# A Comparative Analysis of Machine Learning Techniques for Classification and Detection of Malware

Maryam Al-Janabi
Computer Science Department
Applied Science Private
University
Amman, Jordan
maryam.aljanabi@asu.edu.jo

Ahmad Mousa Altamimi
Computer Science Department
Applied Science Private
University
Amman, Jordan
a_altamimi@asu.edu.jo

***Abstract*—Malicious software, commonly known as malware, is one of the most harmful threats developed by cyber attackers to intentionally cause damage or gaining access to computer systems. Malware has evolved over the years and comes in all shapes with different types and functions depending on the goals of the developer. Virus, Spyware, Bots, and Ransomware are just some examples of malware. While those described above found themselves causing issues by accident, however, they all share one thing in common, harming the system. As a response, many infection treatments and detecting methods have been proposed. The signature-based methods are currently utilized to delete malware; however, these methods cannot provide accurate detection of zero-day attacks and polymorphic viruses. Contrarily, the use of machine learning-based detection has been recognized as one of the most modern and notable methods. Specifically, these methods can be categorized based on their analysis technique into static, dynamic, or hybrid. The purpose of this work was to provide a survey that determines the best features extraction and classification methods that result in the best accuracy in detecting malware. Moreover, a review of representable research papers in this topic is represented with a detailed tabular comparison between them based on their accuracy in detecting malware. Among these methods, the J48 algorithm and Hybrid analysis outperformed the others with the accuracy of 100% in detecting malware in the Windows system. On the other hand, the same accuracy has been achieved in the Android system when employing the Decision Tree algorithm through Dynamic analysis. We believe that this study performs a base for further research in the field of malware analysis with machine learning methods.***

***Keywords—Machine Learning, Classification Techniques, Cybersecurity, Malware Detection***

## I. Introduction

From individuals to the most secure and confidential organizations in the world, computers and the internet have used extensively due to the wide range of services provided such as communication, e-learning, entertainment, e-banking, shopping, and alike [1]. Such systems are usually contained private information and are vulnerable to breach or attack. The impact of these attacks could harm these systems and result in significant fines [2]. Malicious software, also called malware, is one of the most harmful attacks, which refers to any software that could damage the system by adding, removing, or altering the data or the system software [3]. Viruses, worms, and Trojan horses are some examples of malware, where their developers (cybercriminals) wreak havoc on other systems to steal essential data, gain access, or cause computers to crash and servers to fail especially in business and private organizations [4].

It is important to mention here that mobile operating systems are becoming one of the biggest targets for malware [5]. Arguably, a lot of personal information is stored on smartphones, so they become a popular target. Therefore, the security argument between iOS, Windows, and Android system is heating up yet again. In a recent study conducted in 2019, the researchers found that Android devices are the most exposed to critical attacks [5]. Another study concluded that more than 90% of Android malicious attacks exploit the known vulnerabilities [6]. Each of these threats requires regular updates from the software companies, which is not always done. So, Android users must be careful about their devices and take extra precautions in securing their data.

Because malware is always developing and growing, and cybercriminals always update them to perform their malicious intents, different malware detection methods have been proposed in the literature [7]. These methods are commonly designed based on the malware signatures, or behavior methods [8]. The Signature-based method is based on the malware signature, a series of bytes that identified the malware, to detect malware. It is the most widely used method in anti-virus and anti-malware products. However, such a method cannot provide accurate detection of zero-day attacks as it requires constant updates about the newly released malware for the detection to work [9]. Contrarily, the Behavior-based method works by monitoring the program's behavior and deciding whether it is malware or not. If the program's behavior is suspicious and difference from typical programs, it will be recognized as malware [10]. This approach has a great advantage over the standard signature-based

978-1-7281-8855-3/20/$31.00 ©2020 IEEE

method as it can determine new malware without the need to analyze them by humans or extracting their signatures. However, it is a time consumed method with high false-positive ratio [10].

Recently, Scientists deploying machine learning and data mining techniques in order to detect malware. This is supported by the nature of machine learning algorithms, where they make it possible to learn from given examples of malware and typical programs to recognize patterns in the behavior of the malware and then can detect it quickly [11]. In fact, machine learning algorithms provide effective detection mechanisms [12]. Being said that, the existence of an arms race in adversarial settings has recently challenged such systems.

This paper presents a survey of malware detection using machine learning techniques. To this end, an overview of malware, including their types, detection techniques, and malware features, is first given. Then, we review notable research papers in this topic grouped based on the malware analysis techniques that were used in them. Ultimately, a detailed tabular comparison between the reviewed research work based on their accuracy in detecting malware is provided. The rest of the paper is structured as follows. Section 2 contains background material about malware, types, detection techniques, and the basic concepts of machine learning. Section 3 reviews the malware detection using machine learning. Section 4 contains a comparison of machine learning techniques for malware detection alongside a tabular comparison that is presented in section 5. Finally, section 6 contains the conclusion and final remarks.

