IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 February 2023, accepted 5 March 2023, date of publication 14 March 2023,
date of current version 19 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3256979

TOPICAL REVIEW

Machine Learning Algorithm for Malware Detection: Taxonomy, Current Challenges, and Future Directions

NOR ZAKIAH GORMENT[1,2], (Member, IEEE), ALI SELAMAT[1,3,4,5], (Member, IEEE), LIM KOK CHENG[2], (Member, IEEE), AND ONDREJ KREJCAR[1,5]

[1]Malaysia–Japan International Institute of Technology, Universiti Teknologi Malaysia, Kuala Lumpur 50088, Malaysia
[2]College of Computing and Informatics, Universiti Tenaga Nasional, Jalan IKRAM-UNITEN, Kajang, Selangor 43000, Malaysia
[3]School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Johor Baru, Johor 80000, Malaysia
[4]Media and Games Center of Excellence (MagicX), Universiti Teknologi Malaysia, Johor Baru, Johor 80000, Malaysia
[5]Center for Applied Research, Faculty of Informatics and Management, University of Hradec Kralove, 050003 Hradec Kralove, Czech Republic

Corresponding authors: Ali Selamat (aselamat@utm.my), Ondrej Krejcar (ondrej.krejcar@uhk.cz), and Nor Zakiah Gorment (zaagorment@gmail.com)

This work was supported in part by the Fundamental Research Grant Scheme (FRGS) through the Ministry of Education under Grant FRGS/1/2022/ICT08/UTM/01/1; and in part by the Faculty of Informatics and Management, University of Hradec Kralove, through the Specific Research Project (SPEV) under Grant 2102/2023. The authors are also grateful for the support of student Michal Dobrovolny in consultations regarding application aspects.

ABSTRACT Malware has emerged as a cyber security threat that continuously changes to target computer systems, smart devices, and extensive networks with the development of information technologies. As a result, malware detection has always been a major worry and a difficult issue, owing to shortcomings in performance accuracy, analysis type, and malware detection approaches that fail to identify unexpected malware attacks. This paper seeks to conduct a thorough systematic literature review (SLR) and offer a taxonomy of machine learning methods for malware detection that considers these problems by analyzing 77 chosen research works related to malware detection using machine learning algorithm. The research investigates malware and machine learning in the context of cybersecurity, including malware detection taxonomy and machine learning algorithm classification into numerous categories. Furthermore, the taxonomy was used to evaluate the most recent machine learning algorithm and analysis. The paper also examines the obstacles and associated concerns encountered in malware detection and potential remedies. Finally, to address the related issues that would motivate researchers in their future work, an empirical study was utilized to assess the performance of several machine learning algorithms.

INDEX TERMS Malware detection, machine learning algorithms, state-of-the-art.

I. INTRODUCTION

Malware is still a primary concern worldwide, and the nature of malware is continually changing as technology advances. This happens because computer system usage and internet connection are highly in demand. Thus, malware attacks have caused a severe threat to computer software and smart devices and have become a real challenge [1] to secure the data for professional or personal uses.

A. RESEARCH MOTIVATION

Several research articles that explicitly employ machine learning algorithms to identify malware have been published; however, none of the studies contain a comparative analysis of several machine learning methodologies. Furthermore, although malware detection approaches are widely explored, there is a lack of information on machine learning algorithms' effectiveness in detection rates, accuracy rates, analysis type, and classification methods. This scenario has led to insufficient evidence restricting malware detection usage in related research areas. The current challenges and future directions on machine learning algorithms to detect malware also need to be highlighted.

The associate editor coordinating the review of this manuscript and approving it for publication was Jemal H. Abawajy.

B. RESEARCH CONTRIBUTION

The comparison studies between our research work and existing related research initiatives are shown in TABLE 1, and their abbreviations are explained in TABLE 2. From 2017 through 2022, we present the most recent research on ten types of machine-learning algorithms for malware detection. We thoroughly examine machine learning algorithms for identifying malware utilizing a systematic literature review (SLR) methodology, which sets us apart from past work. Our research also contains cutting-edge machine-learning techniques and examines existing limitations and future research directions in machine learning for malware detection.

This study employs an SLR to provide the research community with extensive research on a machine learning approach motivated by a lack of research efforts. The following are the significant contributions made by this research study.

1) A complete review of machine learning methods for malware detection was provided.
2) We offered a malware detection taxonomy and a machine-learning approach for categorizing malware into various classifications.
3) We addressed the challenges and related issues faced in malware detection, highlighting to propose suitable solutions.
4) We conducted an empirical study to evaluate the efficacy of numerous machine learning algorithms and address related difficulties that might motivate future research.

TABLE 1. Existing studies' limitations and the novelty of this research.

No.	Type of MLA	Remark	References
1	DT, RF, k-NN, GNB, BNB, ADB, LR, SVM	• Compare the performance of 8 types of MLA. • There is no discussion of existing MLA technology or the current obstacles for future directions.	[6]
2	NB, Bayesian, Hybrid, Ada grad	• The studies were conducted from the Year 2016 to 2017. • There is no info on the state-of-the-art MLA.	[5]
3	NB, ADB, DT, Bagging, SMO	• The studies were conducted from the Year 2013 to 2017. • There is no discussion on the current challenges for future research works.	[3]
4	SVM, NB, DT, RF, LR, and k-NN	• Compare the performance of 6 types of MLA • There is no discussion on the current challenges for future research works.	[4]
5	SVM, DT, FR and NB	• The studies were conducted from the Year 2013 to 2019. • Compare the performance of 4 types of MLA • There is no discussion of existing MLA technology or the current obstacles for future directions.	[2]
6	K-means, DT, NB, SVM, k-NN, RF	• The studies were conducted from the Year 2016 to 2021. • Compare the performance of 6 types of MLA • There is no discussion on the current challenges for future research works.	[7]
7	DT, NB, SVM, RF, ADB	• The studies were conducted from the Year 2016 to 2017. • Compare the performance of 5 types of MLA.	[8]
#	Meta-heuristic, Neuro-fuzzy, K-means, Gaussian, DT, Bayesian, NB, SVM, k-NN, n-grams	• The studies were conducted from the Year 2017 to 2022. • Compare the performance of 10 types of MLA. • The most up-to-date MLA is shown. • Discuss current issues and future research activities.	**Our research work**

TABLE 2. List of acronyms for the machine learning algorithm.

Abbreviation	Explanation
SLR	Systematic Literature Review
ML	Machine Learning
MLA	Machine Learning Algorithm
NB	Naïve Bayes
SVM	Support Vector Machine
DT	Decision Tree
GB	Gaussian Bayes
NF	Neuro-fuzzy
KNN	K-Nearest Neighbour
RF	Random Forest
GNB	Gaussian Nave Bays Classifier
BNB	Bernoulli Naive Bays Classifier
ADB	AdaBoost Classifier
LR	Logistic Regression
SMO	Sequential Minimal Optimization
TPR	True Positive Rate
FPR	False Positive Rate
FNR	False Negative Rate
ROC	Receiver Characteristic Operator
RQ	Research Question
IoT	Internet of Thing

The following is how the rest of the research is organized: Section II provides the research methodology while a background study of malware, malware attacks, machine learning algorithms for malware detection, and previous research works for malware detection using machine learning algorithms represented in Section III. Section IV provides a taxonomy of malware detection using a machine-learning algorithm to categorize them based on several classifications. Section V discusses the current concerns and obstacles that machine learning faces in the fight against malware. Section IV analyses any potential study gaps and makes recommendations for further research. An empirical analysis and a mapping of current challenges with the present research gap are included in Section VII. Finally, Section VIII summarizes the study's findings and recommends further research.

II. RESEARCH METHODOLOGY

This part is divided into two primary sub-sections, each providing a comprehensive understanding of the research domain. The first sub-section introduces the SLR method used in this research investigation. The second sub-section examines malware's history, machine learning, and machine learning methods to identify malware.

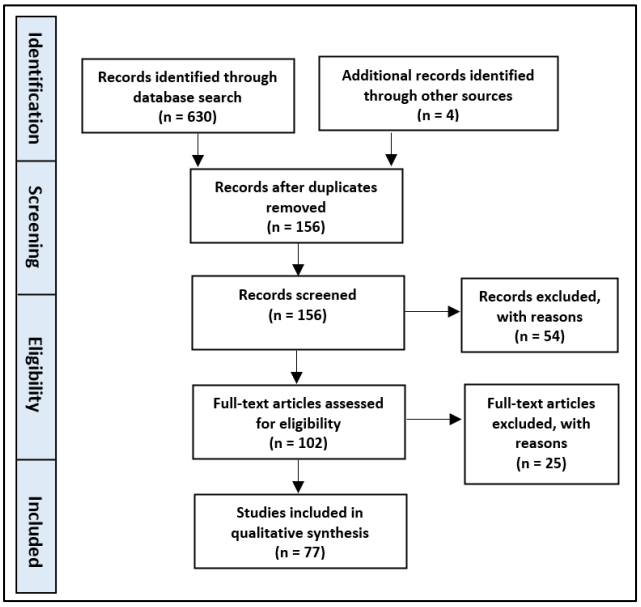

FIGURE 1. SLR process in selecting the relevant studies.

A. SYSTEMATIC LITERATURE REVIEW

The SLR guidelines were adopted from Kitchenham [9], and the selection process by PRISMA guidelines [10]. The starting process is the formulation of review questions. The following step is to create and validate a review methodology, after which we will search for primary screen studies using the review protocol's criteria. The whole text of the chosen papers was identified utilizing the review technique to determine their quality. Finally, the result was extracted from the review, where data was analyzed and synthesized [123]. FIGURE 1 illustrates the process of selecting the relevant studies. The reviewing method can be seen in the following subsections.

1) RESEARCH QUESTION

This study aims to examine and evaluate a variety of malware detection machine-learning techniques. To be highlighted in this SLR, a list of research questions (RQs) has been created. The following are the review's research questions:

RQ1:What kind of machine learning algorithm was used?
RQ2:What is the mechanism of the machine learning algorithm?
RQ3:How is the algorithm's performance measured?
RQ4:What categorization method was used?
RQ5:What kind of analysis was used?
RQ6:What restrictions and obstacles were found while running the algorithm?

2) REVIEW PROTOCOL

The review processes are based on the SLR standards derived by Kitchenham [9], [10]: search strategy, inclusion and exclusion criteria, quality evaluation, data extraction, and data analysis.

a: STRATEGY OF SEARCH

An automatic search strategy was built based on the study questions. The syntax and search methods from numerous digital libraries have been used in database searching to address the query string. Meanwhile, the search scope and query strings are as follows:

- Query String - (''malware detection'' or ''anti-malware'' OR ''malicious detection'') AND (''machine learning algorithm'' OR ''machine learning approach'') AND (''K-means'' OR ''NaC/ve Bayes'' or ''Support Vector Machine'' OR ''Decision Tree'' OR ''Meta-heuristic'' OR ''Neuro-Fuzzy'' OR ''Bayesian'' OR ''Gaussian'' OR ''K-Nearest Neighbour'' OR ''N-grams'')
- The timespan to gather the studies is from 2017 to 2022.
- The medium of writing the survey is using the English language.
- Different reference sources, including journals, symposiums, conferences, workshops, and book chapters, are referred.

