an ineffective method to keep up with the growing quantity of malware, to detect zero-day malware and advanced malware code obfuscation.

Static analysis performs a white box analysis by using reverse engineering on the APK and starts an analysis of the code of this APK, scanning for the key features.

Dynamic analysis, a type of black-box analysis, runs the APK in a controlled environment and monitors the behavior of the app.

1) Polymorphism: Polymorphism is a widely used class obfuscation technique employed by malware developers to evade signature-based and static analysis detection techniques. This technique involves encrypting an application's bytecode, data, or payload, which transforms the program's static components into unique identifiers after each execution, making it difficult to detect by antivirus and intrusion detection systems.

One of the most commonly used polymorphism techniques is package transformation, which involves renaming identifiers and packages and repackaging the malware. For instance, by renaming the variables, functions, and class names, developers can make it harder for static analysis tools to detect the malware, even if they have the same functionality as the original code. By changing the package name, the code can appear as a new or different application. Repackaging involves modifying the APK or executable file to avoid detection by signature-based detection techniques.

2) Metamorphism: Metamorphism is another class of obfuscation techniques malware developers use to evade detection by antivirus and intrusion detection systems. This technique includes several strategies, such as anti-emulation transformation, code obfuscation, and code transformation.

Anti-emulation transformation involves modifying the malware's behavior to evade detection when run in a virtual environment. This technique can be achieved by detecting whether the malware is being run in a virtual machine or sandboxed environment and changing its behaviors accordingly.

Code obfuscation is another technique employed in metamorphism, which involves manipulating the malware's runtime binary by reordering the code, inserting dead code, and using call indirection. These techniques can make the signature detection process more challenging for static analysis tools, as the code structure is significantly altered from the original.

Code transformation is another technique used in metamorphism. This process includes dynamic code loading, modifications, or reflection to change the behavior of the malware at runtime. These evasion measures make it more difficult for dynamic analysis tools to detect and analyze the malware as it behaves innocently. [6].

C. Deep Learning and Machine Learning Models

1) Overview of Models:

a. Decision Trees (DT): Models based on DTs learn to classify with a tree structure. Nodes on the tree represent features and permissions in the dataset. The model analyzes a feature set and works through the decision tree. A determination is made when the tree reaches a leaf. Decision trees are easy for humans to interpret and work well with large data sets. Small changes in

a dataset can impact a DT, significantly impacting the overall tree, and they are susceptible to overfitting.

b. Support Vector Machine (SVM): Models that employ SVM make determinations by representing the data in multi-dimensional space. When SVMs attempt to make a determination, the algorithm tries to maximize the margin between classes on the plane. SVMs solve non-linear problems and work very well when there is a clear margin of separation between the classes. SVMs can struggle with larger datasets with a high feature count or missing values.
c. Naive Bayesian (NB): NB is a relatively simple classification technique that leverages Bayes' theorem. It assumes that there are independent predictors and features, meaning a given feature will not be present in another class.
d. Random Forest (RM): RM follows a similar methodology of decision trees, but with an ensemble of trees. The final classification occurs by choosing a final tree among all trees created. RM can correct the problems of overfitting found in traditional decision tree models. RM has a higher overhead due to the number of trees that can be created.
e. Convolutional Neural Networks (CNN): A DL algorithm intended to be used with image and video data to extract relevant features and classify them into categories. CNNs have an input layer, several hidden layers, and an output layer composed of neurons. The process of creating a CNN involves training on data and assigning weights and biases to each neuron with an activation function. Connections through the network are fed forward through the network. The strongest node in the output layer is the determination of the CNN model.
f. Recurrent Neural Networks (RNN): A DL algorithm that implements a neural network of nodes similar to CNNs. While CNNs are a one-way feedforward network, RNNs allow for looping back, where the model feeds results back into the network. These algorithms are typically used for text or video analysis.
g. Long Short-Term Memory (LSTM): A DL algorithm similar to RNN with feedback connections expanding the model by adding a short-term memory to the composted network. This ML algorithm is designed to work in the areas such as speech recognition, handwriting recognition, and machine translation.