## II. Background Material

### A. Malware

Malware (malicious software) is any program that harms the system. It is any code that's added, removed, or altered in a system or software that causes it to be malevolent and dangerous. Malware is a general term that includes viruses, spyware, Trojans horses, worms, and much more [3]. Malware often causes significant damage through their malevolent behavior. Spying malware hides in the system without the user knowing it, collects the user's data, and sends it to the attacker who created the malware. This can lead to significant damage, especially in private organizations or in business, as they can enter the system through various ways such as vulnerability in the network that permits the malware to infect the systems connected to the network. Downloading files from the internet. Some files already contain malware within them that the user is unaware of. If the browser lacks security, the malicious file could be downloaded and executes itself on the host's machine. The attackers use a powerful method to make the users install malware on their computers, such as: luring the user to install software that would help them watch movies or clicking links in spam emails [13].

The ideal way is to prevent malware infection in the first place. However, this goal is sometimes impossible to achieve. In case the infection happened, the following steps describe a general explanation of the things that need to be done: Firstly, if the system is infected by malware, detect that the infection happened and locate the malicious program. Secondly, identify the malicious program that has infected the system. Finally, all traces of the malware must be removed from the system to restore it to its original state and stop the danger from spreading [9]. In this regard, people usually use software products to protect their systems from malware, such as anti-virus and anti-malware products. Nonetheless, no matter how much progress the anti-malware industry makes, cybercriminals are always developing new methods to enhance malware and make them undetectable. It is like an everlasting battle between the attackers and the malware defenders [1].

### B. Malware Types

Many researchers studied malware types [13-14]. They categorized malware based on their intents and purpose. Below are some of the most common malware:

- Viruses: are code chunks that can infect other programs by appending themselves to program codes. A virus needs to be activated by the host program; it is not able to execute itself individually.
- Trojans: are programs that trick the user by pretending to be helpful and doing tasks that the user needs; however, in the background, they are doing malicious and harmful tasks.
- Worms: Worms do not need to be activated by the host program and can run by itself. A worm can replicate itself once it enters a host program, which makes it more dangerous.
- Spyware: as its name suggests, is a malware that spies on the user. It can steal confidential information, observe the user's activity, and send it to cybercriminals.
- Ransomware: A ransomware is a type of malware that attacks a user's computer and damage it severely. The victim is supposed to pay a ransom to the attacker to decrypt it.
- Scareware: is a malware type that persuades the user to buy and download a harmful and dangerous software that is supposed to be of use to the user, such as anti-malware products or so.
- Rootkits: are hidden programs that ruin the security of the system by disabling specifically chosen programs and grant attackers access to the computer. They can mess up with the operating system and the application programming interface.
- Bots: are malware that causes the infected system to be controlled remotely by the attacker who created the bot. The botmaster can take control of the whole system and install all types of malicious software such as worms, Trojans, spyware, and much more. The infected system can be transformed into a botnet, which is often used to launch Distributed Denial of Service (DDoS) attacks.

- Hybrid Malware: This type is a combination of two or more malware types to enhance the attack and make it more dangerous and damaging.

### C. Mobile Malware

Mobile phones and mobile devices are targeted also by malicious software. It causes loss, leakage of confidential information, or even collapse of the whole system [15]. Many types of malware have been recognized that employ different methods for distribution and infection. In fact, the growth of such a threat is partial because users are now using their mobile devices to access corporate networks. This brings potential threats into their environments as attackers can infect mobile devices and lunch an attack on the network [16]. For example, the Remote Access Tools (RATs) can be utilized by attackers for remotely accessing the infected devices and cryptomining the devices' data. Cryptomining or intelligent collecting of the data (e.g., web browsing history, call history, and installed applications) is done through executing Trojan code on the targeted devices [17].

In fact, almost all mobile operating systems have some vulnerabilities. There are some severe vulnerabilities in Android, just like windows and possibly like all operating systems. There are no operating systems that can claim they do not have any vulnerabilities [18]. For example, the two main dangerous malware that comes preloaded on Android phones were Chamois and Triada [18] that installs background apps, generates ad fraud, or sends special text messages.

### D. Malware Detection Techniques

Three main malware detection techniques are widely used to detect malware:

- Signature-based malware detection: This method is the most widely used in anti-virus and anti-malware products. A signature is an exclusive limited series of bytes for a particular malware; it allows the malware to be identified. Each malware has its signature [19]. However, the method requires constant updates about the newly released malware for the detection to work. For each new malware that's released, the anti-malware companies need to analyze it, discover its signature, and store it in the database that the anti-malware product can access to detect the new malware. Although the error rate may be low, the problem is it takes some time for a newly released malware to be detected by anti-malware tools. The time needed between the release and the anti-malware software update is about 54 days and 180 days in the worst case. The main disadvantage of this technique is that it requires a lot of time and effort to detect malware and extract different signatures for different types of malware [20].