IEEE Explore and Mendeley were chosen as digital libraries, which gathered papers from Scopus, SpringerLink, ACM Digital, Science Direct, and Web of Science. Both digital libraries were used to undertake snowballing and identify intended research papers during the search phase. All articles that fulfilled the inclusion and exclusion criteria were sorted for further examination. Furthermore, the starting timeframe is from 2017 to 2022, with only the most recent and up-to-date papers included in this SLR.

b: CRITERIA OF INCLUSION AND EXCLUSION

Based on the research objectives, inclusion and exclusion criteria were developed to narrow down the relevant literature for this SLR. TABLE 3 shows the shortlisted studies for inclusion and exclusion based on their ability to satisfy the criteria.

The selection procedure was divided into three steps. The first of which was the search for all potential primary studies. The following step was viewing and reading the titles and abstracts of all papers found in the search results. Then we found all the research that satisfied the criteria for inclusion and exclusion. Finally, all the discovered studies were read in their entirety before being shortlisted for final selection.

The selection procedure was divided into three steps. The first of which was the search for all potential primary studies. The following step was viewing and reading the titles and abstracts of all papers found in the search results. Then we

found all the research that satisfied the criteria for inclusion and exclusion. Finally, all the discovered studies were read in their entirety before being shortlisted for final selection.

c: ASSESSMENT OF QUALITY

The quality of the research studies that satisfied the inclusion and exclusion criteria was assessed using the ten criteria used to evaluate the studies' credibility, relevance, rigorousness, and independence. Meanwhile, the entire text of each paper VOLUME XX, 2022 9 was examined, and the assessment criteria were applied to rate its quality. The quality evaluation question criteria derived from the checklists published by [11] and [12] are shown in TABLE 5. Each question had four possible scores, as shown in TABLE 4. All primary studies were sorted out based on their score, as shown in TABLE 12 in Appendix A.

TABLE 3. Criteria for inclusion and exclusion.

No.	Inclusion Criteria	Exclusion Criteria
1	The study either described or applied a machine-learning algorithm to support malware detection activities, including the related processes	The study described the machine learning algorithm. However, the machine learning algorithm was not implemented or used for malware detection
2	If the study was a review paper, the studies were summarized, and each publication was given its treatment	The study was identical to the existing selected paper or considered as duplicate paper
3	The study was deemed the latest version among other studies in the same field. Only the latest version is selected	The study was not used English as a medium for writing the research works

d: DATA EXTRACTION

An in-depth analysis was conducted to illustrate the research questions, and the necessary data was gathered. Based on the selected primary studies, the following data were extracted to be included in a predefined extraction form.

- Type, bibliographic information, and reference ID
- Name of publication
- Country of institution
- Discipline of research
- The type of machine learning technique used to detect malware
- Algorithms, models, and ideas that are fundamental
- Identification of machine learning algorithm with a specific classification approach and analysis type
- Tools were used to support the malware detection process.

TABLE 4. The score for assessment of quality.

No.	Description	Score
1	Thoroughly addressed	3
2	Discussed adequately	2
3	Little mentioned	1
4	Not mentioned at all	0

e: DATA ANALYSIS

After extracting data from each main study, in-depth data analysis was conducted to address each research question. To answer RQ1, the machine learning algorithms implemented were identified, and to answer RQ2, the machine learning algorithms were evaluated to see how they worked. For each category, related ideas or models were discovered. Meanwhile, the algorithm's outputs were assessed in terms of performance to answer RQ3. RQ5 has the same classification methods and analysis type as RQ4. Finally, to answer RQ6, any limitations and challenges found during the execution of the machine learning algorithm were identified.

TABLE 5. Quality assessment criteria.

No.	Criteria
	Problem Statement:
Q1	Is the objective of the research adequately defined and well-motivated?
	Research Design:
Q2	Is the machine learning algorithm's performance sufficient to assist the malware detection process?
Q3	Are the machine learning algorithms' categorization method and analysis type well stated?
	Data Collection:
Q4	Is the data collecting and measurement process sufficiently explained?
Q5	Do the constructs and measures represent the best appropriate methodologies for answering the research question/issue?
	Data Analysis:
Q6	Is the data analysis used appropriately explained?
Q7a	Is the proof explanation given clearly? (Qualitative study)
Q7b	Has the data's relevance been assessed? (Quantitative study)
Q8	Is it clear how machine learning algorithm function and how they're implemented?
	Conclusion:
Q9	Are the study's findings reported and supported by the data?
Q10	Has the study's validity or limitations been discussed?

3) VALIDITY THREAT

Using keywords and terminology relevant to malware detection, the selected papers that were evaluated in the literature review will be obtained. The obtained studies will then be manually filtered using selection criteria. However, there is a chance that the studies chosen do not accurately reflect the research. Thus, four common types of validity threats have been considered: internal validity, external validity, constructing validity, and conclusion validity.

a: INTERNAL VALIDITY

To conduct the automatic search, there are six online databases utilized. However, they still covered some top-ranking journals with high-impact factors. Furthermore, a snowballing strategy was used for the manual search to limit the risk of missing necessary research and ensure that the paper selection process was fair.

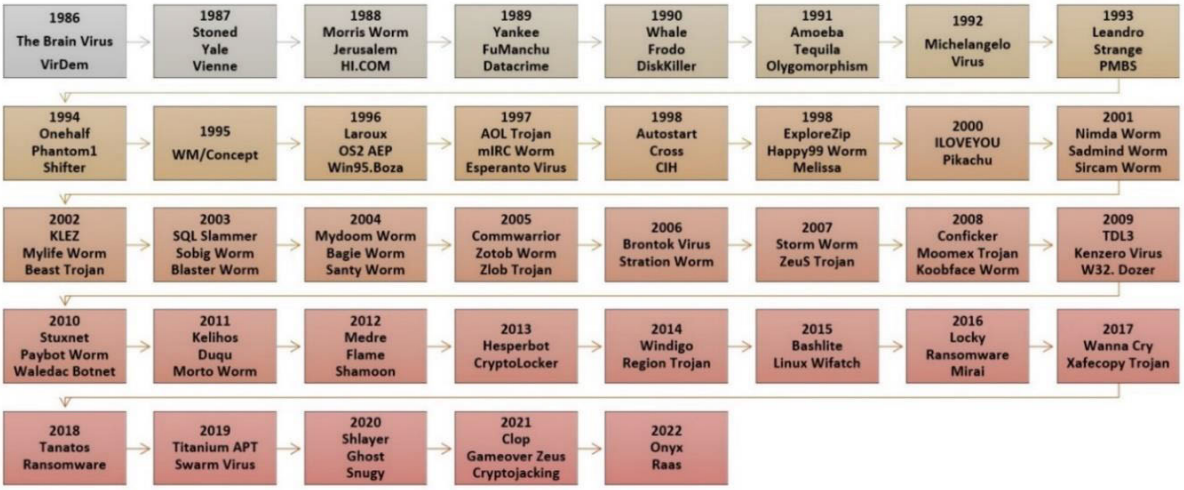

FIGURE 2. Evolution of malware from 1986 to 2022.

b: EXTERNAL VALIDITY

The validity threat was minimized to generalize the research's findings by finding published papers between 2017 and 2021. The number of documents collected for this SLR increased in parallel with the number of research papers produced each year in the related areas, which indicates that this SLR might remain a generalized report according to the research's external validity requirement.

c: CONSTRUCTING VALIDITY

Constructing a validity process can be ensured through automated and manual searches to obtain the gathered data. In addition to the present study objectives, article inspection, quality evaluation, inclusion, and exclusion criteria were utilized to restrict the possibility of validity threat.

d: CONCLUSION VALIDITY

Based on the guidelines adopted from different authors [13], [14], conclusion validity was managed by implementing the techniques. Thus, there is a possibility to repeat each procedure in this SLR, and the same results can still be produced.

III. BACKGROUND

This section briefly introduces malware attacks and machine learning algorithms, including their definition and evolution over the years. Besides, this section compares the machine learning algorithm with other malware detection techniques to highlight its advantages. Grasp, why machine learning has grown as a possible answer for future research direction requires a basic understanding of malware's history and how it was created.

A. MALWARE

Malware (short for ''malicious software'') is designed to obtain unauthorized access to our systems. It can slow down your computer's performance and an internet connection, steal or gather sensitive data, access private computer systems, transmit spam from your computer, attack other computers, and upload your files to criminal entities. Its definition is continually expanding since new exploits continue to evolve. Furthermore, malware threats continue to grow by volumes, categories, and features caused by the opportunities offered by technological advances. IoT devices, smart devices, social networks, internet connections, smartphones, etc., allow malware to create smart, sophisticated, and more advanced malware.

If we look at the history of malware, it can be divided into five categories [15]. The early variation of malware is when the first malware comes to life. This is considered the first category of malware. The second category of malware is when Windows come to ease our daily tasks. It is described as the first malware that attacks Windows, including mail worms and macro worms. The third category of malware is the evolution of worms that attacks our network. This malware has become famous as the internet has become widely used. The fourth category of malware is when rootkits and ransomwares take over the digital world to attack our computer system. This malware was the most dangerous malware before 2010. Then, the fifth category of malware came as virtual espionage and sabotage, where the secret services of some countries created this malware. Other than those five malware categories, currently, there is more advanced malware for the modern era with artificial intelligence [16] technology. FIGURE 2 shows the evolution of malware from 1986 to 2022 [16], [17] with the top highlighted malware.

1) MALWARE TYPES

The malware appears in many names and variations [18], as seen in FIGURE 2, while the description of each malware with its threat strategies [122] is represented in APPENDIX A, TABLE 11. Meanwhile, TABLE 6 shows the various type of malware which recently found and can be

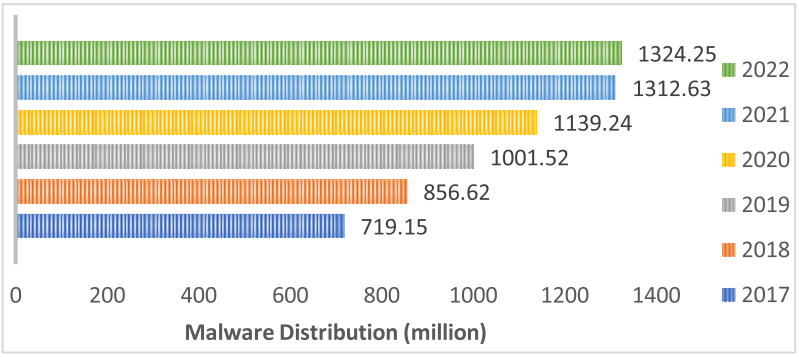

FIGURE 3. Total distribution of malware from 2017 to 2022.

TABLE 6. Type of malware with description.

