A SURVEY ON ARTIFICIAL INTELLIGENCE TECHNIQUES FOR MALWARE DETECTION

Hend Faisal[1,2], Hanan Hindy[1], Samir Gaber[2,3], Abdel-Badeeh Salem[1]

[1]Faculty of Computer and Information Sciences, Ain Shams University, Egypt
[2]Egyptian Computer Emergency and Readiness Team (EG-CERT), National Telecom Regulatory Authority (NTRA)
[3]Faculty of Engineering in Helwan, Helwan University, Egypt

ABSTRACT

The rapid evolution of technology in the past years largely contributed to the digital transformation, however, attackers took advantage of it to spread malicious software (malware). Nowadays, malware has become more sophisticated, which makes it harder to be detected with traditional techniques. Over the years, attacks became, not only limited to computer-based operating systems, but also to that of mobile-based, which makes it even harder for analysts. Furthermore, this increases the need for more research in this direction. The technological evolution also gives researchers the chance to utilize Artificial Intelligence widely and leverage its capabilities in many fields in general and in the field of malware detection in particular. This paper provides a literature review on malware detection using Artificial Intelligence techniques and specifically, Machine Learning and Deep Learning techniques. The paper helps researchers to have a broad idea of the latest malware detection techniques, available datasets, challenges, and limitations.

KEYWORDS

Malware Detection, Artificial Intelligence, Machine Learning, Deep Learning, Android Malware

1. INTRODUCTION

Despite the significant improvement of technology and its positive impact on bringing ease to humans' lifestyle, and the rapid increase in the usage of internet in people's daily life, all of that evolution provides a great opportunity for malware authors to expand their work and expand spreading malware. This surge was specifically evident in last two years, which were considered record-breaking due to COVID-19 [1]. The pandemic prompted people to shift to remote work and increase the daily usage of internet coinciding lockdown. The International Telecommunication Union (ITU), 2021, statistics showed that at 2019, the beginning of the pandemic, about 54% of worlds' population used internet with about 4.1 billion people. In 2020, the number of internet users grew by 10.2% which is considered a leap according to the statistics reported in [2]. This number is still growing; nearly two-thirds of the world's population uses the internet [3].

In conjunction with this increase, Malicious Software (Malware) is considered a major threat worldwide, which continues to expand exponentially. According to an up-to-date study by AV-Test institute [4], about 450,000 malware instances are discovered daily. Originally, malware authors targeted computer-based users, especially Windows users, but through the evolution of technology, people become less dependent on computers and depend on other Operating Systems (OS); Android, iOS. This creates a new path for malware authors to spread their malware through other OS. Figure 1 illustrates how the number of Android users increase in comparison to other OS. Malware authors expand to target more than one platform, making it more difficult to limit malware spreading.

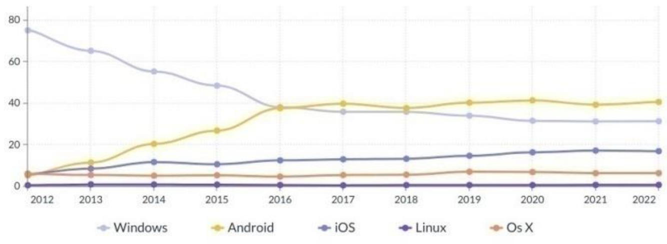

Figure 1: Operating System Market Share [2012-2022] [5]

Malware exist in different forms; Virus, Worm, Rootkit, Ransomware, and etc. Ransomware [6] is considered the hardest among them as it encrypts the victim's files and ask for ransom in exchange to decrypt them, which make attackers head on using it widely as a business. Furthermore, they target organizations, not only individuals [7].

Based on previous research in the field of malware detection, to analyze a malware sample there exists two methods; static analysis and dynamic analysis. Static analysis [8][9] depends on examining the suspicious sample without executing it. Traditionally, static analysis relied on Heuristic-based and Signature-based methods. Heuristic-based analysis encompasses a set of rules that are determined by experts, while signature-based depends on signatures, which are the unique identifier for a binary file. Both methods are effective and straightforward in detecting malware with a limited false positive ratio. However, these techniques fail in detecting any variations of this malware, beside failing in detecting unknown malware.

On the other hand, dynamic analysis [10] comprises running the suspicious sample in a safe environment which allows analysts to trace its behavior, for example; Application Programming Interface (API) calls, system calls or network traffic trying to find any suspicious activity.

Dynamic analysis is more effective, compared with static [11], in its ability to detect new and unknown malware. However, dynamic analysis is time and resource consuming, furthermore, the existence of evasive techniques is an obstacle in dynamic analysis where sample curtail its functionality when detecting that it run on an isolated environment [12] which makes it difficult in detecting advanced malware.

Artificial Intelligence (AI), with all its advancement, can provide powerful detection against advanced malware. When combining the modeling of malicious and benign behavior, it could make it easier to pick out the malicious ones. The goal of researchers is to build models that are capable of detecting different types of malware, including unknown and zero-day malware. In the literature, various malware detection techniques using AI are proposed, including Machine Learning (ML) [13][14][15] and Deep Learning (DL) [16][17][18].

This paper provides a review of the literature on malware detection using AI techniques; specifically, ML and DL techniques on both computer-based and mobile-based malware. In addition, this paper discusses the analysis methods, the latest datasets, limitations and challenges that face researchers.

This paper is organized as follows; Section 2 provides a summary of recent research in the field of malware detection using AI. It discusses malware detection pipeline including recent used datasets in both computer and mobile based, how analysis is done and its types, what kind of features could be useful in the detection phase and how feature selection impact detection accuracy. Section 3 describes the challenges and limitations facing malware analysts. Section 4 summarizes the concluding observations of this survey.

2. RELATED WORK

This section provides a summary of academic research that addresses how malware is detected using AI techniques.

2.1 Machine Learning

Over the past decade, ML algorithms have been used to detect and classify malware. Based on previous studies [14][19], Malware detection based on ML provides promising results as long as the suitable model and features are used. However, there exists some barriers as it requires large and labeled datasets to be able to predict malware with high accuracy, also dozens of malware are created daily [4], the computational cost that is used to periodically train and update the ML classifier is high. Furthermore, the wide variety of platforms make it difficult as each malware is implemented differently to target different platform. The following subsections summarize computer and android based related work that use ML to detect malware.