- Behaviour-Based malware detection: This technique works by monitoring the program's behaviour and deciding whether it is malware or not. It detects malware by its suspicious behaviour and difference from typical programs. The behaviour of malware is its significance in this type, and sometimes there are different malware under the same signature [21]. The advantage over the standard signature-based method is that this technique can determine new malware without the need to analyze them by humans and extract their signatures, simply because the behaviour of a new malware may belong to a behaviour signature of malware. However, it has some problems with time consumption and the false-positive ratio [22]. The behavior-based malware detector is made of the following parts: Data collector, which is responsible for gathering information about the executable program. Interpreter, which transforms the information collected from the data collector to a useful representation. Matcher, which compares the representation obtained from the interpreter with the behavior of the program we are analyzing [23].

- Heuristic-Based malware detection: These methods learn the behavior of malware using use machine learning and data mining techniques. It solves many problems present in the signature-based and behavior-based detection techniques [24]. Here, the classification techniques are used for malware detection because classification requires a dataset of input features and output labels to classify malware [25].

## III. Machine Learning for Malware Detection

Machine learning (ML) is a modern technique that involves learning machines from experience. ML can detect hidden patterns that are not hard for users to detect by themselves [26]. The process starts by providing the ML algorithm with an input dataset. A model then is built that can make accurate predictions for new datasets. This way, the machine became more intelligent through learning.

ML techniques are classified into three main categories. Firstly, Supervised learning, which involves learning the machine by using a dataset of examples with their responses. The algorithm uses these examples to respond to any new input based on what it has learned. Classification and regression are examples of supervised learning [27]. Secondly, Unsupervised learning, which uses a dataset without responses. Here, the algorithm categorizes the input based on the similarities between these values. Clustering method is an example of unsupervised learning [27]. Finally, Reinforcement learning, which takes the advantages of the previous two methods and improves its performance through the interaction with the environment. Specifically, it corrects their mistakes and improves its results by trying out different possibilities [27].

It is worth to mention here that the supervised learning technique is the most common type of learning, especially the classification technique. So we mainly focus on this work on the researches that considered the classification algorithms for predicting malware. In fact, various classification algorithms can be used to build a model that can classify whether a new sample is a malware or not based on previous data that the model has trained on. The typical steps in detecting malware are feature extraction and classification. The feature extraction

Identify applicable funding agency here. If none, delete this text box.

step involves extracting features such as API calls, binary strings, and program behaviour. These features express the characteristics of the file, which can help the classifier determine whether the file a malware or not. The second step consists of deploying an intelligent algorithm that can analyze the extracted features and help build the model. The algorithm can be a classification or a clustering algorithm, yet the most widely used is the classification algorithms that take features alongside their labels as input [28].

As mentioned above, machine learning classification techniques are used to detect malware. These techniques require a set of features as their input. Some of the most widely extracted features to detect malware are [29-31]:

API/System calls: Application Programming Interface (API) is the source code that supports services made by programs. It is provided by the operating system. API is one of the best features that can be used by machine learning to predict malware because they capture the program's activity, which makes it easy to know its behaviour.

N-Grams: They are all substrings that can be extracted from a larger string of size N. many researchers carried out malware detection experiments based on this feature.

OpCode: Shortcut for Operational Code, is a section of machine language with the purpose of identifying which operation to be executed. The program is a series of instructions in assembly. Each assembly instruction is made of an OpCode and an operand.

Strings: Strings in the code that are interpretable convey valuable information about the program's malicious intents. Such as HTML tags, for example, the string:

"<html> < scriptlanguage = 'javascript' > window.open (' readme.eml' )" that exists in the "Nimda" worms.

Control Flow Chart (CFG): Is a flow chart that illustrates the control flow of a program and is very popular in software analysis.

## IV. Review of Malware Detection Using Machine Learning

This section presents research done in this area grouped into three categories: static, dynamic, and hybrid [32-37]. These types are used for malware analysis. We will provide an explanation of each alongside its advantages and disadvantages, followed by the research papers that used them. There are research papers targeting malware detection in Windows and Android. Most of the research work targeting Windows focused on analyzing Portable Executable files (PE) because they are the most likely to be infected with malware, while most of the Android researched focused on APK files.

### *A. Static Analysis*

The static analysis deals with analyzing files without executing them. It recognizes specific patterns in the file's binary code. The most significant advantage of static analysis is its ability to detect malware without having to execute the file, which makes it quicker and easier to be used [25]. For example, the features that can be extracted using a static analysis contain a control flow graph (CFG), and opcodes (from disassembling the binary file). [32]. The advantage of static analysis is its speed and safety (since the malware will not be run). On the other hand, the major disadvantage is that it is not always accurate since some patterns do not show up unless the file is executed [34].