Type	Description
Virus	A virus is a piece of code that may duplicate itself and spread to other programs on the system. Viruses often propagate by attaching themselves to a variety of applications, then executing code whenever a user runs a particular program. Viruses spread across the system through documents, script files, and web application vulnerabilities. [18]
Worms	A worm is a computer program that can replicate itself and spread from device to device over a network without human involvement. Worms can carry "payloads," which use bandwidth and cause congestion on web servers to harm the host device and ruin host networks. [18]
Trojans	A Trojan is a form of malware that disguises itself as an innocuous program to entice users to download and install malware. This form of malware allows attackers to get remote access to steal data, and money, remove and edit files, develop malware versions, and monitor user actions such as monitoring screens and logs, among other things. [18]
Spyware	A sort of virus that monitors user activity without the user's knowledge or agreement. Collecting key logging, screen monitoring, and stealing account information are examples of their actions. Spyware causes network settings to be disrupted by altering security processes. Spyware hides in legitimate software or Trojans to take advantage of flaws. Examples of smartphone spyware programs such as Acallno and FlaxiSpy. [18]
Botnets	A bot is a software application that allows an attacker to get remote access and control over the infected device's activities without the user's consent. Bots are part of botnets, which are a collection of computers controlled by a botmaster. Bonnets launch distributed denial of service (DDoS) attacks, web spiders that hack server data, malware masquerading as well-known sites, and spam bots that collect information, making it a serious security threat. [18]
Ransomware	A kind of malware that holds computer resources hostage until the victim pays a ransom. Ransomware locks down a computer, restrict access, encrypts files, and displays messages to push users to pay money. The ransomware malware will unlock the machine after payment. [18]
Adware	It is ad-supported malware that is specially intended to serve advertisements to users on an ad-hoc basis. Adware is software that displays adverts and pop-up ads on websites. Adware is typically delivered for free, with advertising corporations sponsoring it and generating cash in some circumstances. Adware is solely meant to offer an ad; when a user clicks on an ad, adware activates and steals information or records user actions. [18]
Rootkits	A rootkit is a form of malware that exploits consumers by gaining remote access and controlling a device. A rootkit comprises a dropper, loader, and rootkit to perform destructive acts. It acquires administrator access to carry out harmful actions such as stealing data, disrupting the system's regular operations, making changes to the system, causing system configuration changes, etc. When a rootkit is placed on a computer, it runs every time the machine boots up. [18]
Keylogger	As users write on the system, the virus records everything to collect their log-in data and other sensitive information, which it then sends to the key-logging application. Many groups commonly use keyloggers to acquire information about computer activity. [18]
Backdoor	Backdoors are a kind of malware that opens a backdoor onto a device in order to provide the groundwork for subsequent infections. It assists other malicious operations by providing a network connection via which they may enter and snip information. [18]

used to classify each malware. These malware variants are not distinct from one another, and a single malware variant might develop multiple new characteristics simultaneously. As a result, malware is one of the most severe digital threats to cyber security. According to the McAfee Labs [19] report, the average number of malware attacks per minute was 588 in the third quarter of 2020 and grew to 648 in the fourth quarter of 2020.

Meanwhile, AV-TEST institute [20] reported that when this paper was written, over 450000 potentially unwanted applications and new malware were registered every day. Furthermore, for the last six years, malware increased significantly from 2017, with 719.15 million to 1324.25 million malware in 2022, as shown in FIGURE 3. This scenario indicates that further action should be taken to curb malware attacks.

2) MALWARE BEHAVIOR

Ilker kara [21] investigates malware's capacity to see the damage it can do to the target system, recover from that damage, and, if possible, detect specific information about the attacker. He suggested a method for analyzing malware, including behavior, memory, and code analysis using digital

material and an actual malware attack. It was shown that malware might be tracked by looking at the Whois information of the server to which it connects and studying its typical behavior. Therefore, understanding the malware's behavior, including camouflage and obfuscation techniques [22], might help researchers plan and generate an efficient algorithm for malware detection.

Camouflage [22] approach refers to concealing malware for as long as possible to avoid detection by malware detectors. Malware developers employ various strategies, ranging from simple techniques like encryption to more complicated techniques like oligomorphic, polymorphic, and advanced ones like metamorphic.

Encryption is used when malware programmers want their malware to go unnoticed and undetectable by malware detectors [22]. Encryption is the most basic method 2 of concealment that they employ. It is the first approach for malware concealing [23]. It is made up of two modules: encryption and decryption. Encryption is performed with a separate key, whereas decryption is performed with the same key. Their detection is feasible since the decryption mechanism does not offer uniqueness.

Oligomorphic virus, Whale was the first of its kind to arrive in 1990. It turned out to be a DOS virus [22]. This is seen as a step forward in virus concealment and semi-polymorphic encryption. The decryptor, like encryption, remains the same for each infection, whereas oligomorphic uses a different decryptor for each condition. Even though oligomorphic selects different decryptors for each new attack, antivirus may still detect it by inspecting all of the decryptors.

Polymorphic was first used in the Polymorphic virus, 1260, which Mark Washburn created in 1990 [22]. Polymorphic viruses are more sophisticated than ordinary viruses because they combine encryption and oligomorphic. Antiviruses are difficult to identify since each copy has a different look. They have no limit on the number of decryptors they may create. To disguise itself, this virus employs a variety of obfuscation techniques. A mutation engine carries out this technique of alteration.

Metamorphism is not incorporated in encryption; instead, the malware's content changes [22]. As a result, a decryptor isn't required. In 1998, the first metamorphic virus, ACG, was created for DOS. It also uses polymorphism similar to that of a mutation engine, but it changes the entire body instead of just changing the decryption. The core concept is that the syntax changes with each new copy while the semantics remain the same, i.e., the visible virus changes with each infection. Still, the meaning or functionality remains the same.

Obfuscation [22], on the other hand, is a method used by malware programmers to make malware challenging to read and interpret. The main goal of this technique is to conceal malware's destructive activities. Various researchers classify obfuscation techniques [124] in different ways. The six most prevalent obfuscation techniques are dead code insertion, instruction replacement, register reassignment, subroutine reordering, code transposition, and code integration [23].

Dead Code Insertion is the simplest method for changing code without changing its meaning [22]. By using NOP instructions and pushing, followed by popping, garbage code or statements are added to the code. These statements are utilized so that the code semantics are unaffected.

Instruction Replacement is replacing the existing instruction with comparable instructions, making detection difficult. Like synonyms in natural languages, this technique substitutes instructions with others that create the same meaning [22].

Register Reassignment re-assigns the register in each copy without affecting the virus's semantics [22]. It is the most basic approach but may be difficult to detect when used in conjunction with other methods.

Subroutine Reordering is permuting a series of instructions in a piece of code so that the look of the code changes, but the behavior remains the same [22].

Code Transposition is where the original code's sequence of instructions is reorganized so that its meanings do not change [24]. Code transposition can be accomplished in two ways. The first way is to reorder the instructions at random to retrieve the original code. The second way is more difficult to adopt than the first, but it is far more effective. The unconditional statements and jumps are employed in one way, while the instructions independent of the other are picked and reordered in another.

Code Integration is where the malicious code is integrated or embedded within the software that needs to be affected. This is a viable approach in which the original software is disassembled, and malicious code is inserted, making it difficult to detect [25].

3) THE STAGES OF MALWARE ATTACK

The phases of a malware attack aren't always the same, but they follow a similar pattern every time one is launched. One of the examples of a malware attack lifecycle [26] can be seen in FIGURE 4. The malware attack starts with the entry point stage, where potential targets will be identified and discover any defenses that have been implemented. Then, the most suitable attack method will be set. The next stage is breaking in, where malware bypasses the perimeter defenses and accesses the intended attack area. Then, malware will start its activities of command and control. Once they have established a connection to the deliberate attack area, the infection stage will be implemented. Finally, the execution stage will be considered to profit and fulfill their attack objectives, including stealing sensitive data, corrupting the critical system, and disrupting the operations of the intended targeted business.

B. MACHINE LEARNING

In 1956, a group of computer scientists suggested that computers might be programmed to think and reason so that

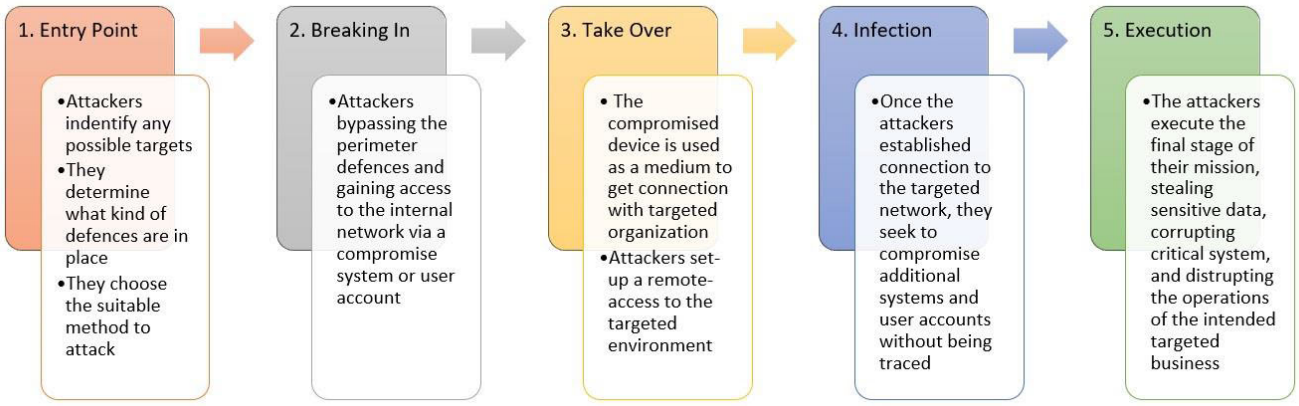

FIGURE 4. Five stages of malware attack.

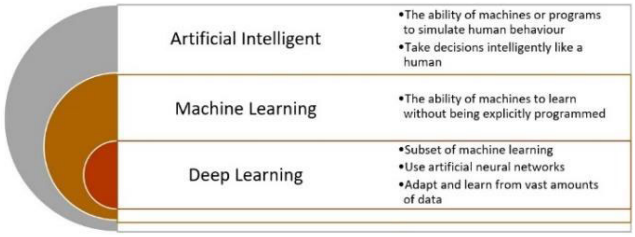

FIGURE 5. The connection of AI, ML and DL.

any part of the learning or any other aspect of intelligence could theoretically be specified so precisely that a machine could imitate it. Their idea is known as artificial intelligence (AI) [27]. AI is a field focused on automating intellectual processes that humans typically handle. Machine learning (ML) and deep learning (DL) are specialized ways to reach this aim [28]. FIGURE 5 depicts AI, machine learning, and deep learning.