2) Feature Selection: Feature selection is a crucial step in building these models. Identifying relevant features in the dataset will help the model make accurate predictions for APKs that contain malware. Proper feature selection can improve the model's performance, reduce overfitting, and make it more interpretable. Conversely, choosing the wrong features can lead to poor performance and inaccurate predictions.

Researchers of the paper "Analysis of Feature Selection Techniques for Android Malware Detection" studied feature selection. They approached the problem by not targeting which features are best to extract because this is domain and dataset-specific. Instead, they show how ML model accuracy varies across feature selection algorithms, how feature set size affects the model accuracy across feature selection methods, and the primary guidelines for building a feature set in Android malware detection. Ultimately they show that the feature set size is essential. Generally, the best results are achieved with feature counts of 200 or greater. If a feature set is larger than the minimum required, accuracy does not decrease. It is safe to use more features than required when in doubt [7].

The Relief feature extraction evaluates the quality of each feature by measuring how well it distinguishes between examples of different classes. Relief has shown to be a poor performer in a large dataset due to significant randomness. If the Relief performs poorly, the CFS Relief also performs poorly. The features it selects are significant, but the algorithm stops too soon, thus providing an insufficient feature set size. The RF with Perceptron as a feature selection method is unpredictable. While the accuracy of the results were acceptable in these experiments, the repeatability was poor, implying that not all feature sets selected would be acceptable.

Information Gain, Chi-Square, Ridge Regression, and LASSO Regression are excellent feature selection methods with minimal variance across machine learning classifiers. They all include features' weight factors for a large percentage of the feature input vectors. All the feature selection methods cited above have 99.7% accuracy with a RF classifier, 99.2% accuracy with a SVM classifier, and 98.8% accuracy with a Neural Net (Perceptron) classifier.

Based on the papers we encountered, it is evident that using feature selection techniques for malware detection and classification can result in high accuracy rates. The approach proposed by Islam (2023) et al. achieved the highest reported accuracy rate of 98.15% for detecting Windows malware, followed by the approach proposed by Sun et al(2022). with an accuracy rate of 98.5% for detecting Android malware [8] [9]. FSDroid achieved an accuracy rate of 97.34% for detecting Android malware using a genetic algorithm for feature selection. However, it is important to note that the performance of different approaches may not be directly comparable, as they may have used different datasets, feature selection techniques, and evaluation metrics [10]. Nonetheless, the promising results suggest that feature selection techniques can be effective in improving the accuracy and efficiency of machine learning models for detecting and classifying malware. The selection of appropriate feature selection techniques and evaluation metrics is crucial for achieving optimal performance and generalizability of the models. Overall, these findings highlight the potential of feature selection techniques in enhancing the efficacy of machine learning approaches for malware detection and classification.

3) Datasets: Machine learning and deep learning require significant training to perform well. Repeated use of datasets occurs across studies. The datasets most likely in use are AMD, AndroidMalGenome, AndroZoo, DroidCollector, and

Drebin. An examination of the Drebin dataset reveals it was created in 2013. At the time of its creation, Drebin's researchers argued that existing datasets in 2014 were insufficient for training the models because they were small, outdated, or limited in scope [2]. The Drebin dataset contains 120,000 labeled malicious APKs. Following its creation, fellow researchers' adoption of Drebin data set followed. Unfortunately, repeatedly using the same dataset allows malware creators to develop software that can evade detection by avoiding features found in the Drebin dataset.

In further studies into the DREBIN, researchers found that the dataset contains many samples that are very similar to each other, which may affect the generalizability of models trained on the dataset. Secondly, they found that many features in the dataset are highly correlated, which may lead to overfitting when using machine learning models. Thirdly, they found that some features in the dataset are highly indicative of malware, while others are less so, suggesting that future work should focus on identifying the most relevant features for malware detection [11]. The Drebin dataset did not continue to update its APK malware database after it was announced.