Schultz et al. [35] were one of the first to deploy data mining techniques to detect malware. The framework was built using multiple classifiers. The data was gathered from public sources and divided into malicious and benign executables. The executables formats were Windows or MS-DOS. The dataset consisted of 4,266 programs, 3,265 were malware, and 1,001 were benign programs. The malware was downloaded from websites, 95% were Trojans, and 5% were viruses. The clean programs were obtained from the Windows 98 machine. The dataset also contained some Portable Executables (PE) format files. The binary was extracted from the files in the dataset, and then features were extracted from the binary to use in the classifiers. The features extracted did not require executing the binary, so they were collected using static analysis. GNU's Bin-Utils suit was used to analyze the binaries, and the libBFD library was used to extract information in object format. The vital information obtained from object format was: List of DLLs used by the binary, the list of function calls used, and the number of various function calls in each DLL. Next were GNU strings. During the analysis of binaries in libBFD, the headers of files were in plain text. The headers were extracted because they contained useful information. The final feature was the byte sequence. A tool called hexdump was used to convert the binary into hexadecimal files. This led to discovering a sequence of machine code instructions in which each byte sequence was used as a feature. The algorithms that were used in this experiment were RIPPER, Naive Bayes, and a Multi-Classifier, each tried with different features. The RIPPER was used with the DLL generated from libBFD. Naive Bayes used both the strings and the byte sequence to predict the malware based on their features. Finally, the Multi-Naive Bayes, which is a collection of Naive Bayes classifiers, was used. The results showed that the RIPPER, Naive Bayes, and Multi-Naive Bayes classifiers gave an accuracy of 90%, 97.76%, and 95%, respectively.

In [36], Cakir and Dogdu used deep learning methods for malware detection based on static analysis. The dataset used in the research was obtained from Microsoft; it is called Microsoft Malware Classification Challenge Dataset, which contained 10,869 samples of malware from 9 different classes. The research used the word2vec technique, which generates embeddings for word. The word embeddings or vector representation of word was used to classify text. The classifier used was Gradient Boosting Machine (GBM). K-Cross validation method was used to evaluate the built model, and the results showed an accuracy of 96%.

Liu et al. [37] proposed a malware analysis system based on machine learning. The dataset which the experiment was carried on was obtained from ESET, NOD32, VX Heavens, and Threat trace security. A total of 21,740 malware were captured, which can be categorized into viruses, worms, backdoors, or Trojan horses. The dataset was split into 19,740 samples for training and 2,000 samples for testing. For the data preprocessing part, three methods were used. Firstly, the grayscale image method (a static analysis method) in which the contents of the malware's binary file are mapped into a matrix called the grayscale matrix based on some criteria. The grayscale matrix is converted into a grayscale vector, so it is possible to work with it. IDA Pro, which is a disassembler, was used to extract the binaries of malware.

Secondly, the n-gram model (accurately the 3-gram model) was used to obtain the opcode features of malware. Furthermore, thirdly, the control flow graph (CFG) was used to analyze the malware. These methods were used to extract malware features. In order to improve the classification further, feature selection was applied to reduce the number of features obtained from the previous step. The information gain was calculated and used to reduce the features to 500 features. Due to the high dimensionality of data, the shared nearest neighbour clustering algorithm SNN technique was used to cluster the data, then several classifiers were applied such as Forest Random (RF), Naive Bayes (NB), Logistic Regression (LR), K-NN, Gradient-Boosting (GB), SVM-poly (SP), and Decision tree (DT). The experiment used an Oracle database, python 2.7 alongside scikit-learn, which is a python module for machine learning. The results showed that the system had a classification accuracy of 98.9%, and can successfully detect 86.7% of new malware.

### B. *Dynamic Analysis*

The dynamic analysis, on the contrary of static analysis, executes the file, whether in a real or virtual environment, for example, in a sandbox application to check if it is a malware or not. The problems associated with dynamic analysis are: firstly, it is sometimes hard to build the required controlled environment for malware execution. Secondly, it is time-consuming to observe the malware's behaviour, primarily if it was run in a slow virtual environment [25]. Dynamic analysis can help extract features like API calls, system calls. Register changed, memory writes, instruction traces, etc. [32]. The main advantage of dynamic analysis is its accuracy and pattern matching since the file will be executed. However, this type of analysis is a lot more time-consuming in comparison with the static analysis, mainly if the file is run in a virtual environment (which happens most of the time) [34].

In [18], Ye et al., an Intelligent Malware Detection System (IMDS), were developed using Object-Oriented Association (OOA) based classification. The system contained three components, namely PE parser, OOA rule generator, and malware detection module. The PE parser collected windows API calls for each file. If the file was compressed, it was decompressed using dissembler W32Dasm. The API calls generated would be stored in a database and treated as signatures for the PE files. The OOA algorithm was used to obtain the class association rule. Finally, to detect whether a file is a malware or not, the API calls were passed to the malware detection module alongside the generated rules. The data used in the system consisted of 29,580 PE files, 12,214 were benign, and 17,366 were malicious, containing worms, Trojan horses, and backdoors. The malware was obtained from Kingsoft Anti-Virus Corporation laboratories. The IMDS was compared to other machine learning algorithms such as Naive Bayes, SVM, and J48 algorithms with the help of WEKA and C++. The IMDS outperformed the other classifiers with 93.07% accuracy.