Machine learning has positively impacted real-world problem-solving from 1950 to 2021 [29], as highlighted in FIGURE 6. With each sort of learning, there are success stories of firms that have made significant progress and added value to their businesses. Each kind of machine learning offers a strategic and competitive advantage, but the availability of high-quality data is considerably more important than the approach employed. It's important to understand the different types of machine learning algorithms and when to employ them. Any machine learning task, and everything else being done in the area, aims to break down a real problem into design forms for machine learning systems. Understanding the different types of machine learning algorithms and when to utilize them is essential.

1) MACHINE LEARNING ALGORITHM

It's challenging to design a general system that allows for the efficient distribution of regular ML since each method has its communication pattern [30]. As indicated in FIGURE 7, the challenge of machine learning can be divided into two parts: training and prediction. The training process encompasses feeding a large body of training data to a machine learning model and updating it with an ML algorithm. The learned model is implemented in practice during the prediction phase. The trained model takes raw data as input and produces predictions as output. While the model's training phase is often computationally costly and requires substantial data sets, the inference step may be done with minimal processing resources. The phases of training and prediction are not mutually exclusive. Incremental learning combines the training and inference stages and uses new data from the prediction, phase to constantly train the model.

Every effective machine learning algorithm requires a mechanism that drives the system to increase its accuracy by forcing it to improve itself based on new input data. The most frequent algorithms for a range of ML models, which can be implemented for malware detection, are listed in the following.

Support Vector Machine (SVM) [31] assembles a hyperplane or group of hyperplanes for configuration in a high or unbounded dimensional space. When everything is said and done, the hyperplane uses a sensible partition with the biggest partition to the nearest getting ready data motive behind any class (called useful edge), since a bigger edge means a smaller characterizer speculation error.

K-Nearest Neighbour (KNN) [32] is generally used for classification approaches. Still, many terminally argue that in malware detection, its assessment is also based on "easy of interpretation output, computation speed, and prediction capacity," according to our study. KNN is possible to use to help with planning and relapse concerns. However, it is utilized to categorize malware in our problem set because k prepares models or instances in the majority regarding the input, i.e., which class it is closely related with. The information can only connect to one of two classes: whether or not malware has been discovered.

Naïve Bayes (NB) [33] is an algorithm for analyzing variables' connection using an estimator classification technique. The NB classification uses a series of computations based on

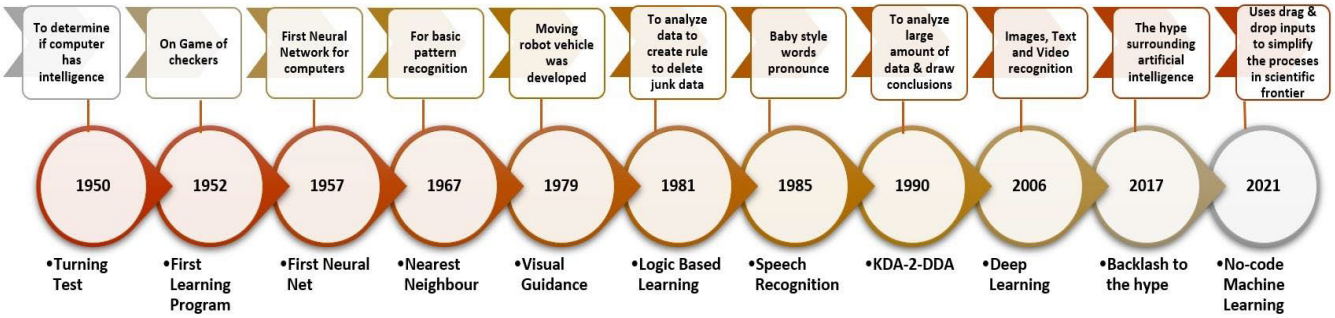

FIGURE 6. Evolution of machine learning techniques from 1950 to 2021.

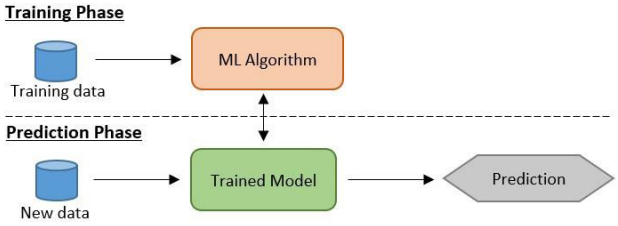

FIGURE 7. General phases in ML algorithm.

probability concepts to identify the class or category of data supplied to the system. The system is given a specific set of data in the NB categorization. For the training stage, a class of data must be supplied. New test data provided to the system is generated using the previously acquired probability values. It is attempted to determine which category the given test data belongs to using probability operations on the trained data.

Decision Tree (DT) [34] is a root hub; branches and leaf hubs are all grouped in this configuration. Each hub represents a test quality, each branch represents a test result, and each leaf hub represents a class name. DT does not need any area learning, but it uses the concept of data entropy, which is simple to grasp, and the choice tree's learning and characterization phases are simple and fast.

The **N-gram** is a valuable tool in the field of Natural Language Processing. N-grams [35] can learn binary code and code region information, which correlate to greater entropy levels. The strategy relies on statistical learning and isn't entirely reliant on specific viruses. The N-gram approach divides a given text or audio sequence into N different size combinations. When N is 1, 2, or 3, it is sometimes referred to as a ''unigram,'' ''bi-gram,'' or ''trigram.''. N-gram considers what comes before and after the words to capture the most critical attribute.

Bayesian [36] algorithm is a distributed function that integrates classification and characteristics and computes the joint probability of the training set to estimate sample classification. The Bayesian model is based on classical mathematics theory and offers a high level of classification accuracy. However, because the posterior probability is determined by determining the preliminary data, the classification decision has a specific mistake rate, and the procedure is sensitive to the input data expression form.

Gaussian [37] applies Lazy learning, and Laplace approximation is used in this procedure, which implies data generalization is deferred until a query is performed, as opposed to eager learning, which generalizes training data before a query is made to the system. It's prolonged, but it gets the job done.

Meta-Heuristic [38] algorithm is a self-learning method for solving complex optimization problems up to the optimal solution. An accurate optimization approach cannot tackle several real-world optimization issues. Heuristics and metaheuristics are two types of approximate algorithms that are used to address such problems. Heuristics algorithms are problem-specific and based on experience, whereas metaheuristics algorithms serve as a foundation for optimization and guide heuristics design.

Neuro-Fuzzy (NF) [39] when systems combine the benefits of fuzzy logic and artificial neural networks, neural networks' potential is expanded. The growing neuro-fuzzy systems combine a neural network's adaptive and evolving learning power with the estimated reasoning and substantial interpretation of fuzzy rules. These are recent breakthroughs in neuro-fuzzy approaches. The capacity of the rule base to evolve with adaptive parameters is a crucial feature of developing neuro-fuzzy systems.

K-Means [40] is a cluster analysis approach in which the defined 'k' separates the clusters, and all the grouped items share a center value. However, the K-Means clustering technique isolates the temporal period between normal and anomalous data in the same training dataset from a data mining standpoint. As a result, the clustering approach groups objects based on their data point characteristics. Each data point in a cluster is identical to those in the same cluster but distinct from those in other clusters.

Meanwhile, malware detection was accomplished by categorizing via different machine learning algorithms, as described in the previous paragraphs. According to research conducted in system calls on Android [4], the Random Forest approach could offer the most outstanding

accuracy value of 76%. Compared to other approaches, the True Positive Rate (TPR) is 76%, while the False Positive Rate (FPR) is 13.3%. However, the KNN approach has the fastest or least minimum calculation time, followed by Random Forest and Naive Bayes. On the other hand, log regression takes the longest to compute, followed by SVM and DT. This happens because the three approaches have more parameters, corresponding to a longer computation time. Furthermore, based on these findings, it can be concluded that high recall values will follow high accuracy results but lower precision numbers.

On the other hand, various machine learning algorithms [41] are utilized to identify the app as benign or malicious. Various performance metrics are used to evaluate each algorithm to determine which ones are best for detecting harmful software. The results demonstrate that Random Forest delivers the most significant result, with an accuracy of 90.63%, making it the most successful malware detection tool. Regarding the area under the operating curve (AUROC), the support vector machines (SVM) are second best and perform well in other areas. It has a lower False Positive Rate than Random Forest. Meanwhile, while Nave Bayes has the best (lowest) False Positive Rate, it has a poor True Positive Rate and poorly in other criteria.

Furthermore, to overcome the difficulties faced by conventional methods to detect unknown and zero-day Android malware apps, an empirical study and performance comparison [42] of six supervised machine learning algorithms, including KNN, Decision Tree, SVM, Random Forest, Nave Bayes, and Logistic Regression, which are commonly used in the literature for detecting malware applications, was conducted. The results of the experiments revealed that all six machine-learning algorithms performed well in detecting Android malware. Random Forest, for instance, had the highest detection accuracy of 99%, while Nave Bayes had the lowest detection accuracy of 95.59% in detecting Android malware.

IV. TAXONOMY

This section discusses each result received from the SLR, including the machine learning algorithm used, how the algorithm works, the performance result, the classification method, and the selected analysis type employed to answer RQ1 through RQ5.

A. CLASSIFICATION OF MACHINE LEARNING

Machine Learning is divided into four categories, as indicated in FIGURE 8 [27], based on the nature of the learning and learning system, including unsupervised, supervised, semi-supervised, and reinforcement learning.

Unsupervised learning is machine learning that searches a data set for previously uncovered patterns with no pre-existing labels and minimal human supervision [27]. Clustering and Dimensionality Reduction are the two basic unsupervised learning methods, while an example of an algorithm for clustering is K-Means.

Supervised learning is learning a function that maps an input to an output using machine learning based on sample input-output pairs [27]. It uses labeled training data and training examples to infer a function. Classification and regression are the two main supervised learning techniques. Meanwhile, KNN, SVM, DT, RF, and NB are classification methods, while linear and logistic regression is examples of regression methods.

Semi-supervised learning is a machine learning method that combines the benefits of both supervised and unsupervised learning [27]. A semi-supervised learning strategy comes in handy when we only have a small amount of labeled data but a vast number of unlabeled data to train with. The small amount of label data can be exploited using supervised learning characteristics. On the other hand, unsupervised learning characteristics can let you take advantage of a large amount of unlabeled data.

Reinforcement learning is one of the most common machine learning techniques to determine the best agent actions to maximize reward in each environment [27]. The agent learns to refine its activities to maximize the total reward. Agent, environment, action, and reward are the four main components of reinforcement learning.

Agent is a trainable program that performs the duties given to it.

Environment is the physical or virtual environment in which the agent performs its tasks.

Action is a change in status in the environment that occurs when an agent moves.

Reward is the action that determines whether a negative or positive recompense is given.

B. MALWARE DETECTION

Malware is a global issue, and malware detection tools are the first line of protection against it. The approaches that a malware detection tool employs determine its effectiveness. For malware detection, various mechanisms exist, such as Data Mining [43], Deep Learning [44], Hypothesis Exploration [45], and so on. However, one of the most well-known methods for detecting malware is the Machine Learning algorithm (MLA).