2.1.1 Computer-Based

In recent years, researchers have widely applied ML to detect computer-based malware. Table 1 lists computer-based related work that use ML in the detection of malware. Nicola *et al.* [20] proposed a malware taxonomic classification pipeline that was able to detect malicious Portable Executable (PE) files by extracting static features and used them for the classification of malware. In addition, the authors labeled those detected malware to their malware category. However, the authors faced some limitations as the mislabeling of malware samples in the used dataset, few samples were used to train classifiers and an overlap exists among different threat types and behavior. Al-Kasassbeh*et al.* [21] selected seven static features and used them as the input to the classifier and out of multiple classifiers used in their experiment, J48 was the most promising classifier. However, the highly unbalanced dataset that was used in their experiments leads to biased results. Sanjay *et al.* [22] studied the frequency of Opcode occurrence to detect unknown malware. They used multiple classifiers to compare between them in addition to multiple feature selection methods to compare between them too. Their results showed that Fisher Score (FS) performed better than other methods with multiple classifiers. Unlike previous methods, Rabadi*et al.* [23] used dynamic features in their experiment as they execute samples in an isolated virtual machine using Cuckoo Sandbox [24] to extract API-based features that were used in the detection phase. Nevertheless, the authors’ work targeted only Windows 7 and as a result of depending on Cuckoo Sandbox, their method is limited to Cuckoo’s hooked API calls only. Muhammad *et al.* in [25] used Gradient Classifier on both static and dynamic features separately and reported that the accuracy of the classifier based on static features was better than that of dynamic features. However, their study is limited due to the small size of the used dataset.
The detection of mobile-based malware vary from that of computer-based malware, the following subsection shows related work that use ML in the detection of Android-based malware.

Table 1: Summary of related work on Computer-based malware detection based on ML Techniques. Where: TPR: True Positive Rate, FPR: False Positive Rate, Acc: Accuracy

Year	Algorithms	Techniques	Features	Measures		Dataset		
				Metric	Value	Dataset	Count	Availability
2021 [20]	GBDT	Static	Raw PE	TPR FPR	86.3 % 0.1 %	Malicious Benign Unlabeled	400K 400k 300k	✓
2020 [21]	Ridor, RF, PART, J48, and IBk	Static	NumberOfSections, VirtualSize2, ResourceSize, ExportSize, IatRVA, ImageVersion, and Debug Size	TPR FPR	98.56 % 5.68 %	Malicious Benign	100,000 16,000	✓
2020 [23]	SVM, XGBoost, RF, DT, and PA	Dynamic	API calls	Acc.	98 %	Malicious Benign	7105 7774	✓
2019 [22]	RF, LMT, J48 Graft, and NBT	Static	Opcodes	Acc.	100 %	Malicious Benign	6010 4573	✓
2019 [25]	Gradient Classifier	Static Dynamic	Static: DOS, PE, Optional, and Sections Table Dynamic: API, Summary Information, DLLs, Registry keys changed	Acc. Acc.	99.36 % 94.64 %	Malicious Benign Malicious Benign	39,000 10,000 2200 800	Not Mentioned

2.1.2 Android-Based

The rising number of android malware nowadays raises the need for researchers to build models that are able to detect them. Table 2 summarizes mobile-based related work that use ML in the detection of malware. McDonald *et al.* [26] investigated the effectiveness of four different ML algorithms in conjunction with features selected from Android manifest file permissions to classify if the inputted file was malicious or benign. The findings showed that compared to all algorithms, Random Forest (RF) had the highest accuracy. Their paper is limited only to permissions, while neglecting some other static features that could have been extracted from manifest file too. Pandey *et al.* [27] proposed a methodology for detecting malicious android applications. Seven static features were extracted, then they applied two feature selection algorithms to remove irrelevant features. They worked on a balanced dataset and tried different classifiers, but RF was the one with the best accuracy reported. Moutaz*et al.* [28] depended on static features as well; the authors proposed three different grouping strategies to choose the most valuable API calls to maximize the ability to identify Android malware apps. In addition, their approach could determine the similarities among malware families. Sangal*et al.* [29] extracted android permissions and intents features to be then used as an input for the ML algorithm. Their study showed that RF was the best classifier. However, they used an unbalanced dataset. The benefit of Myar*et al.* [30] approach was that the system is platform independent so that different versions of Android OS can benefit from it. They combined features from static and dynamic analysis. However, the time complexity was high in real smartphone because of hardware requirements, which makes real-time detection is not applicable with their proposed approach.

As demonstrated, permission-based features are the most commonly used feature in the classification of an Android malware, followed by API calls features.

Table 2: Summary of related work on Android-based malware detection based on ML techniques. Where: Acc: Accuracy

Year	Algorithms	Techniques	Features	Measures		Dataset		
				Metric	Value	Dataset	Count	Availability
2021 [26]	RF, SVM, Gaussian NB, and K-Means	Static	permission-based	F1	75.00 % - 84.00 %	Malicious Benign	6000 4597	✓
2021 [27]	RF, KNN, XGBoost, and DT	Static	permission-based, providers, activities, receivers, services, and opcodes	Acc.	92.50 % - 99.00 %	Malicious Benign	15,000 15,000	✓
2020 [28]	RF, KNN, RT, NB, and J48	Static	permission-based and API calls	Acc.	87.90 % - 94.30 %	Malicious Benign	13,719 14,172	✓
2020 [29]	RF, SVM, KNN, NB, and DT	Static	permission-based and intents	Acc.	88.23 % - 96.05 %	Malicious Benign	396 1126	✓
2019 [30]	RF, NB and, KNN	Hybrid	permission-based, API calls and Providers	Acc.	72.00 % - 89.00 %	Malicious Benign	113 219	✓

2.2 Deep Learning

When the amount of training data is large, DL, which is a subset from ML, is expected to perform better. Different DL algorithms such as Recurrent Neural Networks (RNN) [31], Convolutional Neural Networks (CNN) [32], Multilayer Perceptron (MLP) [33], Deep Belief Network (DBN) [34], and Long Short Term Memory Network (LSTM) [35] have been applied in the process of malware detection [36]. The following subsections summarizes computer and mobile based related work that use DL to detect malware.

2.2.1 Computer-Based

Table 3 summarizes computer-based related work that use DL models to detect malware. Azeez*et al.* [33] combined DL models with ML models. The initial stage classification was done by a stacked ensemble of fully-connected MLPs and one-dimensional CNN and the second stage classification was done by ML models. It was reported that RF achieved the best accuracy. The proposed framework is limited to supervised learning, and that requires that samples must be identified and labeled by experts. This raised the need for developing unsupervised ensemble learning frameworks for malware recognition. In addition, the authors used an unbalanced dataset, which led to biased results. In [37], Zhong*et al.* proposed a Multi-Level Deep Learning System (MLDLS) for malware detection. First, Ngram analysis was applied on both static and dynamic features so that candidate n-gram features were extracted from both. Additionally, the dataset was partitioned into clusters using K-Means clustering algorithm. CNNs, Deep RNN, and Deep Fully Connected Feed Forward networks (FC) models were used in the training phase and the one with the best results was selected to classify the input sample.