Shabtai et al. [21] developed "Andromaly," which is a malware detection framework for Android. The framework used machine learning classification methods to detect malware. Several classifiers were used, such as k-Means, Logistic Regression, Histograms, Decision Tree, Bayesian Networks, and Naive Bayes. However, four attack classes were proposed: unsolicited information, theft-of-service, information theft, and Denial-of-Service (DoS), and four malicious applications were created to perform the intended malicious tasks. Two for DoS attacks since they either overwhelm the device with requests or they drain the device's power. Two for information theft since they may be either target stealing the device's location and other temporary data (Transient information) or device's sensitive data such as contact information, login details, or any static information. The four malicious programs developed by the researchers for this experiment were: Tip Calculator, Schedule SMS, and Lunar Lander, Snake, and HTTP upload. The malware was installed on two android devices and ran alongside other applications, while in the background ran the malware detection system that classified the applications into either "game," "tool," or "malware" and stored them in a database every 2 seconds. The researchers conducted four experiments. For experiments I and II, Decision Tree gave the best accuracy. Moreover, for experiments III and IV, Naive Bayes outperformed the rest. The best accuracy in the research was the Decision Tree in experiment I at 100% for predicting the "game" class and 99% for predicting the "tools" class.

Firdausi et al. [22] analyzed different machine learning techniques for malware detection. The dataset used consisted of a total of 220 malware (Indonesian malware samples). 250 benign files were obtained from the Windows XP system. The analysis of the data was done using a free online service named Anubis, which performed dynamic analysis and generated a report file. The classification was carried out using the WEKA tool. Various algorithms were tried out, such as Support Vector Machine (SVM), k-Nearest Neighbor, Naive Bayes, J48 decision tree, and Multilayer Perceptron neural network. The authors experimented with the classifiers, both with and without feature selection. The feature selection was made using the Correlation-Based technique, which reduced the attributes from 5,191 to 116 attributes only. The best accuracy obtained was using the J48 classifier without feature selection, which yielded 96.8% of accuracy.

### C. *Hybrid Analysis*

The hybrid analysis combines both static and dynamic analysis. Zakeri et al. [25] proposed a system using fuzzy classification algorithms to detect malware and packed files. The dataset used consisted of 63,000 windows PE files containing 7,000 benign files gathered from Windows XP and Windows 7 and some other applications, and 56,000 malicious files collected from malicious sites such as Virus Collection, Offensive, and Computing. About 21,000 files in the dataset were compressed or packed. PEiD tool was used to separate them, and OllyDbg tool was used for analyzing whether a file is packed or not. The authors decided to use the static method to study the input files and separate the benign files from files that show some malicious signs. The suspected malicious files were then investigated using a dynamic method. This way, they benefited from both the speed of the static approach and the accuracy of the dynamic approach. Over 70 features were used in the experiment to improve accuracy. Some were extracted from the PE file headers, while others were calculated from anomalies found in PE file headers. Each feature was compared to the rest and assigned a different weight based on its information gain (IG) that was calculated using the information theory. Various machine learning algorithms were used, such as Instance-Based Learner (IBk), Naive Bayes (NB), Decision Tree (J48), J48-Graft, Fuzzy Unordered Rule Induction Algorithm (Furia), and Inductive Rule Learner (RIPPER). The tool used was the Waikato Environment for Knowledge Analysis (WEKA). The system achieved an overall accuracy of 99.97% in detecting packed files.

In [23], Sethi et al. developed an intelligent malware detection framework for classifying various files such as exe, pdf, and PHP into either malware or not. The dataset in this research was built from scratch by the authors. The malware samples were obtained from OpenMalware and the clean files from Softonic. To analyze the dataset, Cuckoo Sandbox was used. It generated static and dynamic analysis by running the files in a virtual environment. The output of the Cuckoo Sandbox was an analysis report. The report contained essential features that would be used in the classification phase, such as CuckooScore, CuckooIsMalware, and CuckooMalwareType. The WEKA tool was used for comparing multiple machine learning classification algorithms. The results showed an accuracy of 100%, 99%, and 97% using J48, SMO, and Random Forest.

In [38], Yuan et al. developed the DroidDetector, which detects malware in androids. The DroidDetector was put for online use, and any user could upload an APK file to it to discover if it is a malware or not. The authors applied deep-learning in the experiment and used both static and dynamic analysis techniques to extract features. The APK file of an app is what was needed to determine whether it is malware or not. The static phase included analyzing the APK file. The 7-zip tool was used to uncompress the APK file and then AndroidManifest.xml and classes.dex were parsed to obtain the application's permissions from AndroidManifest.xml and API calls from classes.dex. The dynamic phase consisted of running the app in DroidBox, which is a sandbox application. It could execute the APK file and monitor its behaviour to get the dynamic behaviour of the application. Examples of the application's behaviour include short message services (SMS), phone calls, and network input/output. A total of 192 features were extracted through both dynamic and static analysis. The deep-learning model that was applied is Deep Belief Networks (DBN). The dataset was built using a wide range of 20,000 benign applications from the Google Play store and 192 malicious applications collected from the Contagio Community and Genome Project. The model was trained and tested using the dataset mentioned earlier. The DBN achieved an accuracy of 96.76% with an architecture of 2 layers and 150 neurons for each layer.