Other researchers have investigated how biased datasets can lead to biased trained models. Datasets are biased towards certain families of malware, certain types of malware behaviors, and certain features used to represent Android apps. The authors proposed a methodology for measuring and mitigating dataset bias in Android malware detection. They suggest that researchers carefully select and preprocess the data, use appropriate evaluation metrics, and conduct experiments with diverse models. Some datasets can have homogeneous data. They showed that debiasing Drebin requires 1.8% to 12.7% new samples, and debiasing VirusShare requires 0.15% to 2.9% new samples [12].

Using older datasets can leave gaps in both the APK versions that are currently in use by modern applications, and lack samples of sophisticated malware that was evolved. Researchers need to be aware that even newer datasets can reply upon samples of malware from older datasets or be compilations of several datasets. For example, newer datasets, like CICMalDroid2020 which contains 11,598 APK collected from several sources, including VirusTotal, Contagio , AMD, MalDozer, and other datasets from recent research contributions [38].

4) Performance: Table I. compares the performances of different research papers that explore the use of machine learning and deep learning models for detecting Android malware. Comparing the performances, it is clear that the use of deep learning models, such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), have generally led to higher accuracy in detecting Android malware than the use of traditional machine learning models, such as Logistic Regression, Random Forest, and Decision Trees.

For example, the research paper by Vina et al. (2017) achieved an accuracy of 98.1% with RNN and 99.8% with LSTM, respectively [35]. Syrris et al. (2022) achieved an accuracy of 95.86% using traditional machine learning models such as Logistic Regression, Random Forest, and Decision Tree. Similarly, the research paper by A. Kim et al. (2021) achieved an accuracy of 98.3% using a combination of CNN and RNN. However, it is important to note that all of these methods are limited to specific malware families and may not perform well on previously unseen malware. Additionally, some of these approaches require a large amount of training data, which may not always be readily available. Therefore, future research is needed to improve the effectiveness and generalizability of these approaches.

Many studies have shown that LSTM models have good approaches to detect and classify malware, as evidenced by the papers we read. All of the LSTM-based approaches achieved high accuracy rates for detecting and classifying Android malware. The approach proposed by Vinayakumar et al. achieved the highest reported accuracy rate of 99.4%, followed by the approach proposed by Alkahtani et al(2022). with an accuracy rate of 98.6%. However, it is important to note that the accuracy rates reported by different papers may not be directly comparable, as they may have used different datasets and evaluation metrics. Nonetheless, the promising results suggest that LSTM-based approaches can be effective in improving the security of Android devices [35] [39].

Studies have shown that high accuracy rates for detecting and classifying Android malware using the Naive Bayes algorithm as well. YILMAZ et al.(2022) achieved the highest reported accuracy rate of 98.06%, followed by Sahin et al(2022). with an accuracy rate of 97.9%, and Ab Razak et al. (2022) with an accuracy rate of 96.1%. However, it is important to note that the performance of different approaches may not be directly comparable due to variations in datasets and evaluation metrics. Nonetheless, the promising results indicate that the Naive Bayes algorithm can be effective in detecting and classifying Android malware. The selection of appropriate features and evaluation metrics is crucial for achieving optimal performance and generalizability of the models.

In addition to high accuracy rates, the relted papers reported other performance metrics, such as precision, recall, and F1-score. YILMAZ et al.(2022) reported a precision of 98.9%, a recall of 98.2%, and an F1-score of 98.5%, while Sahin et al(2022). reported a precision of 97.9%, a recall of 97.9%, and an F1-score of 97.9%. Ab Razak et al. (2022)reported a precision of 94.4%, a recall of 94.2%, and an F1-score of 94.3%. These results suggest that the Naive Bayes algorithm can not only achieve high accuracy rates but also perform well in terms of precision and recall, which are important metrics for evaluating the performance of a malware detection and classification system [40] [41] [42].