Static and Dynamic features only provide information in text formats. However, researchers started to embrace the fact that image-based features could improve the malware detection research moving forward. As a result, they started looking for vision-based approaches to use due to their advancement in the image field.

Table 3: Summary of related work on Computer-based malware detection based on DL techniques. Where: Acc: Accuracy, TPR: True Positive Rate, AUC: Area Under Curve

Year	Algorithms	Techniques	Features	Measures		Dataset		
				Metric	Value	Dataset	Count	Availability
2021 [38]	Dense Net	Vision based	Gray-scale image	Acc. Malimg BIG 2015 MaleVisMalicia	 98.23 % 98.46 % 98.21 % 89.48 %	Malicious(Malimg, Big2015, MaleVis, Malicia) Benign	9339, 21741, 14226, 9670 1043	✓
2021 [33]	Stage 1: MLP, CNNs Stage 2: NB, DT, RF, Gradient boosting, and AdaBoosting	Static	Number Of Sections, Major Linker Version, Address Of Entry Point, etc..	Acc. F1	100 % 100 %	Malicious Benign	14955 5012	✓
2020 [39]	Stage 1: CNNs Stage 2: Deep Forest	Vision based	Gray-scale images	Acc.: Malimg BIG 2015 MaleVis.	 98.65 % 97.2 % 97.43 %	Malicious(Malimg, Big2015, MaleVis Benign	9339, 21741 14226 1044	✓
2019 [37]	CNN, RNN, FCs	Hybrid	N-gram	Static:TPR Static:AUC Dynamic:TPR Dynamic:AUC	72-92 % 68-85 % 68-89 % 69-90 %	Malicious Benign	2,242,234 3,425,176	✗

Vision-based Approach In recent years, researchers started to employ visual-based approaches in the detection of malware by converting malware source code into malware binary and then convert binaries into either RGB images or Gray-scale images. These images were then classified into their relative family. It has been observed that the images of malware that belongs to the same family are quite similar in structure and texture [38]. Therefore, malware could be converted into images and then DL algorithms are used. Hemalatha*et al.* [38] proposed a visualization-based method, where malware binaries were depicted as two-dimensional images and then classified by DenseNet model. The study was evaluated on four different datasets;Malimg [40], Big2015 [41], MaleVis [42] and Malicia [43]. The model achieved high accuracy, however the proposed method came with high false negative rate. In [39], images were processed into two phases, sliding window phase and cascade layering phase. Sliding window phase preserved the spatial relationship between raw pixels, where each input image was scanned with parallel processing of two sliding windows. While cascade layering phase consisted of sequential layers, with each layer consisting of four ensemble forests so that the final prediction was obtained and the class with the highest probability was the matching class for the input image. Having discussed computer-based malware detection using DL techniques, in the following subsection authors proposed different DL models to detect mobile-based malware.

2.2.2 Android-Based

Table 4 summarizes Android-based related work that use DL models to detect malware. Pei *et al.*[44] proposed AMalNet model that extracted static features like APIs, permissions, and components then stored them in a database. Natural Language Processing (NLP) features were then extracted and mapped into vectors to be the input for the proposed hybrid DL technique. The authors combined Graph Convolutional Networks (GCN) and Independently Recurrent Neural Network (IndRNN) in order to take full account of the semantic distribution information of malware. Their proposed method is limited to static features, so if the sample is obfuscated, this method would not work as expected. On the other hand, Alzaylaee*et al.* [45] suggested DL-Droid, a DL-based dynamic analysis system for Android malware detection. DL-Droid utilized a state-based input generation approach, Stateful vs. Stateless input generation, and experiments were executed four times; first based on stateful dynamic features, second based on stateless dynamic features, third based on stateful hybrid features and last was based on stateless hybrid features. The third one was the one with the best reported performance.

Lu *et al.* in [46] proposed an android malware detection algorithm which combined Deep Belief Network (DBN) and Gate Recurrent Unit (GRU). The algorithm extracted both static features and dynamic features. The DBN was used to process the static features, while GRU was used for the dynamic ones. A total of 351 features were extracted, including 303 static features and 48 dynamic features. The features from both models are fed into the Back Propagation (BP) neural network. The authors stated that the results were within an acceptable range. However, they mentioned that their method needed to be optimized to reduce the time complexity. A two-layer method was proposed in [47] for the detection of malicious android applications. The first layer combined static features like permissions, intent, and component information based with fully connected neural network. However, in the second layer a new method cascaded CNN and AutoEncoder which was used to detect malware through network traffic features. Unlike previous methods, Feng *et al.* in [48] used a pre-installed solution. By leveraging customized Deep Neural Networks (DNN) and binary features, sample were classified as malicious or benign. Their proposed approach combined features extracted from binary code and behavioral features. The proposed model was tested on six different real mobile devices with high accuracy, but the difference was in the prediction time. Due to limited training dataset, their proposed model failed in the detection of new malware families.

Table 4: Summary of related work on Android-based malware detection based on DL techniques. Where: Acc: Accuracy

Year	Algorithms	Techniques	Features	Measures		Dataset		
				Metric	Value	Dataset	Count	Availability
2020 [46]	DBN, GRU and BP Neural Network	Hybrid	Static: resource features and semantic features Dynamic: behavioral features	Acc.	96.82 %	Malicious Benign	6298 7000	✓
2020 [47]	First layer: Neural Network Second layer:CNN and AutoEncoder	Hybrid	First layer: permission, intent and component information based Second layer: Network traffic features	Acc.: First layer Second layer	95.22 % 99.3 %	Malicious Benign	4354 5065	✓
2020 [48]	RNN	Hybrid	Behavioral-based features & Binary code	Acc.	96.75 %	Malicious Benign	45284 29010	✓
2020 [44]	GCN, IndRNN	Static	permissions, components and APIs	Acc. F1	99.69 % 99.7 %	Malicious(DREBIN, AMD, lab-built, AndroZoo and Praguard) Benign	5560, 24553, 4664, 50000 and 1497 50000	✓
2020 [45]	First phase: MLP Second phase: SVM, PART, NB, RF, and J48	Dynamic(D Hybrid(H)	) API calls, action, and events	Acc.: D (stateful) D (stateless) H (stateful) H (stateless)	95.21 % 94.95 % 98.5 % 94.95 %	Malicious Benign	11505 19620	✓