Wen and Yu in [39] proposed a light-weight system using machine learning to detect malware on Android devices. The system is divided into two major parts, client and server. The client is responsible for providing the user interface (UI) that displays the prediction of the system. The server is responsible for static and dynamic analysis of the application. In the static analysis phase, the Androguard tool, which is a decoder, was used to decompile the application and extract features from the AndroidManifest.xml file and classes. Dex file. The features that were extracted in the static analysis stage included: API calls, user-features, intents, and permissions. The dynamic analysis phase consisted of using the DroidBox virtual environment to test the application. The features that the dynamic analysis extracted were: battery and CPU consumption and the number of messages and running processes. The system focuses mainly on static analysis because the dynamic analysis is time-consuming and requires much computational power.

The dynamic analysis is used in case the static analysis fails in decompiling the APK file of the application. Both Relief, which is a feature selection algorithm, and PCA, which is a dimensionality reduction algorithm, were combined to produce PCA-Relief and were used for the feature selection phase. The dataset used consisted of 2,000 Android applications, 1,000 of them were benign applications obtained from Google Play by the crawler technology, and 1,000 of them were malware collected from the Drebin Project and the Android Malware Genome Project. To build the classifier, the Support Vector Machine (SVM) was used, and the dataset was divided into 80% for training and 20% for testing. The experiments were carried out using PCA only, Relief, or both PCA-Relief as the feature extraction technique. The accuracy of combining PCA-Relief methods alongside SMV gave the best accuracy of 95.2%.

## V. Publication principles

A tabular comparison between all the research mentioned above is given next. The comparison is made based on accuracy and can be seen in Table 1. The table has eight elements, as follows:

Author that contains the researcher/s who made the paper.
Machine Learning Technique/s, which contains the algorithms employed in the research.
Best Technique Found that represents the algorithm with the highest accuracy in the research.
Dataset: The dataset used to build the detection system.
Tool: The tools or programming languages used to build the model or perform the required malware detection.
Analysis Method: The malware analysis method, either static, dynamic, or hybrid.
System: The environment targeted to detect malware in it, either Windows or Android.

Accuracy: The accuracy found by the author/s and presented in the research.

## VI. Conclusions and Final Remarks

This paper presented an overview of the malware detection methods using machine learning alongside basic concepts of both topics of malware detection and machine learning. Several representational researches were surveyed and categorized based on their analysis technique, either static, dynamic, or hybrid. The accuracy of the proposed models varies depending on the used method, the number of attributes, the dataset and records in the dataset, the preprocessing techniques, as well as the tools implemented in the model. It also depends on the analysis technique and the used feature. As seen in Table 1, the researchers that presented the best accuracy in detecting malware in the Windows system were Sethi and Chaudhray [23]. They achieved an accuracy of about 100% and used the J48 algorithm and Hybrid analysis. On the other hand, in the Android system, the researchers that achieved the highest accuracy, according to the surveyed papers were Shabtai et al. [21]. The Decision Tree algorithm they used through Dynamic analysis achieved an accuracy of 100% and 99% in experiment I for predicting "game" and "tool" classes.

J48 is a type of Decision Tree, so based on the surveyed papers, the Decision Tree algorithm achieved the best results. Nevertheless, admittedly, this conclusion does not allow us to make a statement that Decision Tree is the best algorithm for malware detection. There are always other measures to take care of, such as the dataset, the preprocessing techniques, the features extracted, and the feature selection methods used. If the Decision tree algorithm is used in a model that has fewer features or the analysis techniques that were used were weak, then it is not guaranteed to perform excellently.