Ten machine learning algorithms for malware detection were discovered, as shown in FIGURE 9, based on the analysis of 77 selected studies using the SLR technique to assess their performance in detecting malware. We found that SVM is the most widespread malware detection algorithm, with 24%, followed by DT, with a percentage of 15%. N-grams and Naïve Bayes were almost equivalent distribution with 14% and 12%, respectively. Besides, KNN, Bayesian, and K-Means have 10%, 8%, and 6%, respectively. Gaussian and NF contributed the same portion of 5%, while Meta-heuristic is the least contribution with a percentage of 1%.

SVM [119], DT [88], and N-grams [96] have the highest detection accuracy rate, at 100%, while NB [81] has the

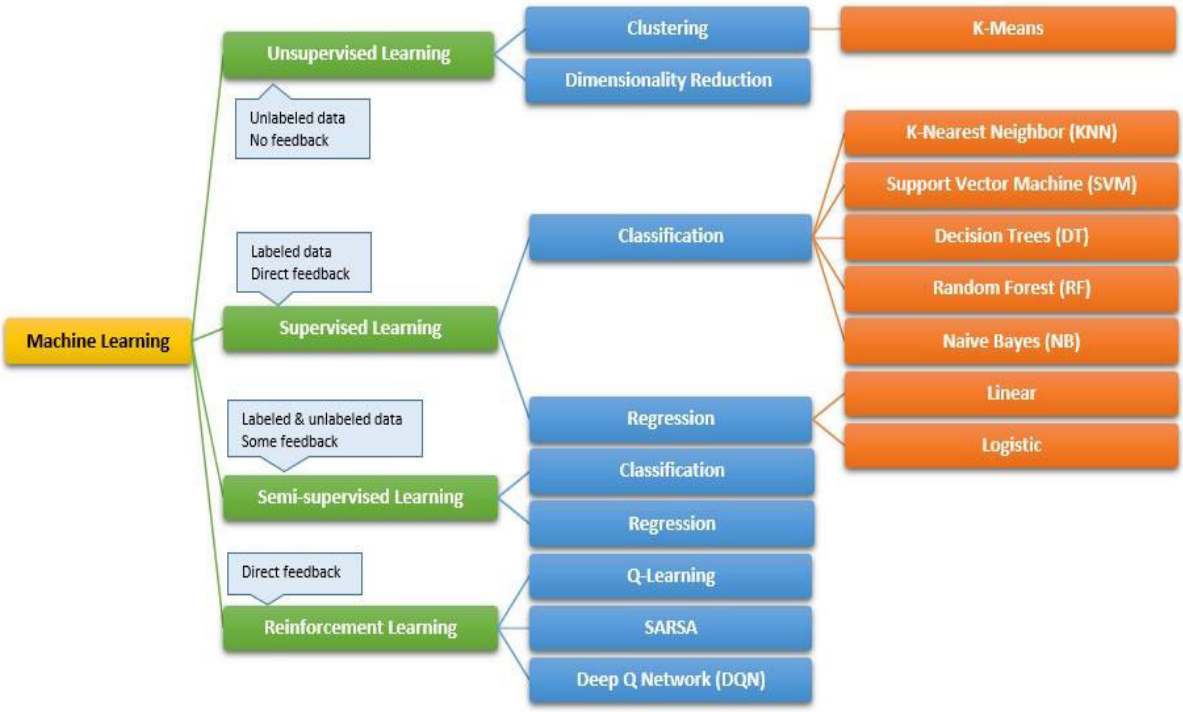

FIGURE 8. Taxonomy of machine learning techniques.

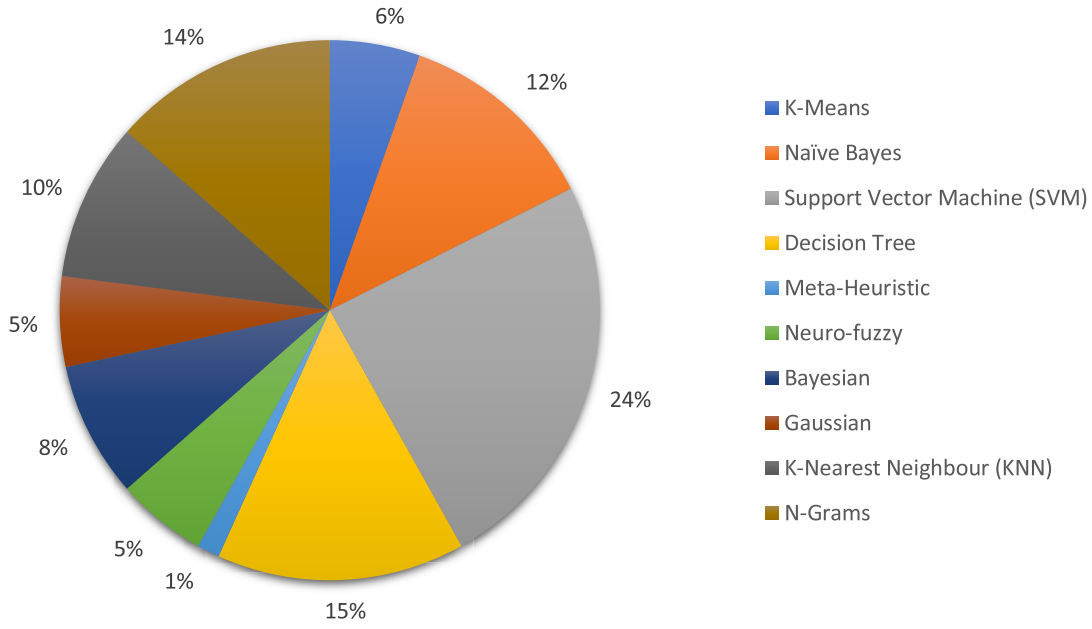

FIGURE 9. Distribution of MLA.

lowest detection accuracy rate, at 64.7%. However, only a limited dataset was used to examine the performance of DT and, N-grams for malware detection. Therefore, there is a risk of biased analysis because not all attributes may have been incorporated due to the limited number of samples; as suggested by the authors, future research will require a larger dataset. TABLE 7 shows a summary of malware detection accuracy rates.

Meanwhile, each algorithm's average detection accuracy rate has been obtained, and SVM continues to perform well, with a 90.55% accuracy rate. N-grams have the greatest average detection accuracy rate of 97.80%, followed by KNN 92.72%, DT 92.23%, K-Means 89%, Bayesian 89.08%, Gaussian 87.42%, NB 86.45%, NF 83.48%, and Meta-Heuristic with 81.23%. FIGURE 10 shows the details of the average detection accuracy rate.

TABLE 7. The following is a summary of the SLR-based malware detection accuracy rate.

	Related Works	Range of Detection Accuracy Rate
K-Means	[51], [52]	88% - > 90%
NB	[29], [38], [39], [40], [4], [52], [55], [56], [57], [58], [37], [36],[79], [86], [92], [95]	73.01% - 98%
SVM	[36], [39], [40], [4], [43], [44], [45], [46], [49], [53], [37], [61], [62], [71], [40], [74], [75], [76], [77], [78], [79], [80], [81], [89], [90], [33], [94], [95]	64.7% - 100%
DT	[29], [33], [34], [35], [36], [4], [46], [48], [49], [58], [65], [66], [69], [70], [79], [81], [84], [85], [86], [92], [94]	78.62% - 100%
Meta-Heuristic	[30]	81.23% - 99.91%
NF	[82], [87], [88]	69.44% - 91%
Bayesian	[32], [47], [63], [64], [83]	80% - > 97%
Gaussian	[32], [36], [50], [67], [89]	80% - > 91.1%
KNN	[28], [29], [37], [46], [49], [53], [58], [66], [68], [69], [70], [36], [80], [91], [33], [94]	80.50% - 99.2%
N-grams	[30], [31], [41], [42], [27], [54], [59], [60], [72], [73], [93]	81.23% - 100%

1) MALWARE DETECTION PROCESS

The overall process of malware detection can be seen in FIGURE 11. Malware detection can be broken down into two stages: malware analysis and malware detection. The malware analysis focuses on gathering data from previously identified malware to generate and extract its features. Following that, an algorithm will be created based on those features. Malware analysis approaches also assist analysts in comprehending the risks and intents connected with a malicious code sample. The knowledge gained can react to new malware development patterns or take preventative measures to deal with future threats. Furthermore, unknown malware can be grouped into existing families using features gained from malware analysis.

On the other hand, the malware detection phase use malware detector 'D' specified as a function whose scope and range are the set of executable program 'P' and the set {malicious, benign} [46]. A malware detector, in other words, can be defined as indicated below.

$$D(p) = \begin{cases} \text{malicious} & \text{if p contains malicious code} \\ \text{benign} & \text{otherwise} \end{cases}$$

The detector examines the program 'p' ε P to determine whether it is benign or malicious. Testing aims to determine the percentage of false positives, false negatives, and hit ratio. The malware is detected by the malware detector using malware signatures. A signature is a binary pattern in the machine code of a particular malware. Anti-malware technologies compare their malware signature database to files on the hard disc, removable media (including boot sectors), and RAM. The anti-malware provider routinely updates the signatures and makes them accessible to clients via the Web.

False Positive occurs when a malware scanner discovers 'malware' in a non-infected file [46]. False positives result when the signature used to detect a specific infection is not unique to the malware and appears in legitimate, non-infected software.

False Negative when a malware scanner unsuccessfully detects malware in a compromised file [46]. Due to new malware and the lack of a signature, the anti-malware scanner may fail to detect malware because of configuration settings or even erroneous signatures.

Hit ratio occurs when a malware detector identifies the malware [44]. This happened because the malware signature matches the signatures contained in signature databases. The formula is shown below.

$$\text{Hit ratio} = \text{D(p)}/\text{Number of detected malware}$$

2) MALWARE DETECTION TECHNIQUES

Malware detection techniques are classified into four types: signature-based, behavior-based, heuristic-based, and specification-based. These approaches are used to identify and detect malware and countermeasures against it to protect computer systems from data and resource loss.

a: SIGNATURE-BASED

Most antivirus applications use signature-based detection techniques. The antivirus program disassembles the code of the infected file and searches for a malware family pattern [18]. A sequence of bits known as a signature is embedded in the code when malware is produced, which can be used to determine which malware family it belongs to [46]. Meanwhile, malware signatures are stored in a database and compared during detection. This kind of detection is sometimes called string, pattern scanning, or pattern matching. It might be static, dynamic, or a combination of the two, called a hybrid.