Other than conventional ML and DL techniques, there are other algorithms that are adopted from other domains that are used to detect malware, NLP was considered to be one of them. As shown in Table 5, Mimura *et al.* [49] extracted all printable strings from dataset of malicious and benign samples and split strings into word then used Doc2vec or LSI, which are common NLP methods used in representing input as vectors. Hence, a classifier was trained with labeled feature vectors. The accuracy ratio of their proposed method was high, however, they stated that their method might not be applicable to sophisticated packed samples. In addition, Dovom*et al.* in [50] employed fuzzy and fast fuzzy pattern trees for edge malware detection by using the opcodes of Internet of Things (IoT) applications. The authors mentioned that their proposed model performed better than other ML classifiers. On the other hand, Amer*et al.* [51] depended on dynamic analysis. They used a technique in NLP called Word2Vec in their initial phase, which was used to produce word vectors from large corpus of text. The input used in their proposed method was a sequence of API calls. Next, they computed the similarity between API calls and for the final step they used K-means to cluster the similarity matrix to either benign or malware. The authors relied on pre-processed API call sequence datasets and did not mention any evaluation on real-life samples testing. Pre-processed API samples could lead to misleading interpretations of the proposed model.

Table 5: Summary of related work on malware detection based on AI Techniques from other domains. Where: Acc: Accuracy, DS: Dataset

Year	Algorithms	Techniques	Features	Measures	Dataset		
				MetricValue	Dataset	Count	Availability

2021 [49]	NLP, SVM, CNN, MLP. XGB and RF	Static	Strings	98.8 % - Acc. 99.7 %	Malicious(DS1)
					Benign(DS1)
					Malicious(DS2)
						287,375
						250,000
						28,488
							✓

2020 [51]	NLP	Dynamic	API call sequence n-gram	F1 99 % Acc.99.7 %	Malicious Benign	30,658 21,422	✓
2019 [50]	Fuzzy Pattern Tree	Static	OpCodes	86 % - Acc. 100 %	Malicious & Benign	33,363	✓

Following the discussion about malware detection techniques in the literature, the following section outlines the most recent datasets used in malware research. Moreover, it discusses the different features' categorization which influence building any AI-based model for malware analysis.

2.3 Malware Detection Contextual Map

Figure 2 summarizes the pipeline that is used in the detection of malware using AI, whether it is Computer-based or Mobile-based. Each phase in the pipeline affects the detection accuracy, starting from the samples collected, and the importance of using a balanced dataset. Passing by analysis approaches and features extraction which differs according to the platform as each platform consists of its own features and ending by feature selection and classification where the sample is classified as malware or benign one. Since that most targeted platform are Windows-based and Android-based, this paper focuses on them.

The following subsections discuss each phase in the contextual map in detail.

2.3.1 Datasets

ML and DL both require large datasets for training before being used to detect malware [52]. However, finding a ready to use an up-to-date dataset is a challenging task that faces most of the researchers. Table 6 summarizes datasets that were most used by researchers in recent years.