5) Limitations: Overall, using machine learning and deep learning algorithms for Android malware detection has shown promising results. However, several limitations and challenges must be addressed to ensure the effectiveness and security of these approaches.

One major limitation is the availability of high-quality datasets that accurately represent real-world malware threats. Many existing datasets are limited in size and may not include

TABLE I
MODEL PERFORMANCE

Ref	Year	Model(s)	Accuracy	Precision	Recall	Dataset
[25]	2021	SVM	96.2%	98.1%	94.4%	CICAndMal2017
		KNN	97.2%	98.3%	96.0%	CICAndMal2017
		DT	96.6%	96.2%	97.3%	CICAndMal2017
		RF	97.8%	98.8%	97.2%	CICAndMal2017
		NB	93.9%	94.3%	93.5%	CICAndMal2017
		GRU*	98.2%	96.9%	99.2%	CICAndMal2017
[27]	2020	Ensemble (DBN+Genetic)		97.9%	98.5%	Original v1
		Ensemble (DBN+Genetic)		98.3%	98.1%	Original v2
[29]	2016	DBN*(S+D)	96.76%	97.79% / 95.77%	95.68% / 97.84%	Contagio+Genome
		DBN*(S)	89.03%	97.77% / 83.00%	79.89% / 98.18%	Contagio+Genome
		DBN*(D)	71.25%	67.09% / 78.08%	83.41% / 59.09%	Contagio+Genome
		SVM (S+D)	82.84%	93.63% / 92.08%	91.93% / 93.75%	Contagio+Genome
		NB (S+D)	83.86%	90.27% / 79.22%	75.91% / 91.82%	Contagio+Genome
[32]	2019	MNN	98.0%	98.0%	99.0%	VirusShare+Malgenome
[34]	2019	NB	92.1%	93.5%		Genome
[35]	2017	RNN (1 Layer)	98.1%	100.0%	97.7%	CDMC2016
		RNN (2 Layer)	96.7%	100.0%	95.9%	CDMC2016
		LSTM (1 Layer)	99.8%	99.9%	99.8%	CDMC2016
		LSTM (2 Layer)	98.2%	99.9%	97.8%	CDMC2016
		LSTM (2 Layer)	89.7%	91.0%	96.0%	CDMC2016 Real World Test
[36]	2021	NB	95.5%	99.7% / 49.0%	95.6% / 96.6%	Drebin
		RF	99.4%	99.5% / 97.4%	99.9% / 89.1%	Drebin
		SVM	99.7%	99.8% / 97.8%	99.9% / 95.6%	Drebin
[37]	2022	kNN		97.23%	94.38%	Genome
		LR		93.4%	93.4%	Genome
		RF		97.45%	93.4%	Genome
[38]	2023	NB	65.04%			CICMalDroid2020
		SVM	83.30%			CICMalDroid2020
		RM	94.32%			CICMalDroid2020
		LGBM	95.49%	95.48%	94.70%	CICMalDroid2020

the latest and most sophisticated malware strains. Additionally, the feature selection challenge can significantly impact the performance of machine learning and deep learning models, requiring domain knowledge and expertise to select relevant and informative features for malware detection.

The dynamic nature of Android applications and the continuous evolution of malware tactics also pose challenges for developing robust and generalizable models that can accurately detect and classify new and previously unseen malware strains. Adversarial attacks on machine learning models used for malware detection are also a concern, particularly for deep learning models that are highly vulnerable to such attacks due to their complex and non-linear nature.

Another challenge is the need for extensive and diverse datasets to train machine learning models effectively, which can be challenging to collect and label as the landscape of Android malware continues to evolve. Additionally, using machine learning and deep learning for Android malware detection raises privacy concerns as these models may require access to sensitive information on the device, such as user data and system logs. Further research is needed to address these issues and develop more robust and secure approaches for Android malware detection using machine learning and deep learning algorithms.