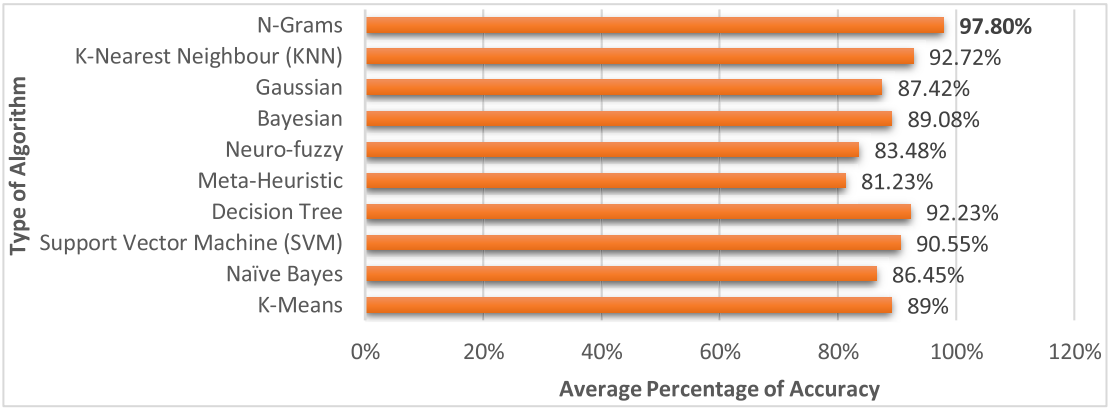

FIGURE 10. Average of detection accuracy rate.

b: BEHAVIOR-BASED

Behavior-based detection is conducted based on malware behavior [50]. A behavior-based technique is used to overcome the limitation of the signature-based technique. The main advantage of this technique is that zero-day malware can be detected. However, if all malware scenarios are not thoroughly investigated, this technique can result in many false positives.

c: HEURISTIC-BASED

Heuristic-based detection detects or distinguishes between a system's normal and unusual activity, allowing for the identification and resolution of known and undiscovered malware attacks [46]. There are two steps to the heuristic-based detection method. In the first step, the system's behavior is watched in the absence of an attack, and a record of vital information is kept that may be confirmed and checked in the event of an attack. In the second step, this difference is monitored to detect malware from certain family. The behavior detector employed in the heuristic-based technique, as illustrated in FIGURE 12, consists of three fundamental components: data collection, interpretation, and a matching algorithm. The behavior detector [22] is depicted in FIGURE 12, describing how these components interact.

Data collection is concerned with the collection of either static or dynamic data.

Interpretation will analyze and convert data from the data collection component into an intermediate format.

Matching algorithm is where the behavior signature will be compared to the converted data in the interpretation component.

d: SPECIFICATION-BASED

Specification-based detection approaches, in which applications are monitored and examined for normal and deviant behavior [44] in accordance with their specifications. The main difference between specification-based and heuristic-based detection is that heuristic-based detection techniques use machine learning and artificial intelligence methods to detect a legitimate program's valid and invalid activity. In contrast, specification-based detection is based on analyzing the behavior described in the system specification. This method is essentially a manual comparison of some systems' typical actions. Lowering the

false positive rate and raising the false negative rate overcomes the limitations of heuristic-based approaches.

According to the SLR results, as seen in TABLE 8, most studies with a percentage of 48.5% use behavior-based classification methods, including two studies that used DT [88] and SVM [119] that achieved 100% accuracy rate in detecting the malware. On the other hand, signature-based contributed 43.6%, followed by permission-based and images-based, with 5.9% and 2.0%, respectively. It shows that the behavior-based classification method is more relevant and effective in detecting malware. The comparative studies for classification methods are represented in Appendix A, TABLE 13.

3) MALWARE DETECTION ANALYSIS

Malware analysis is the first step in detecting malware [47]. To identify malware, we must first understand how it works and why it was created so malware detector developers can easily integrate protective capabilities. Based on the time and technique used to perform the analysis, malware analysis techniques are classified into static, dynamic, or hybrid.

a: STATIC ANALYSIS

Static analysis, often known as code analysis [48], is the process of analyzing software or a piece of code without running it. Static information is collected from the code to assess whether the software contains harmful code. The malware is reverse-engineered using various tools, and the malicious code's structure is evaluated to determine how it operates. Debuggers, dissemblers, de-compilers, and source code analyzers are some of the tools used to perform static analysis. Meanwhile, File Format Inspection, String Extraction, Fingerprinting, AV scanning, and Disassembly are some of the methods utilized in static analysis.

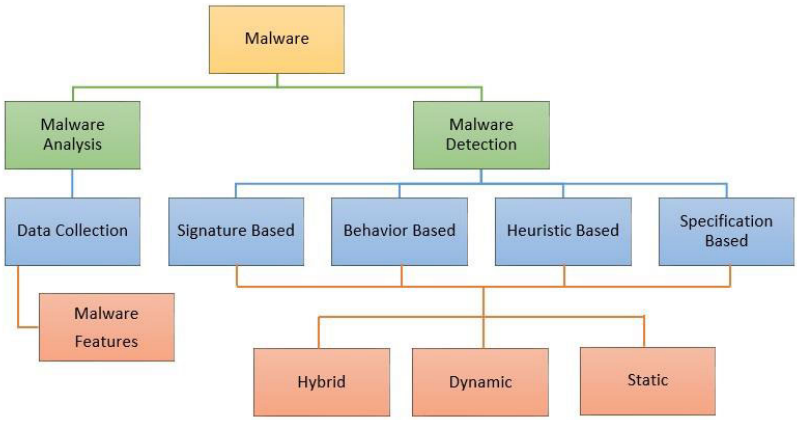

FIGURE 11. The overall process of malware detection.

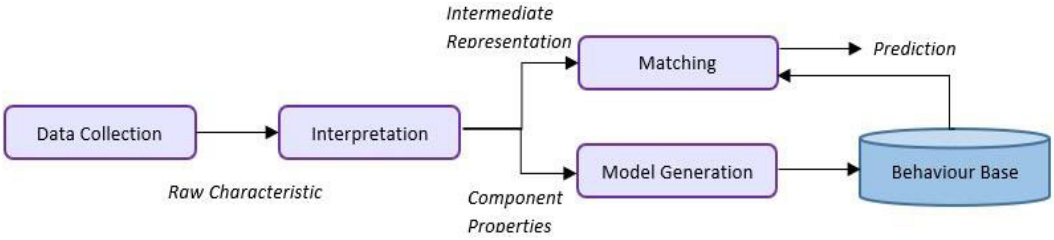

FIGURE 12. The overall process of malware detection.

TABLE 8. Distribution of MLA based on the classification method use.

No	Reference	Classification Method	Number of studies									
			KNN	DT	NB	SVM	N-grams	K-Means	Gaussian	Bayesian	NF	Meta-heuristic
1	[73], [84], [93], [117]	Image-based	2	1	0	2	0	0	1	0	0	0
2	[33]	Permission-based	1	0	1	0	0	0	0	0	0	0
3	[51], [53], [56], [57], [59], [60], [62], [65], [69], [70], [36], [71], [40], [74], [75], [77], [81], [86], [87], [88], [90], [92], [97], [98], [100], [101], [102], [104], [105], [106], [108], [39], [110], [111], [39], [113], [115], [116], [119]	Behavior-based	6	10	8	12	3	2	1	3	3	1
4	[52], [54], [55], [58], [37], [61], [63], [64], [66], [67], [68], [72], [76], [78], [79], [80], [82], [83], [85], [89], [91], [94], [95], [96], [99], [103], [107], [112], [114], [118]	Signature-based	6	7	6	12	8	0	3	2	0	0

b: DYNAMIC ANALYSIS

Dynamic analysis, also known as behavioral analysis [49], analyzes and observes malware functioning while it is being executed. This analysis will examine the function calls and control flows and evaluate the instructions and parameters. Malicious codes are executed in a simulated environment to observe their behavior and countermeasures can be developed. Sandbox, simulator, emulators, RegShot, and Process

Explorer, are some of the tools used for dynamic analysis. Moreover, dynamic analysis outperforms static analysis since the infected software is run in a virtual environment for monitoring. As a result, this technique can detect a wide range of malware. However, this analysis takes longer to be conducted because we need to create an environment in which malicious software may be executed and tested.

c: HYBRID ANALYSIS

Hybrid analysis blends static and dynamic analysis techniques to gain the advantages of both techniques. The malware is first examined using code analysis and a malware signature check. The malware will then be launched in a simulated environment to observe its actual behavior.

Meanwhile, based on the SLR results, as seen in TABLE 9, most of the studies with a percentage of 53.3% use static analysis, which is also used in the study of N-grams [96], achieving 100% accuracy rate to detect malware. However, the other two studies used dynamic analysis, which achieved a 100% accuracy rate in detecting malware, DT [88] and SVM [119]. We found that dynamic analysis only contributed 28.9%, followed by hybrid analysis with a percentage of 17.8%. Even though static analysis is popular among researchers, it is also contributed to one of the limitations [70] in detecting malware. Thus, the effectiveness of static analysis requires further research work. Details of the comparative studies for analysis type are represented in Appendix A, TABLE 13.

4) CLASSIFICATION BY DATASET

A dataset is one of the crucial elements in conducting any experiments for malware detection. The most prevalent datasets are DREBIN and Android Malware Genome Project, according to the earlier SLR [2], which employed machine learning using hybrid analysis for android malware detection. Even though DREBIN is a popular dataset among researchers, the samples collected were from August 2010 to October 2012, which is quite outdated. However, it is still relevant to achieve some results. Furthermore, most researchers obtain innocuous apps from Google Play and local app stores. Also, ContagioDump, VirusTotal, and VirusShare were also employed for malware samples. However, in this SLR study, VirusShare [56], [60], [65], [72], [81], [92], [96], [101], [110] [111], [113], is found as the most popular dataset used in their experiments, followed by DREBIN, [59], [61] [64], [69], [75], [82], [83], [102], [103], Malware Genome Project, [36], [39], [61], [106], [108], [109], Google Play Store, [61], [80], [108], [109] and many more type of [36], datasets as shown in TABLE 14 in Appendix A.

V. CURRENT CHALLENGES

To answer RQ6, this section is one of the answers which summarizes the current challenges and limitations found in the selected studies. The following are the common issues that require further action as the future direction of the research study.

A. DATASET USED

Dataset issues were found as the most common research gap in the selected studies. It is not only because the size of the dataset is small [57], [60], [69]; the lack of a standard dataset benchmark [56] also contributed to this issue. Besides, various types of datasets were used in the experiment leading to poor performance [37], [66]. The researchers also received an insufficient sample of data [40], and some of them had difficulties in finding suitable datasets [76], [96] for their experiments. Others, since the dataset is created from scratch with a minimum sample of files for analysis, bias is detected in producing the results [88]. Furthermore, the outdated dataset used in the experiment, which offers little or no utility as a benchmark for the performance of malware detection systems on a modern network [112], also contributed to this issue.

B. OBFUSCATED MALWARE

One of the challenging issues in detecting malware is related to the obfuscation technique described in the previous subsection about malware behavior. Modern stealthy malware attacks hide their behavior in virtual environments and security tools [53]. This technique will make the malware challenging to be detected. For instance, the current trends of botnets use the obfuscation technique to change their structure and the packet data [51] in the respective network environment. During the experiment, some malware behavior can also not be performed in the Android application process [115]. The packed code is a well-known method to obfuscate malware and make it difficult to detect [77]. Other than that, decompiling the APK with Dex2jar [80] is difficult since various obfuscation and feature-hiding techniques are challenging to manage.