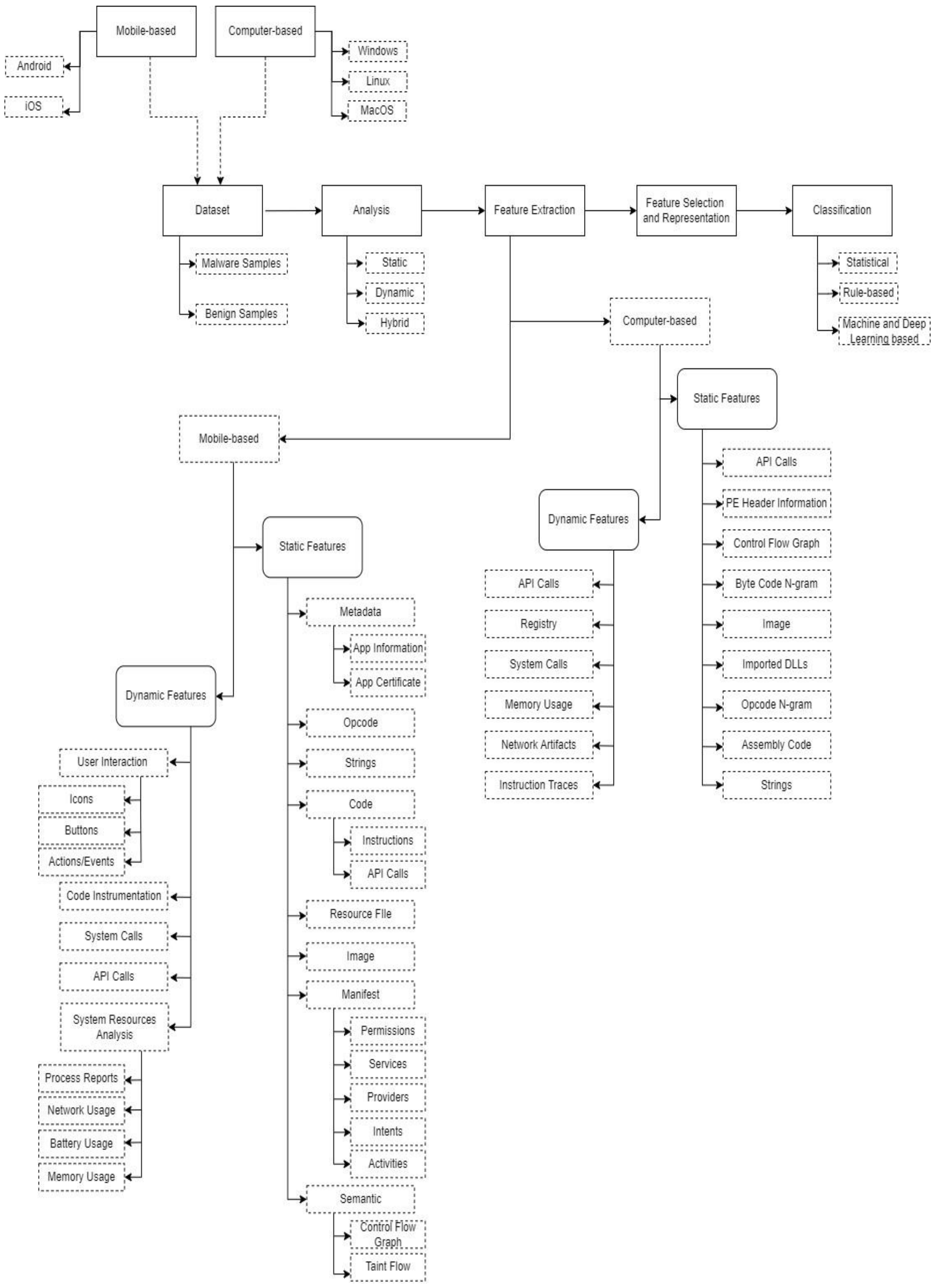

Figure 2: Malware Detection Contextual Map

Table 6: List of Prominent Malware Datasets

Name	Year	Platform	Description
CCCS-CIC-AndMal-2020 [53]	2020	Android-based	The dataset includes 200,000 malware and 200,000 benign samples.
MaleVis [42]	2019	Computer-based	An open-set image malware dataset. It contains 9100 training and 5126 validation RGB images.
CIC-InvesAndMal2019 [54]	2019	Android-based	The dataset consists of static features of 1522 applications which consists of 396 malware and 1126 benign application.
Benign & Malicious PE Files Kaggle [55]	2018	Computer-based	The dataset is a collection of malicious and benign data from PE Files.
EMBER [56]	2017	Computer-based	The dataset is a collection of features calculated from 1.1 million PE file samples (400K malware, 400K benign, 300K unlabeled).
Malimg [40]	2017	Computer-based	The dataset contains 9,339 malware images from 25 families.
CICAndMal2017 [57]	2017	Android-based	The dataset includes 10,854 samples (4,354 malware and 6,500 benign).
AMD [58]	2017	Android-based	The dataset contains 24,553 malware samples ranging from 2010 to 2016.
AndroZoo [57]	2016	Android-based	The dataset contains 18,215,449 different APKs.
BIG 2015 [41]	2015	Computer-based	The dataset contains a set of known malware files representing a mix of 9 different families.
APIMDS [59]	2015	Computer-based	The dataset contains a full list of malicious API sequences, hash information.
PRAGuard [60]	2014	Android-based	The dataset contains 10,479 malicious samples, obtained by obfuscating the MalGenome and theContagioMinidump datasets with seven different obfuscation techniques.
DREBIN [61]	2014	Android-based	The dataset contains 5560 malware and 123453 benign samples.
FFRI [62]	From 2013	Computer-based	The dataset was generated by Cuckoo Sandbox and FFRI yarai analyzer Professional, It contains dynamic logs.
Malicia [43]	2013	Computer-based	The dataset contains 11,688 malware binaries collected over a period of 11 months.

2.3.2 Analysis

To understand the behavior of an input file, it must be analyzed. There are two techniques that help in the process of analyzing samples [52]; Static analysis and Dynamic analysis. Static analysis is the process of analyzing a sample without running it. Dynamic analysis, unlike static, means that a sample has to be running to be able to understand its behavior and functionality

[63]. Analysts use hybrid analysis techniques; which are a combination of both static and dynamic analysis. Analysis plays an important role in extracting features from the files so that these features can then be used as the input for the classifier that labels samples into either malware or benign.

2.3.3 Feature Extraction

This process is where features are extracted from samples. They provide an abstraction view of the file so that it can be later classified into either malicious or benign. This process plays a vital role and highly influence the accuracy and efficiency of any model [52]. Features can be classified into either static or dynamic features, which differ based on the analysis technique used as explained in the previous section. It is important to mention that features extracted from computer-based files vary from that of those of mobile-based as explained below.

Windows-based Features A taxonomy of features is provided in Figure 2; Windows based features could be derived by extracting (a) Windows API calls. API calls are used to describe the behavior of the executable. API call could be extracted statically or dynamically. In static, API calls could be extracted from source code, while in dynamic, API call sequences are extracted while running the sample. Liu *et al.* [64] extracted API call sequence after running samples on Cuckoo sandbox. The authors conducted an experiment and their accuracy reached 97.85%. (b) Control Flow Graph (CFG) features could be extracted from static analysis. These covers all possible paths during execution of a sample. Behera *et al.* in [65] used CFG to detect obfuscated programs by getting a basic executable before and after obfuscation, then decompiling both executables to an assembly code. Additionally, the authors made a basic block of codes and constructed CFGs from those basic block of codes. Finally, the authors compared both CFGs using a graph matching algorithm and if the CFG of the original executable is found to be isomorphic to that of obfuscated CFG, then the executable is classified as obfuscated.

Recently, researchers turned into using (c) Images; binaries are transformed into grayscale/ RGB images. Kancherla*et al.* [66] extracted different image-based features like Intensity based, Wavelet based and Gabor based features to be the input for ML models to classify samples as malware or benign relying on their image. The accuracy of used method reached 95.95%. (d) Strings also could be an indicator for a malware sample. Ito *et al.* in [67] used ASCII strings and converted them into words, then used NLP techniques to convert the words to a feature vector. On the other hand, (e) system calls traces, which are the way for programs to interact with the kernel of the OS, could be used to detect malware. Kim. in [68] extracted system calls traces. The author used API tracer to collect system call traces and implemented N-grams, where the features are sequences of system calls instead of a single system call, then applied ML technique on preprocessed traces. The accuracy of the proposed method reaches 96%. (f) Registry, where low-level settings for the Microsoft Windows OS and for applications are stored, was also considered by researchers as a feature that could help in the detection of malware. Tajoddin*et al.* [69] extracted registry accesses by running samples in Cuckoo sandbox and searched for anomalous registry accesses.

Android-based Features Android-based features described in Figure 2 differ from that of windows-based. Features are extracted from the corresponding application package (APK) where each APK consists of Meta-Inf folder, assets, Manifest file, classes.dex, lib, and resources. Manifest file describes information about the app. (a) Permissions, which are one of the most used features in the field of android malware detection as it identifies the privileges that any android app need. (b) intents, defined as “A messaging object you can use to request an action from another app component.” [70] and more features like (c) services, (d) providers and (e) activities are stored in that manifest file. Khariwal*et al.* [71] chose permissions and intents features in their experiments. Then, the authors extended their features by using NLP word embedding technique to create Bag-of-Words and applied three different ML algorithms. (f) Network traffic could also be an indicator for any suspicious activity on the mobile device. Wang *et al.* [72] combined network traffic analysis with ML in their method. The authors specifically used HTTP requests and TCP Flows as their feature set, and at last they applied ML to classify whether mobile application is malicious or not. Mobile device statistics like (g) memory usage, (h) battery usage, (i) process reports, and network usage could also be an indicator for suspicious behavior. Authors in [73] extracted previous features using Mal-warehouse Information Extraction Tool (MIET), and by using ML algorithms on the exported features, the sample is categorized into rather malicious or benign.