TABLE I. Machine Learning Techniques for Malware Detection

| Analysis Method | Author | Machine Learning Technique/s | Best Technique Found | Dataset | Tools | System | Accuracy |
|---|---|---|---|---|---|---|---|
| Static | Schultz et al. [35] | RIPPER, NB, and Multi-Naive Bayes | NB | 4,266 programs: 3,265 malware and 1,001 benign programs | GNU's Bin-Utils, libBFD, and hexdump | Windows | 97.76% |
| | Cakir and Dogdu [36] | GBM | *n/a | Microsoft Malware Classification Challenge Dataset (contains 10,869 malware) | n/a | Windows | 96% |
| | Liu et al. [37] | SNN, RF, K-NN, GB, NB, LR, SP and DT | n/a | 21,740 malwares | IDA Pro, Oracle and Python 2.7 | Windows | 98.9% |
| Dynamic | Ye et al. [42] | IMDS, NB, SVM, and J48 | IMDS | 29,580 files, 17,366 malware and 12,214 benign | W32Dasm WEKA and C++ | Windows | 93.07% |
| | Shabtai et al.[21] | k-Means, Logistic Regression, Histograms, DT, Bayesian Networks, and NB | DT in experiment I | Created by the authors | n/a | Android | 100% for predicting the "game" class and 99% for predicting the "tools" class |
| | Firdausi et al.[22] | KNN, NB, SVM, J48 DT, and Multilayer Perceptron (MLP) neural network | J48 without feature selection | 250 Indonesian malware and 250 benign files | Anubis and WEKA | Windows | 96.8% |
| Hybrid | Zakeri et al.[25] | IBK, J48 DT, J48-Graft, NB, inductive rule learner (RIPPER) and FURIA | n/a | 63,000 files, 56,000 malware, and 7,000 benign files | PEiD, OllyDbg, and WEKA | Windows | 99.97% |
| | Sethi et al.[23] | J48 decision tree, Random Forest and SMO | J48 | malware from OpenMalware and clean files from Softonic | Cuckoo Sandbox and WEKA | Windows | 100% |
| | Yuan et al. [38] | Deep Belief Networks (DBN) | n/a | 192 malicious apps from Contagio Community and Genome Project and 20,000 benign apps from Google Play store | 7-zip and DroidBox | Android | 96.76% |
| | Wen and Yu [39] | SVM | n/a | 2,000 android applications, 1,000 malware and 1,000 benign | Androguard and DroidBox | Android | 95.2% |

*n/a: not applicable.

Finally, the field of using machine learning for malware detection is proven to be of great importance to both scientists and researchers [40-41]. Recently a success of neural networks in the malware domain has shown in many studies [43-44]. The motivation behind neural network approaches is to build detection systems that do not rely on the experts' knowledge of the domain to define discriminative features. In fact, there is much research works done in this area because machine learning can help us detect malware faster and more efficiently [34]. The research opportunities are always open in the machine learning area for malware detection.

## ACKNOWLEDGMENT

The authors are grateful to the Applied Science Private University, Amman-Jordan, for the full financial support granted to cover the publication fee of this research article.

## REFERENCES

[1] Y. Ye et al., "A survey on malware detection using data mining techniques," ACM Computing Surveys (CSUR), vol. 50, no. 3, pp. 41:1–41:40, 2017.

[2] E. Gandotra et al., "Malware analysis and classification: A survey," Journal of Information Security, 2014.

[3] N. Idika and A. P. Mathur, "A survey of malware detection techniques," Purdue University, 2007.

[4] Chip Epps,"Best practices to deal with top cybercrime activities," Computer Fraud & Security}, vol. 2017, no. 4, pp. 13-15, 2017.

[5] M. Alejandro et al., "The Android OS stack and its vulnerabilities: an empirical study," Empirical Software Engineering, vol. 24, no. 4, pp. 2056-2101, 2019.

[6] F. Xuan et al.,"Understanding and Securing Device Vulnerabilities through Automated Bug Report Analysis," USENIX Security Symposium, pp. 887-903, 2019.

[7] PB. Pathak, "Malware a growing cybercrime threat: Understanding and combating malvertising attacks," International Journal of Advanced Research in Computer Science," vol. 7, no. 2, 2016.

[8] M. Hashimoto et al., "A survey of security research for operating systems," IPSJ Transactions on Advanced Computing Systems (ACS), vol. 5, no. 2, pp. 51–62, 2012.

[9] A. Silberschatz, P. B. Galvin, and G. Gagne, Operating System Concepts, 9th ed. Wiley Publishing, 2012.

[10] T. Dube, "Malware type recognition and cyber situational awareness," 2010 IEEE Second International Conference on Social Computing, pp. 938-943, 2010.

[11] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," Human-centric Computing and Information Sciences, vol. 8, no. 1, pp. 1–125:22, 2018.

[12] A. Azmoodeh, "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning," IEEE Transactions on Sustainable Computing, vol. 4, no. 1, pp. 88-95, 2018.

[13] R. Sihwail et al., "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," International Journal on Advanced Science, Engineering and Information Technology," vol. 8, no. 4-2, pp. 1662, 2018.

[14] R. Pirscoveanu, "Analysis of malware behavior: Type classification using machine learning," 2015 International conference on cyber situational awareness, data analytics and assessment (CyberSA), pp. 1-7, 2015.

[15] Wang, Shanshan, et al. "A mobile malware detection method using behavior features in network traffic." Journal of Network and Computer Applications 133 (2019): 15-25.

[16] Qamar, Attia, Ahmad Karim, and Victor Chang. "Mobile malware attacks: Review, taxonomy & future directions." Future Generation Computer Systems 97 (2019): 887-909.

[17] M. Yamada, M. Morinaga, Y. Unno, S. Torii and M. Takenaka, "RAT-based malicious activities detection on enterprise internal networks," 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, 2015, pp. 321-325.

[18] Kouliaridis, Vasileios, et al. "A survey on mobile malware detection techniques." IEICE Transactions on Information and Systems 103.2 (2020): 204-211.