C. IMPLEMENTATION TIME

Different kinds of algorithms might be used simultaneously in the empirical experiment; it takes longer to detect malware. Furthermore, implementation will take longer during the classification process [54]. Meanwhile, better effectiveness comes at the cost of poorer efficiency.

D. TYPE OF ANALYSIS

Since new malware samples constantly arise [83], the chosen analysis type, as described in the previous sub-section, might influence the performance of the detection accuracy rate. For instance, in some experiments, new malware instant or updated malware attacks can't be detected using static analysis [70]. The approach fails to detect some samples of malware like Pjapps and Geinimi. [61].

E. MALWARE FEATURES

The feature attributes chosen must be independent of one another or have a low correlation coefficient [36]. Besides, many issues arise because of the large number of unrelated or duplicate characteristics, including confusion about the

TABLE 9. Distribution of MLA based on the analysis type.

No.	Reference	Analysis Type	Number of studies									
			KNN	DT	NB	SVM	N-grams	K-Means	Gaussian	Bayesian	NF	Meta-heuristic
1	[51], [56], [57], [62], [65], [40], [74], [77], [81], [86], [90], [92], [97], [98], [100], [102], [105], [106], [109], [39], [113], [115], [116]	Dynamic	2	6	5	9	2	2	1	1	2	0
2	[53], [59], [60], [69], [36], [71], [87], [88], [101], [104], [107], [108], [110], [111], [119]	Hybrid	2	4	3	4	1	0	0	3	1	1
3	[52], [54], [55], [58], [37], [61], [63], [64], [66], [67], [68], [70], [72], [73], [75], [76], [78], [79], [80], [82], [83], [84], [85], [89], [91], [93], [33], [94], [95], [96], [99], [103], [112], [117][114], [118]	Static	12	9	7	16	8	0	4	1	0	0

learning algorithm, over-fitting, and reduced classification accuracy [39].

F. CLASSIFICATION METHOD

Minor software changes have a significant impact on the signature-based method. As a result, malware cannot be identified [73] if this method is used and the modified program codes are used. The classifiers have similar challenges when detecting new malware types. It can prevent some malware from acting differently in real-world situations [119]. Previous research has utilized various approaches, but none has looked at distance measures for classification [75]. Besides, distinguishing between malware families is a more challenging problem than a binary classification of malware and benign files [91]. Meanwhile, text classification studies [33] commonly meet the sparse matrix problem.

G. OTHER ISSUES

The high false positives rate [86] is one issue that needs further consideration. In another case, some applications have requested excessive permissions [78] for malware detection. However, because the mobile device's computational resources, processing capability, and memory storage [83] are limited, they have not utilized it. Furthermore, a model learned from attack data collected from one platform cannot be directly applied to analyze attacks targeting other platforms [87]. It's also impossible to create a linguistic model that describes the decision [111]. Moreover, when a researcher considers the kind and mode of new malware, the embedding space may have an unknown distribution. An ensemble method would improve the malware detection algorithm's robustness [118].

VI. FUTURE DIRECTIONS

This section responds to RQ6 by suggesting future research areas based on machine learning, which researchers and developers can use to reduce malware threats in cybersecurity. The following are potential solutions for each research gap as a future direction.

A. OBFUSCATION TECHNIQUE

According to [51], a dynamic framework for predicting future malware behavior and testing it with several benchmark datasets is required.

B. MALWARE FEATURES

In empirical experiments, a combination of state-of-the-art lexical and statistical [54] techniques could aid in determining the efficiency of malware features. It is also recommended that more malware families are studied [59] or that different learning approaches be used for family identification, such as deep learning techniques. In addition, use an ensemble model [62] to classify malware using various malware features, including system calls, API calls, and opcode sequences [66]. Those features can strengthen the feature space. Furthermore, to detect unknown malware [36], more records with unknown features [71] are needed to feed on the detection model. Thus, more datasets are also required [88]. An extensive set of features for visualizing the performance on a broad spectrum can be obtained.

Meanwhile, because the current method [89] only uses a few features, more histogram-related features can be added to improve accuracy. Malware detection and analysis can also be sped up by incorporating various technologies [108]. The malware feature extraction techniques [119] also require more research, and a hybrid malware feature extraction strategy that includes both static and dynamic analysis might be developed.

C. CLASSIFICATION METHOD

One of the ways to solve classification method issues is to extend the work by studying other classifiers' performance [75], [79] and considering their hyper-parameters for efficient and high-accuracy classification. The efficiency-accuracy trade-off must be thoroughly examined [84]. On the other hand, an experiment with additional statistical scoring techniques 96 needs to be conducted in malware classification. Also, the researcher can fine-tune the machine learning classification parameters [93] and add more APK samples, hopefully increasing the accuracy. Extending classification using other techniques, such as Deep Learning [97], can also help in solving this issue.

Referring to [102], other query strategies can be applied to see if they can perform better in terms of cost-effectiveness and accuracy. It is also suggested to combine the classifier with dynamic analysis [106]. The proposed classifier should then be tested and evaluated on various platforms, including desktop computers. Researchers can fine-tune the machine learning classification parameters [93] and add more APK samples, hopefully increasing accuracy. Extending classification using other techniques, such as Deep Learning [97], can also help in solving this issue.

D. DATASET

It is essential to test the robustness of the clustering mechanisms against adversarial samples [56] and their usability in the cloud environment by evaluating the model performance on a larger dataset. Since most of the issues related to the dataset are about the insufficient dataset, a more extensive dataset [57] is needed during empirical experiments for more accurate training of the model and to expand the known malware [33]. Furthermore, evaluation using a more extensive range of malware is essential [96] so that the results can be more representative. This may include executable and script files, images, PDF files, ransomware, etc. [112], to assess the number of unknown samples that one would expect to see in realistic environments and better datasets and reevaluate thereon. On the other hand, the use of deep learning classifiers and different feature selection approaches on the dataset might be examined further [63].

E. ANALYSIS TYPE

Most of the solutions combine static and dynamic analysis [58] to improve efficiency further. It is also in [68], to adjust the fitness scheme to evaluate the method's performance more reasonably. According to [77], the author compares dynamic analysis to an analogous static approach. The other option [80] is to use blockchain to construct a deep neural network framework for malware detection that combines static and dynamic analysis. Thus, a more robust way to resolve cycles over time [87] can be conducted. Meanwhile, based on [104], expanding the methodology by considering two categories of dynamic and hybrid malware analysis and comparing the results is another solution that can be considered further.

F. FALSE POSITIVE RATE

To lower the false-positive rate, more experiments [92] are needed. It can boost the ability to detect unknown viruses while also guaranteeing that detection is accurate and precise.

VII. EMPIRICAL ANALYSIS

This section examines three different types of machine learning algorithms to demonstrate some of the current concerns raised in the previous section. The chosen dataset and the experiment's selected attributes are described first. The experiment setup is then briefly described. Finally, the experiment findings show the present issues that current machine learning confronts in malware detection.

A. DATASET SELECTION

Elastic Malware Benchmark for Empowering Researchers (EMBER 2018) [120] was utilized for this experiment, which gathered features from 1 million PE files scanned in or before 2018 and divided them into eight groups of raw features that comprise both parsed and format-agnostic information. The five types of parsed features are general file, header, imported, exported, and section information. In contrast, format-agnostic features include byte histogram, byte-entropy histogram, and string information.

The EMBER repository makes it simple to train the benchmark models repeatably, expand the feature set supplied, and categorize additional PE files using the benchmark models. This repository makes it simple to produce raw features and/or vectorized features from any PE file. The Library to Instrument Executable Formats (LIEF) project [121] extracts features from PE files in the EMBER dataset. The raw features are converted to JSON and added to the publicly accessible dataset. From these raw features, vectorized features can be created and saved in binary format, which can then be translated to CSV, data frame, or any other format. The dataset was divided into two parts: 80% for training, which included 800K training samples (300K malicious, 300K benign, and 200K unlabeled), and 20% for testing, which included 200K test samples (100K malicious and 100K benign).

B. EXPERIMENT SETUP

The Python programming language, which comes with a sizeable standard library including valuable codes and functions, was used to create various machine learning models. As a development environment, JupyterLab is used where it serves Jupyter notebooks, code, and data accessible via the web browser, especially for machine learning workflows.

TABLE 10. Comparison of malware detection performance using a small and large dataset.

Type of ML Algorithm	Existing Research			Our Research		
	SVM [119]	DT [88]	N-gram [96]	SVM	DT	N-gram
Type of file	Win32-executable files	Internet files	Win32-executable files	Windows portable executable (PE) files		
Number of samples	1413	220	24	1 mil		
Dataset	CA Technologies VET Zoo & publicly available data sources	Clean & malware files are scrapped from the Internet	openmalware.org	EMBER 2018		
Classification Method	Behavior-based	Behavior-based	Signature-based	Signature-based		
Analysis Type	Dynamic	Hybrid	Static	Static		
TPR (%)	1	1	1	0.94	0.86	0.9
FPR (%)	0	0	0	1	1	1
FNR (%)	0	0	0	2.4	3.5	3.2
ROC (%)	100	100	100	99.93	99.64	99.81
F1-Score (%)	100	100	100	98.24	97.45	97.68
Precision (%)	100	100	100	98.94	98.85	98.91
Recall (%)	100	100	100	98.46	96.08	97.04
Accuracy Rate (%)	100	100	100	**98.62**	**96.49**	**97.43**

Besides, mamba and conda were used to run the Jupyter notebooks. Three types of ML algorithms were selected for this experiment, including SVM, DT, and N-gram, based on the current performance results of malware detection with a 100% accuracy rate.

Meanwhile, the dataset has eight raw characteristics in the general file information, header information, imported functions, exported functions, and section information, as well as format-agnostic histograms such as byte histogram, byte-entropy histogram, and string information.

C. RESULT AND DISCUSSION

TABLE 10 summarizes the outcomes of the experiments. TPR, FPR, FNR, ROC, Precision, Recall, F1-Score, and Accuracy are measures used to evaluate the effectiveness of three different types of malware detection algorithms. Furthermore, TPR, Precision, Recall, F1-score, ROC, and Accuracy are all supposed to be high in an effective malware detection algorithm, whereas FPR and FNR are expected to be low. TABLE 10 shows the performance data of three different malware detection methods. From this table, the accuracy rates of all machine learning algorithms to detect the malware can't obtain a 100% accuracy rate as obtained by the previous researchers. The size of the dataset jeopardizes the accuracy rate. Thus, an insufficient or small dataset might produce an inaccurate accuracy rate for malware detection. Based on the accuracy rates obtained in the SVM, DT, and N-gram experiment, each obtained 98.62%, 96.49%, and 97.43%, respectively, which is not consistent with what has been acquired by previous researchers. However, SVM has achieved the highest performance among the three algorithms, which is relevant to be applied in a more extensive experiment in the future.