There are common features between Windows-based and Android-based, but represented in different forms. API calls are considered one of them. Peiravian*et al.* [74] proved that by combining API calls and permissions, accuracy can be improved. The authors compared thier results of combining both features and each feature separately, they found that by combining both features the accuracy raised to more than 2%. Images also could be used in the detection of Android malware as it is used in that of Windows-based; Ding *et al.* [75] extracted bytecode file from Android APK file, and converted that bytecode stream into a 2D bytecode image to be the input to CNN model to classify them. Zhao *et al.* [76] made a Android malware detection system based on CNN using Opcode sequences. The Opcode features are derived from Dalvik instruction, which is a set that contains operational information about the app.

2.3.4 Feature Selection

To improve model performance and reduce computational cost, not all extracted features are used for classification. Feature selection is the process that reduces the number of input variables/features before being used in the malware prediction classifier.Babaagba*et al.* [77] highlighted the importance of feature selection. The authors experimented different ML algorithms without using any feature selection method, and the accuracy of the algorithms varied from 68.45% to 77.18%. However, after employing Information gain algorithms, the results were improved according to more than one algorithm.

3. CHALLENGES AND LIMITATIONS

With the wide development of malware, detecting it become a challenge to researchers. Based on the previous discussion and mentioned limitations in section 2, the most common obstacles encountered are listed below.

- **Anti-analysis Techniques:** In order to dynamically analyze a sample, it must be run on an isolated environment, for example Virtual Machines (VM) or sandboxes [23]. As a result, Malware developers began to use methods to evade from being run on a VM, sandbox, by detecting whether there exists analysis tools on the system and in addition check if the system is on debug mode or not. In case of existence of the previously mentioned methods, malware avoid running correctly on the system, thus bypass from being analyzed and detected.
- **Obfuscation and Packing:** Another technique malware authors uses to evade from basic static analysis, is to either pack their malware or obfuscate it. Obfuscation [78] is a technique of making a piece of code unreadable while packing is considered a subset of obfuscation, where malware is compressed or encrypted.
- **Dataset:** Finding a dataset is an essential step for building any ML/DL model, which is considered a challenge for malware researchers. Not only finding any dataset is the solution

for the problem also the dataset must be up-to-date, large, well labeled and updated periodically so that a model could be used in the detection of malware.

- **Evolution of malware:** There exists sophisticated malware and undetected malware. To evade from signature-based detection, malware authors develop their malware to bypass basic detection techniques. Sen. *et al.* citesen explained the idea about malware developers who used Genetic Programming (GP) in order to create new variants of malware automatically.
- **AI Obstacles:** AI, as mentioned before, is an effective field for malware detection but as any technology it has some limitations. It requires high computing resources, time and labeled dataset. Furthermore, any model is expected to have a percentage of false positives and the challenge is to reach a tolerable ratio of false positives and true positives.

4. CONCLUSION

The struggle between malware analysts and malware threat actors is a never-ending battle. Therefore, there is a constant need to find new robust ways to detect malware. Furthermore, AI is used widely in many research fields, including Malware detection. This paper presented a literature review of malware detection using different AI approaches, mainly ML and DL. Reviewed papers were categorized into Computer-based and Android-based platforms due to the astonishing rapid evolution in malware. Papers are compared according to the used approaches, classification algorithms, datasets, and techniques. Furthermore, the paper shows how feature extraction and selection processes affect the detection model accuracy and how ML and DL could be effective in detecting malware. Furthermore, the paper highlights and discusses the different challenges and limitations that face malware detection research. Even though different approaches have been proposed to help in detecting malware, none of them is said to detect all never-ending evolution of malware. As a future work, new approaches need to be proposed to increase the detection rate and face sophisticated malware.

REFERENCES

[1] N. A. Khan, S. N. Brohi, and N. Zaman, “Ten deadly cyber security threats amid COVID-19 pandemic,” 2020. TechRxiv, doi: 10.36227/techrxiv.12278792.v1.

[2] Facts and figures, “Internet use,” 2021. Available: https://www.itu.int/itud/reports/statistics/2021/11/15/internet-use/ Accessed: (10 June 2022).

[3] DataReportal, “Digital around the world - datareportal – global digital insights,” 2022. Available: https://datareportal.com/global-digital-overview Accessed: (10 June 2022).

[4] AV-TEST, “Test: Antivirus & security software & antimalware reviews,” 2022. Available: https://www.av-test.org/ Accessed: (10 June 2022).

[5] S. G. Stats, “Operating system market share worldwide,” 2022. Available: https://gs.statcounter.com/os-market-share Accessed: (10 June 2022).

[6] P. O’Kane, S. Sezer, and D. Carlin, “Evolution of ransomware,” Iet Networks, vol. 7, no. 5, pp. 321–327, 2018.

[7] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, “Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions,” Computers & Security, vol. 74, pp. 144–166, 2018.

[8] A. Shalaginov, S. Banin, A. Dehghantanha, and K. Franke, “Machine learning aided static malware analysis: A survey and tutorial,” in Cyber threat intelligence, pp. 7–45, Springer, 2018.

[9] A. Fed´ak and J. Stulrajter, “Fundamentals of static malware analysis: Principles,ˇ methods and tools,” Science & Military Journal, vol. 15, no. 1, pp. 45–53, 2020.

[10] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, “Dynamic malware analysis in the modern era—a state of the art survey,” ACM Computing Surveys (CSUR), vol. 52, no. 5, pp. 1–48, 2019.

[11] D. Gibert, C. Mateu, and J. Planes, “The rise of machine learning for detection and classification of malware: Research developments, trends and challenges,” Journal of Network and Computer Applications, vol. 153, p. 102526, 2020.

[12] A. Afianian, S. Niksefat, B. Sadeghiyan, and D. Baptiste, “Malware dynamic analysis evasion techniques: A survey,” ACM Comput. Surv., vol. 52, nov 2019.

[13] V. Kouliaridis and G. Kambourakis, “A comprehensive survey on machine learning techniques for android malware detection,” Information, vol. 12, no. 5, p. 185, 2021.

[14] S. Saad, W. Briguglio, and H. Elmiligi, “The curious case of machine learning in malware detection,” arXiv preprint arXiv:1905.07573, 2019.

[15] A. Mahindruand A. Sangal, “Mldroid—framework for Android malware detection using machine learning techniques,” Neural Computing and Applications, vol. 33, no. 10, pp. 5183–5240, 2021.

[16] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, “Malware detection using machine learning and deep learning,” in International Conference on Big Data Analytics, pp. 402–411, Springer, 2018.

[17] A. Naway and Y. Li, “A review on the use of deep learning in android malware detection,” arXiv preprint arXiv:1812.10360, 2018.

[18] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, “A multimodal deep learning method for Android malware detection using various features,” IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 773–788, 2018.

[19] J. Singh and J. Singh, “A survey on machine learning-based malware detection in executable files,” Journal of Systems Architecture, vol. 112, p. 101861, 2021.