D. Discussion

We saw that the need for detecting malicious APKs is still rising, and malware developers are finding more and more methods to avoid the detection.

The last decade of malware detection research utilizing machine learning and deep learning has been very successful. As a community, there are a host of implementations using various algorithms and feature selections. These models perform well and have become accurate with percentiles in the mid-to-high 90s. The performance of these implementations is due to several factors as researchers gained a greater understanding of the algorithms and data sets.

The computational power of models has become more complex. Implementations are built with hundreds of features extracted from datasets to potentially improve accuracy and performance.

Nevertheless, these improvements are occurring in the margins after over ten years. By their very nature, no machine learning model will be 100% effective at detecting or solving any task. However, the repeated nature of these studies suggests researchers are continuing to eek fractions of percentiles of improvements in their implementations.

Problems arise in a few areas. First, many of the datasets used by researchers are simply outdated. Drebin and Genome continue to be used as datasets by researchers, but these models have not incorporated any new forms of malware since 2016. Further, newer datasets collect samples from these as

TABLE II
MALWARE DATASETS

Dataset	Collection Period	Malware Samples
Android Malware Dataset [13]	2010-2016	24,550
Android Malware Genome [14]	2010-2011	1,200
AndroMalShare	defunct	×
AndroZoo [15]	2016-*Ongoing*	×
CICAAGM2017 [16]	unknown	426
CICAndMal2020 [17] [18]	unknown	200,000
Contiago Malware Dump [19]	2011-2013	189
Drebin Dataset [20] [21]	2010-2012	5,560
InvesAndMal2019 [22]	426	2017

well. Malware in use today are likely being developed with greater sophistication to evade detection.

Second, the Android APK experienced several updates in this timeframe, which reflects back to dataset bias. Take a hypothetical training dataset. Suppose all of the benign software for a dataset is from the period of 2020 to current while all of the malware are from the Drebin dataset. In that case, the model will likely determine that all malware contain old API calls that no longer exist in newer applications due to updated API versions.

Another problem is that these machine-learning algorithms' original design and usage were for language models and image processing. It is impactful what researchers have achieved using these models, stretching their usage into new areas. However, pushing these models further might lead to only marginal gains. Reviewing newer studies shows this is precisely the type of improvement we see in these models' later implementations.

E. Conclusion

In our study we reviewed the current state of the art methods for malware detection being implemented by researchers. While these implementations are highly accurate, with the current methodologies used by researchers, improvements to ML and DL might have marginal increase going forward. Additionally, the datasets where are still in common use by researchers are outdated and prone to bias.

REFERENCES

[1] AV-TEST. (2021). Security Report 2020/2021. Retrieved from https://www.av-test.org/en/statistics/malware/

[2] Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H., & Rieck, K. (2014). DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. Proceedings of the 23rd USENIX Security Symposium.

[3] Sahs, J., Khan, L., Mariconti, E., & Onwuzurike, L. (2017). Malware Detection on Android: An Investigation of Machine Learning Techniques. Proceedings of the 16th Conference on Detection of Intrusions and Malware & Vulnerability Assessment.

[4] Xu, S., Zhang, Y., Wang, Y., & Qiao, M. (2018). Detection of Android Malware by Analyzing Requested Permissions and Application Metadata. IEEE Access, 6, 29187-29199.

[5] Understanding Android Malware Families (UAMF) – The Foundations. https://www.itworldcanada.com/blog/understanding-android-malware-families-uamf-the-foundations-article-1/441562

[6] Elsersy WF, Feizollah A, Anuar NB. The rise of obfuscated Android malware and impacts on detection methods. PeerJ Comput Sci. 2022 Mar 9;8:e907. doi: 10.7717/peerj-cs.907. PMID: 35494876; PMCID: PMC9044361.