[19] Ojugo, A., and A. O. Eboka. "Signature-based malware detection using approximate Boyer Moore string matching algorithm." International Journal of Mathematical Sciences and Computing 5.3 (2019): 49-62.

[20] Scott, James. "Signature based malware detection is dead." Institute for Critical Infrastructure Technology (2017).

[21] A. Shabtai et al., "andromaly": A behavioral malware detection framework for android devices," Journal of Intelligent Information Systems, vol. 38, no. 1, pp. 161–190, 2012.

[22] I. Firdausi et al., "Analysis of machine learning techniques used in behavior-based malware detection," in Proceedings of the 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies. Washington, DC, USA: IEEE Computer Society, 2010, pp. 201–203.

[23] K. Sethi et al., "A novel malware analysis for malware detection and classification using machine learning algorithms," in Proceedings of the 10th International Conference on Security of Information and Networks. New York, NY, USA: ACM, 2017, pp. 107–113.

[24] Z. Bazrafshan et al., "A survey on heuristic malware detection techniques," in The 5th Conference on Information and Knowledge Technology, 2013, pp. 113–120.

[25] M. Zakeri, F. Faraji Daneshgar, and M. Abbaspour, "A static heuristic approach to detecting malware targets," Security and Communication Networks, vol. 8, no. 17, pp. 3015–3027, 2015.

[26] H. Almarabeh and E. Amer, "A study of data mining techniques accuracy for healthcare," International Journal of Computer Applications, vol. 168, no. 3, pp. 12–17, Jun 2017.

[27] M. Fatima and M. Pasha, "Survey of machine learning algorithms for disease diagnostics," Journal of Intelligent Learning Systems and Applications, vol. 9, no. 01, pp. 1–16, 2017.

[28] M. Siddiqui, M. C. Wang, and J. Lee, "A survey of data mining techniques for malware detection using file features," in Proceedings of the 46th Annual Southeast Regional Conference on. New York, NY, USA: ACM, 2008, pp. 509–510.

[29] Grosse, Kathrin, et al. "Adversarial examples for malware detection." European Symposium on Research in Computer Security. Springer, Cham, 2017.

[30] Narudin, Fairuz Amalina, et al. "Evaluation of machine learning classifiers for mobile malware detection." Soft Computing 20.1 (2016): 343-357.

[31] Cepeda, Carlos, Dan Lo Chia Tien, and Pablo Ordónez. "Feature selection and improving classification performance for malware detection." 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom). IEEE, 2016.

[32] A. Damodaran et al., "A comparison of static, dynamic, and hybrid analysis for malware detection," Journal of Computer Virology and Hacking Techniques, vol. 13, no. 1, pp. 1–12, 2017.

[33] V. Rao and K. Hande, "A comparative study of static, dynamic and hybrid analysis techniques for android malware detection," International Journal of Engineering Development and Research (IJEDR), vol. 5, no. 2, pp. 1433–1436, 2017.

[34] Shalaginov et al., "Machine learning aided static malware analysis: A survey and tutorial," Journal of Cyber Threat Intelligence, pp. 7—45, 2018.

[35] M. G. Schultz et al., "Data mining methods for detection of new malicious executables," in Proceedings of the 2001 IEEE Symposium on Security and Privacy. Washington, DC, USA: IEEE Computer Society, 2001, pp. 38–49.

[36] B. Cakir and E. Dogdu, "Malware classification using deep learning methods," in Proceedings of the ACMSE 2018 Conference. New York, NY, USA: ACM, 2018, pp. 10:1–10:5.

[37] L. Liu et al., "Automatic malware classification and new malware detection using machine learning," Frontiers of Information Technology & Electronic Engineering, vol. 18, no. 9, pp. 1336–1347, 2017.

[38] Z. Yuan, Y. Lu and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," Tsinghua Science and Technology, vol. 21, no. 1, pp. 114–123, 2016.

[39] L. Wen and H. Yu, "An android malware detection system based on machine learning," in AIP Conference Proceedings, vol. 1864, no. 1. AIP Publishing, 2017, pp. 020-136.

[40] Yeo, M et al., "Flow-based malware detection using the convolutional neural network," 2018 International Conference on Information Networking (ICOIN), 2018, pp. 910—913.

[41] A. Kumara et al., "Automated multi-level malware detection system based on the reconstructed semantic view of executables using machine learning techniques at VMM," Journal of Future Generation Computer Systems, vol. 79, pp. 431—446, 2018.

[42] Y. Ye et al., "An intelligent pe-malware detection system based on association mining," Journal in Computer Virology, vol. 4, no. 4, pp. 323–334, 2008.

[43] Raff, Edward, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles Nicholas. "Malware detection by eating a whole exe." arXiv preprint arXiv:1710.09435 2017.

[44] Krčál, Marek, Ondřej Švec, Martin Bálek, and Otakar Jašek. "Deep convolutional malware classifiers can learn from raw executables and labels only." 2018.