[20] N. Loi, C. Borile, and D. Ucci, “Towards an automated pipeline for detecting and classifying malware through machine learning,” CoRR, vol. abs/2106.05625, 2021.

[21] M. Al-Kasassbeh, S. Mohammed, M. Alauthman, and A. Almomani, Feature Selection Using a Machine Learning to Classify a Malware, pp. 889–904. Cham: Springer International Publishing, 2020.

[22] S. Sharma, C. Rama Krishna, and S. K. Sahay, “Detection of advanced malware by machine learning techniques,” in Soft Computing: Theories and Applications (K. Ray, T. K. Sharma, S. Rawat, R. K. Saini, and A. Bandyopadhyay, eds.), (Singapore), pp. 333–342, Springer Singapore, 2019.

[23] D. Rabadi and S. G. Teo, “Advanced windows methods on malware detection and classification,” in Annual Computer Security Applications Conference, ACSAC ’20, (New York, NY, USA), p. 54–68, Association for Computing Machinery, 2020.

[24] Cuckoo Sandbox - Automated Malware Analysis, “Automated malware analysis,” 2017. Available: https://cuckoosandbox.org/ Accessed: (10 June 2022).

[25] M. Ijaz, M. H. Durad, and M. Ismail, “Static and dynamic malware analysis using machine learning,” in 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST), pp. 687–691, 2019.

[26] J. Mcdonald, N. Herron, W. Glisson, and R. Benton, “Machine learning-based Android malware detection using manifest permissions,” 01 2021.

[27] S. Pandey, C. Rama Krishna, A. Sharma, and S. Sharma, “Detection of Android malware using machine learning techniques,” in Innovations in Computer Science and Engineering (H. S. Saini, R. Sayal, A. Govardhan, and R. Buyya, eds.), (Singapore), pp. 663–675, Springer Singapore, 2021.

[28] M. Alazab, M. Alazab, A. Shalaginov, A. Mesleh, and A. Awajan, “Intelligent mobile malware detection using permission requests and api calls,” Future Generation Computer Systems, vol. 107, pp. 509–521, 2020.

[29] A. Sangal and H. Verma, “A static feature selection-based Android malware detection using machine learning techniques,” 10 2020.

[30] S. Myat, “Feature extraction using hybrid analysis for Android malware detection framework,” International Journal of Engineering Research and, vol. V8, 07 2019.

[31] R. Vinayakumar, K. Soman, and P. Poornachandran, “Evaluation of recurrent neural network and its variants for intrusion detection system (ids),” International Journal of Information System Modeling and Design (IJISMD), vol. 8, no. 3, pp. 43–63, 2017.

[32] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” Insights into imaging, vol. 9, no. 4, pp. 611–629, 2018.

[33] N. A. Azeez, O. E. Odufuwa, S. Misra, J.Oluranti, and R. Damaˇseviˇcius, “Windows pe malware detection using ensemble learning,” Informatics, vol. 8, no. 1, 2021.

[34] I. Sohn, “Deep belief network based intrusion detection techniques: A survey,” Expert Systems with Applications, vol. 167, p. 114170, 2021.

[35] G. Van Houdt, C. Mosquera, and G. N´apoles, “A review on the long short-term memory model,” Artificial Intelligence Review, vol. 53, no. 8, pp. 5929–5955, 2020.

[36] B. Yadav and S. Tokekar, "Recent innovations and comparison of deep learning techniques in malware classification: a review," International Journal of Information Security Science, vol. 9, no. 4, pp. 230–247, 2021.

[37] W. Zhong and F. Gu, "A multi-level deep learning system for malware detection," Expert Systems with Applications, vol. 133, pp. 151–162, 2019.

[38] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaˇseviˇcius, "An efficient densenet-based deep learning model for malware detection," Entropy, vol. 23, no. 3, 2021.

[39] S. A. Roseline, S. Geetha, S. Kadry, and Y. Nam, "Intelligent vision-based malware detection and classification using deep random forest paradigm," IEEE Access, vol. 8, pp. 206303–206324, 2020.

[40] Vision Research Lab, "Malware classification on malimg dataset," 2017. Available: https://paperswithcode.com/sota/malware-classification-on-malimg-dataset Accessed: (10 June 2022).

[41] Kaggle, "Microsoft malware classification challenge (big 2015)," 2015. Available: https://www.kaggle.com/c/malware-classification Accessed: (10 June 2022).

[42] A. Bozkir, A. Cankaya, and M. Aydos, "Utilization and comparision of convolutional neural networks in malware recognition," 03 2019.

[43] A. Nappa, M. Z. Rafique, and J. Caballero, "The malicia dataset: identification and analysis of drive-by download operations," International Journal of Information Security, vol. 14, 02 2014.

[44] X. Pei, L. Yu, and S. Tian, "Amalnet: A deep learning framework based on graph convolutional networks for malware detection," Computers & Security, vol. 93, p. 101792, 03 2020.

[45] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Dl-droid: Deep learning based Android malware detection using real devices," Computers & Security, vol. 89, p. 101663, 2020.

[46] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," Security and Communication Networks, vol. 2020, p. 8863617, 8 2020.

[47] J. Feng, L. Shen, Z. Chen, Y. Wang, and H. Li, "A two-layer deep learning method for Android malware detection using network traffic," IEEE Access, vol. 8, pp. 125786– 125796, 2020.

[48] R. Feng, S. Chen, X. Xie, G. Meng, S.-W. Lin, and Y. Liu, "A performance-sensitive malware detection system using deep learning on mobile devices," IEEE Transactions on Information Forensics and Security, vol. 16, pp. 1563–1578, 2021.

[49] M. Mimura and R. Ito, "Applying nlp techniques to malware detection in a practical environment," International Journal of Information Security, 6 2021.

[50] E. Dovom, A. Azmoodeh, A. Dehghantanha, D. Newton, R. Parizi, and H. Karimipour, "Fuzzy pattern tree for edge malware detection and categorization in iot," Journal of Systems Architecture, vol. 97, 03 2019.

[51] E. Amer and I. Zelinka, "A dynamic windows malware detection and prediction method based on contextual understanding of api call sequence," Computers & Security, vol. 92, p. 101760, 2020.

[52] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," Computers & Security, vol. 81, pp. 123–147, 2019.

[53] A. Rahali, A. H. Lashkari, G. Kaur, L. Taheri, F. GAGNON, and F. Massicotte, "Didroid: Android malware classification and characterization using deep image learning," in 2020 the 10th International Conference on Communication and Network Security, ICCNS 2020, (New York, NY, USA), p. 70–82, Association for Computing Machinery, 2020.