[7] F. Guyton, W. Li, L. Wang and A. Kumar, "Analysis of Feature Selection Techniques for Android Malware Detection," SoutheastCon 2022, Mobile, AL, USA, 2022, pp. 96-103, doi: 10.1109/SoutheastCon48659.2022.9764071.

[8] Islam et al (2023) Android malware classification using optimum feature selection and ensemble machine learning https://www.sciencedirect.com/science/article/pii/S2667345223000202

[9] Sun et al (2022) Android Malware Detection Based on Feature Selection and Weight Measurement

[10] Mahindru (2021) FSDroid:- A feature selection technique to detect malware from Android using Machine Learning Techniques https://link.springer.com/article/10.1007/s11042-020-10367-w

[11] "Nadia Daoudi, Kevin Allix, Tegawendé François Bissyandé, and Jacques Klein. 2022. A Deep Dive Inside DREBIN: An Explorative Analysis beyond Android Malware Detection Scores. ACM Trans. Priv. Secur. 25, 2, Article 13 (May 2022), 28 pages. https://doi.org/10.1145/3503463"

[12] Lin, Yan, Tianming Liu, W. Liu, Zhigaoyuan Wang, Li Li, Guoai Xu and Haoyu Wang. "Dataset Bias in Android Malware Detection." ArXiv abs/2205.15532 (2022): n. Pag.

[13] Wei F, Li Y, Roy S, et al. Deep Ground Truth Analysis of Current Android Malware[C]//International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, Cham, 2017: 252-276.

[14] Dissecting Android Malware: Characterization and Evolution Yajin Zhou, Xuxian Jiang Proceedings of the 33rd IEEE Symposium on Security and Privacy (Oakland 2012) San Francisco, CA, May 2012

[15] K. Allix, T. F. Bissyande, J. Klein, and Y. Le Traon. AndroZoo: Collecting Millions of Android Apps for the Research Community. Mining Software Repositories (MSR) 2016.

[16] Arash Habibi Lashkari, Andi Fitriah A. Kadir, Laya Taheri, and Ali A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification", In the proceedings of the 52nd IEEE International Carnahan Conference on Security Technology (ICCST), Montreal, Quebec, Canada, 2018.

[17] David Sean Keyes, Beiqi Li, Gurdip Kaur, Arash Habibi Lashkari, Francois Gagnon, Frederic Massicotte, "EntropLyzer: Android Malware Classification and Characterization Using Entropy Analysis of Dynamic Characteristics", Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS), IEEE, Canada, ON, McMaster University, 2021.

[18] Abir Rahali, Arash Habibi Lashkari, Gurdip Kaur, Laya Taheri, Francois Gagnon, and Frédéric Massicotte, "DIDroid: Android Malware Classification and Characterization Using Deep Image Learning", 10th International Conference on Communication and Network Security (ICCNS2020), Pages 70–82, Tokyo, Japan, November 2020.

[19] https://contagiodump.blogspot.com/

[20] Daniel Arp, Michael Spreitzenbarth, Malte Huebner, Hugo Gascon, and Konrad Rieck "Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket", 21th Annual Network and Distributed System Security Symposium (NDSS), February 2014.

[21] Michael Spreitzenbarth, Florian Echtler, Thomas Schreck, Felix C. Freling, Johannes Hoffmann, "MobileSandbox: Looking Deeper into Android Applications", 28th International ACM Symposium on Applied Computing (SAC), March 2013.