[54] L. Taheri, A. F. A. Kadir, and A. H. Lashkari, "Extensible Android malware detection and family classification using network-flows and api-calls," in 2019 International Carnahan Conference on Security Technology (ICCST), pp. 1–8, 2019.

[55] Kaggle, "Benign & malicious PE files," 2018. Available: https://www.kaggle.com/amauricio/pe-files-malwares Accessed: (10 June 2022).

[56] H. S. Anderson and P. Roth, "Ember: An open dataset for training static pe malware machine learning models," 2018.

[57] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark Android malware datasets and classification," in 2018 International Carnahan Conference on Security Technology (ICCST), pp. 1–7, 2018.

[58] Y. Li, J. Jang, X. Hu, and X. Ou, "Android malware clustering through malicious payload mining," 2017. Available: https://arxiv.org/abs/1707.04795.

[59] Y. Ki, E. Kim, and H. K. Kim, "A novel approach to detect malware based on api call sequence analysis," International Journal of Distributed Sensor Networks, vol. 11, p. 659101, 6 2015.

[60]D.Maiorca, D. Ariu, I. Corona, M. Aresu, and G. Giacinto, “Stealth attacks: An extended insight into the obfuscation effects on Android malware,” Computers & Security, vol. 51, pp. 16–31, 2015.
[61] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, “Drebin effective and explainable detection of Android malware in your pocket,” in Ndss, vol. 14, pp. 23–26, 2014.
[62] FFRI, “FFRI dataset 2017,” 2013. Available: https://www.iwsec.org/mws/2017/20170606/FFRI Dataset 2017.pdf Accessed: (10 June 2022).
[63] E. Gandotra, D. Bansal, and S. Sofat, “Malware analysis and classification: A survey,” Journal of Information Security, vol. 05No.02, p. 9, 2014.
[64] Y. Liu and Y. Wang, “A robust malware detection system using deep learning on api calls,” in 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), pp. 1456–1460, 2019.
[65] C. K. Behera, G. Sanjog, and D. LalithaBhaskari, “Control flow graph matching for detecting obfuscated programs,” in Software Engineering (M. N. Hoda, N. Chauhan, S. M. K. Quadri, and P. R. Srivastava, eds.), (Singapore), pp. 267–275, Springer Singapore, 2019.
[66] K. Kancherla and S. Mukkamala, “Image visualization based malware detection,” in 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), pp. 40–44, 2013.
[67] R. Ito and M. Mimura, “Detecting unknown malware from ascii strings with natural language processing techniques,” in 2019 14th Asia Joint Conference on Information Security (AsiaJCIS), pp. 1–8, 2019.
[68] C. W. Kim, “Ntmaldetect: A machine learning approach to malware detection using native API system calls,” CoRR, vol. abs/1802.05412, 2018.
[69] A. Tajoddin and M. Abadi, “Ramd: registry-based anomaly malware detection using one-class ensemble classifiers,” Applied Intelligence, vol. 49, pp. 2641–2658, 7 2019.
[70] Android Developers, “Intents and intent filters.” Available: https://developer.android.com/guide/components/intents-filters Accessed: (10 June 2022).
[71] K. Khariwal, J. Singh, and A. Arora, “Ipdroid: Android malware detection using intents and permissions,” in 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), pp. 197–202, 2020.
[72] S. Wang, Z. Chen, Q. Yan, B. Yang, L. Peng, and Z. Jia, “A mobile malware detection method using behavior features in network traffic,” Journal of Network and Computer Applications, vol. 133, pp. 15–25, 2019.
[73] V. Kouliaridis, K. Barmpatsalou, G. Kambourakis, and G. Wang, “Mal-warehouse: A data collection-as-a-service of mobile malware behavioral patterns,” 10 2018.
[74] N. Peiravian and X. Zhu, “Machine learning for Android malware detection using permission and api calls,” in 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, pp. 300–305, 2013.
[75] Y. Ding, X. Zhang, J. Hu, and W. Xu, “Android malware detection method based on bytecode image,” Journal of Ambient Intelligence and Humanized Computing, 6 2020.
[76] L. Zhao, D. Li, G. Zheng, and W. Shi, “Deep neural network based on Android mobile malware detection system using opcode sequences,” in 2018 IEEE 18th International Conference on Communication Technology (ICCT), pp. 1141–1147, 2018.
[77] K. O. Babaagba and S. O. Adesanya, “A study on the effect of feature selection on malware analysis using machine learning,” ICEIT 2019, (New York, NY, USA), p. 51–55, Association for Computing Machinery, 2019.
[78] J. Singh and J. Singh, “Challenge of malware analysis: malware obfuscation techniques,” International Journal of Information Security Science, vol. 7, no. 3, pp. 100– 110, 2018.

AUTHORS

Hend Faisal is a MSc researcher at the Faculty of Computer and Information Sciences at Ain Shams University, Cairo, Egypt. Hend works as a senior Anti-Malware Software Development Engineer at Egyptian Computer Emergency and Readiness Team (EG-CERT). She received her bachelor degree (2019) in Software Engineering from the Faculty of Computer and Information Sciences at Ain Shams University, Cairo, Egypt. Her research interests include Malware Detection and Machine Learning

Hanan Hindy is a Lecturer at the Computer Science department at the Faculty of Computer and Information Sciences at Ain Shams University, Cairo, Egypt. Hanan did her PhD at the Division of Cyber-Security at Abertay University, Scotland, UK. Hanan received her bachelor degree with honours (2012) and a masters (2016) degrees in Computer Science from the Faculty of Computer and Information Sciences at Ain Shams University, Cairo, Egypt. Her research interests include Intrusion Detection Systems, Machine Learning, and Cyber Security.

Samir Gaber received the B.S. and M.Sc. degrees from the Department of Electronics and Engineering, Helwan University, Egypt, in 1996 and 2003, respectively, the Ph.D. degree in electronic and electrical engineering from the University College London (UCL), U.K., in 2010. Since 2014, he has been an Honorary Lecturer with UCL. Since 2021, he has been the Executive Director of Cyber-attacks Monitoring and Early warning Systems, Egyptian Computer Emergency and Readiness Team (EG-CERT). His research interests include cyber security, malware analysis, and wireless networks.

Abdel-Badeeh Salem is a professor emeritus of Computer Science since September 2007 till now. He was a former Vice Dean of the Faculty of Computer and Information Sciences at Ain Shams University, Cairo-Egypt (1996-2007). He was a professor of Computer Science at Faculty of Science, Ain Shams University from 1989 to 1996. He was a Director of Scientific Computing Center attain Shams University (1984-1990). His research includes intelligent computing, expert systems, medical informatics, and intelligent e-learning technologies.