[22] Laya Taheri, Andi Fitriah Abdulkadir, Arash Habibi Lashkari; Extensible Android Malware Detection and Family Classification Using Network-

Flows and API-Calls, The IEEE (53rd) International Carnahan Conference on Security Technology, India, 2019
[23] Damshenas M, Dehghantanha A, Choo K K R, et al. M0droid: An android behavioral-based malware detection model[J]. Journal of Information Privacy and Security, 2015, 11(3): 141-157.
[24] RmvDroid
[25] Omar N. Elayan, Ahmad M. Mustafa, Android Malware Detection Using Deep Learning, Procedia Computer Science, Volume 184, 2021, Pages 847-852, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2021.03.106.
[26] R. Lukas and G. Kołaczek, "Android Malware Detection Using Deep Learning Methods," 2021 IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Bayonne, France, 2021, pp. 119-124, doi: 10.1109/WETICE53228.2021.00033.
[27] J. Wang, Q. Jing, J. Gao and X. Qiu, "SEdroid: A Robust Android Malware Detector using Selective Ensemble Learning," 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea (South), 2020, pp. 1-5, doi: 10.1109/WCNC45663.2020.9120537.
[28] D. -J. Wu, C. -H. Mao, T. -E. Wei, H. -M. Lee and K. -P. Wu, "DroidMat: Android Malware Detection through Manifest and API Calls Tracing," 2012 Seventh Asia Joint Conference on Information Security, Tokyo, Japan, 2012, pp. 62-69, doi: 10.1109/AsiaJCIS.2012.18.
[29] Z. Yuan, Y. Lu and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," in Tsinghua Science and Technology, vol. 21, no. 1, pp. 114-123, Feb. 2016, doi: 10.1109/TST.2016.7399288.
[30] T. Kim, B. Kang, M. Rho, S. Sezer and E. G. Im, "A Multimodal Deep Learning Method for Android Malware Detection Using Various Features," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 773-788, March 2019, doi: 10.1109/TIFS.2018.2866319.
[31] N. Peiravian and X. Zhu, "Machine Learning for Android Malware Detection Using Permission and API Calls," 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, 2013, pp. 300-305, doi: 10.1109/ICTAI.2013.53.
[32] H. Cai, N. Meng, B. Ryder and D. Yao, "DroidCat: Effective Android Malware Detection and Categorization via App-Level Profiling," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 6, pp. 1455-1470, June 2019, doi: 10.1109/TIFS.2018.2879302.
[33] S. Y. Yerima, S. Sezer, G. McWilliams and I. Muttik, "A New Android Malware Detection Approach Using Bayesian Classification," 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), Barcelona, Spain, 2013, pp. 121-128, doi: 10.1109/AINA.2013.88.
[34] P. R. K. Varma, K. P. Raj and K. V. S. Raju, "Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2017, pp. 294-299, doi: 10.1109/I-SMAC.2017.8058358.
[35] R. Vinayakumar, K. P. Soman and P. Poornachandran, "Deep android malware detection and classification," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 2017, pp. 1677-1683, doi: 10.1109/ICACCI.2017.8126084.
[36] Vasileios Syrris, Dimitris Geneiatakis, On machine learning effectiveness for malware detection in Android OS using static analysis data, Journal of Information Security and Applications, Volume 59, 2021, 102794, ISSN 2214-2126, https://doi.org/10.1016/j.jisa.2021.102794.(https://www.sciencedirect.com/science/article/pii/S2214212621000387)
[37] Alhebsi, Mohamed Salem, "Android Malware Detection using Machine Learning Techniques" (2022). Thesis. Rochester Institute of Technology.
[38] Hani AlOmari, Qussai M. Yaseen, Mohammed Azmi Al-Betar, A Comparative Analysis of Machine Learning Algorithms for Android Malware Detection, Procedia Computer Science, Volume 220, 2023, Pages 763-768, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2023.03.101.
[39] Alkahtani et al(2022)Artificial Intelligence Algorithms for Malware Detection in Android-Operated Mobile Devices
[40] YILMAZ et al.(2022) Classification of Malicious Android Applications Using Naive Bayes and Support Vector Machine Algorithms
[41] Sahin et al(2022). LinRegDroid: Detection of Android Malware Using Multiple Linear Regression Models-Based Classifiers
[42] Ab Razak et al. (2022) A Bayesian probability model for Android malware detection