VIII. CONCLUSION

Due to the introduction of new technologies and the enormous expansion of data in the big data era, machine learning has emerged as one of the most exciting approaches in cybersecurity, particularly in the detection of malware. To summarize, machine learning has sparked the interest of researchers working in a variety of application domains. Therefore, this study provided a comprehensive assessment of machine learning for malware detection employing in-depth SLR techniques. By evaluating the trends and patterns of 77 selected research from diverse sources, this study provided considerable insight into the present concerns and obstacles that machine learning faces in identifying malware attacks. This study developed a taxonomy for malware identification that categorizes them into numerous subcategories based on a thorough investigation of relevant papers. Malware detection was classified according to classification techniques, analysis types, datasets, challenges, and related issues faced in malware detection and future directions.

Finally, an empirical analysis was carried out to compare the existing performance results produced using VM, DT, and N-grams, which use small datasets with new accuracy rate results using a large dataset. The result shows that if the algorithm is trained using a larger dataset, the accuracy rate is significantly reduced from 100% for SVM, DT and N-grams to 98.62%, 96.49% and 97.43%, respectively. We can say that an insufficient dataset might influence malware detection accuracy. Furthermore, the classification method and analysis type selected for the experiment also contributed to the accuracy rate. The behavior-based classification method and dynamic or hybrid analysis type have a better contribution to detecting the malware than the signature-based and static analysis methods.

In terms of future work, we plan to run experiments on the chosen machine learning algorithms, focusing on feature extraction, classification method, and analysis type.

APPENDIX A

See Tables 11–14.

TABLE 11. The description of malwares with their threat strategies.

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
1	The Brain Virus	1986	Virus	The first computer virus was designed for the IBM Personal Computer and its compatibles. The bad sectors on infected drives are typically five kilobytes in size, while the infected boot sectors frequently have the disc label altered to ©Brain.	A copy of the virus is added to a floppy disk's boot sector. The actual boot sector is relocated and tagged as bad.
2	VirDem		Virus	There are 14 versions of the parasitic virus on DOS platform. It was the first virus to infect a file on the system, and it arrived about a year after the introduction of the Brain boot sector virus.	It infects the first clean DOS executable file by injecting its code at the beginning and moving the original code to the end when the virus is run.
3	Stoned	1987	Virus	A virus on the DOS platform	The Stoned becomes a resident in the RAM as soon as the machine starts up from an infected disc. It examines the Master Boot Record to see if it is clean before infecting it when starting from a disc other than the hard drive.
4	Yale		Virus	Another name for this virus is Alameda. It is said to be poorly programmed, and some virus specialists have dubbed its programmer "lousy."	When an uninfected disc is introduced, it infects it. The virus enters memory and takes up a kilobyte of space. The virus relocates the original boot sector to track 39, head 0, and sector 8 of the file system.
5	Vienne		Virus	A virus on the DOS platform is a straightforward virus that served as a model for more complicated and inventive viruses such as Zerobug, Chameleon, and Ghostballs. Its source code has been widely distributed, resulting in hundreds of versions. Its source code has been extensively circulated, resulting in hundreds of different variations.	It scans the system for .com files and infects one of them. The timestamp of the infected file will read "62," an impossible figure, making them simple to find. When Vienna tries to infect them by overwriting the first five bytes with the hex character string "EAF0FF00F0" instructions that trigger a warm reboot when the program is started, one of six to eight files will be deleted. These files do not contain the Vienna virus; they have just been damaged by it.
6	Morris Worm	1988	Worm	The first internet worm was found to garner widespread media attention and emphasize the need for improved network security. The worm was created at Cornell but was distributed at MIT to conceal its origin.	The worm infects a system by using vulnerabilities in rsh, fingerd, and sendmail on Solaris and BSD systems. If the worm discovers that the new system is susceptible, it sends data that allows the main worm to be downloaded.
7	Jerusalem		Virus	The virus was created for DOS file infector which was found in Israel with evidence from 1991. However, some records indicate that it may be from Italy.	This virus infects any DOS executables. The virus enters the memory after being executed and remains there long after the host software has been stopped. After the first infected file is opened, the virus spreads to any applications that open it, but it stays away from command.com.
8	HI.COM		Worm	A computer worm attacked VAX/VMS computers over the DECnet. The purpose of this worm was to use the compromised system to transmit a Christmas message from "Father Christmas". The other name for this worm is Father Christmas.	The worm was designed to produce a file called "Hi.com." All users on the local access database for every network received a greeting from Santa Claus. It was only directed at VAX/VMS platforms.
9	Yankee	1989	Virus	The other name for this virus is Yankeedoodle which targeted DOS platform. It resembles the Vacsina virus a lot. If it is in memory, the virus will play the song "Yankee Doodle" every day at 17:00.	The virus enters the memory when Yankeedoodle is run. Every .com and.exe file that is launched contracts the virus, which appends itself to the end of the file.
10	FuManchu		Virus	Any executable application, including .BIN files, .SYS, and overlay will become infected by this virus. It resembles a modified form of the Jerusalem virus.	The Fu Manchu virus embeds itself at the start or the end of .com or .exe files. The virus is loaded into memory by running an infected software, which influences runtime

TABLE 11. *(Continued.)* **The description of malwares with their threat strategies.**

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
					performance and corrupts program or overlay files.
11	Datacrime		Virus	When an infected file is sent to and executed by a computer, data crime occurs, which causes very little damage.	It replaces the original portions of the hard disc and shows an error message. When an infected file is run, it scans accessible drives for.com files in the following order: C:, D:, A:, B:. It avoids any file with the letter "D" as the seventh letter, most likely to prevent infecting command.com. Every time the virus runs, it infects one.com file in the working directory.
12	Whale		Virus	A virus on the DOS platform which infects .EXE and .COM files. The whale was the most significant DOS virus ever identified at the time, weighing in at 9,216 bytes.	The virus is placed in memory when an infected file is run. The virus occupies 9,216 bytes in a file but 9,984 bytes in memory. At different periods, the virus operates differently, sometimes just infecting the uninfected. When a .com or .exe file is run, it can infect the system or be read. It appends 9,216 bytes to the file's conclusion. In some instances, the virus may erase a file when it is copied.
13	Frodo	1990	Virus	The first DOS malware with perfect stealth. Frodo, taken from The Lord of the Rings character, is one of the most straightforward viruses to eradicate and takes its name from the text it displays.	The virus enters memory when an infected file is run. It infects every file with a .com or .exe extension that the user accesses, attaching itself to the end of those files. It will add 100 years to the file's timestamp. Some data files may also be corrupted.
14	DiskKiller		Virus	The first boot sector virus on DOS that can handle sectors more significant than 512 bytes and a memory-resident boot sector virus.	Disk Killer conceals itself in sectors it flags as "bad" in the FAT, like several other boot sector viruses. The data on the disc is effectively destroyed because of the virus's encryption of it by XORing sectors with 0AAAAh and 05555h alternatively. Many boot sector viruses have a very similar infection and reproduction method.
15	Amoeba		Virus	The virus targeted DOS and is considered a memory-resident parasitic polymorphic encrypted virus. The 1392 varieties are bug-infested viruses that inflict minor damage and fail to function correctly.	The infectious sequence is complicated to understand. Any executable and OVL files that are accessed become infected by the virus, which hooks INTs 10h, 1Ch, and 21h. While awaiting a chance to infect, the virus becomes a permanent resident of the higher memory.
16	Tequila	1991	Virus	A sophisticated multipartite virus. It is well-known for its armoring tactics, notably in decryption.	The boot sector record becomes infected by the virus when an infected file is run. The virus shrinks the partition of the disc by six sectors while inserting its code in the sectors outside of the partition. The infection settles into memory when the disc boots. The virus adds 2,468 bytes to .exe files when they are run.
17	Oligomorphic		Virus	This virus's decryptors vary with each generation.	Using a series of decryptors rather than just one makes changing the decryptors the most straightforward process possible.
18	Michelangelo	1992	Virus	A virus that infects the DOS boot section. This virus is a Stoned version. Michelangelo gets its name from its activation date, the birthdate of Renaissance artist Michelangelo.	It doesn't interact with the operating system and only acts at the BIOS level. The virus overwrites the first 100 complex drive sectors, including the master boot record and the file allocation table, whether the computer is an AT or a PS/2, and transfers the original relevant boot sector somewhere on the disc.

TABLE 11. *(Continued.)* **The description of malwares with their threat strategies.**

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
19	Leandro		Virus	The virus was set up as a "time bomb". It was set to go off on a specific day and commonly distributed via shareware on floppies, but as Internet usage grew fast.	The virus is transmitted by affixing its code to add files on your computer or network, resulting in some programs stopping functioning properly.
20	Strange		Virus	This virus stops people from viewing pirated content online in an unusual vigilante operation.	It prevents infected users from visiting various sites focused on software piracy.
21	PMBS	1993	Virus	It's a nasty memory resident boot virus that uses the internal string "PMBSVIRS" which examines the ports' input/output and corrects the data intended for export after reading an infected MBR.	It runs a virtual V86 machine for applications and DOS execution, replicates itself into extended memory, and puts the computer into protect mode. It infects and hooks all interrupts, checking critical situations when reading the diskette. Another urgent circumstance causes the computer to hang up after displaying one of the alerts.
22	Onehalf		Virus	Slovak Bomber, Freelove, and Explosion-II are all other names for Onehalf, considered polymorphic boot viruses detected on DOS.	The virus compromises executable files and the hard drive's master boot record. Files with filenames containing any of the following strings are ignored: FINVIRU GUARD NOD SCAN CLEAN MSAV VSAFE CHKDSK
23	Phantom1	1994	Virus	The virus is a parasitic polymorphic virus on DOS that lives in memory. To infect any executable that is launched or opened after the virus has been loaded into memory, it hooks INT 1Ch, 21h, and writes itself to the end of the file.	An image of the Grim Reaper's head with flashing eyes shows on the screen after around 20 minutes of inactivity at the keyboard. After a looping background animation plays, the message is then displayed, and the text "PHANTOM 1" fades in. The keyboard is then turned off.
24	Shifter		Virus	A virus that utilizes DOS	By inserting its code into built .OBJ files, the virus spreads by ensuring it is present in every legitimate application created from the.OBJ file.
25	WM/Concept	1995	Virus	The first uncontrolled macro virus for Word products. It wasn't the first Word macro that distinction belongs to DMV, but it was the first wacky one. It was discovered that several CDs distributed by some significant businesses had it preloaded.	The virus examines the document template normal.dot to find macros named FileSaveAs and PayLoad when opening an infected file. If it finds them, normal.dot is considered infected and stops the operation. Otherwise, it adds the macros to the template.
26	Laroux	1996	Virus	A group of macro viruses that propagate using spreadsheets made with Microsoft Excel.	Once this virus has invaded the Excel environment, it remains active whenever Excel is launched and infects both newly produced Excel workbooks and older workbooks when they are accessed. The two macros that makeup Laroux are checking files and auto open. The check files macro defines the Excel starting path after the auto-open macro, which runs anytime a corrupted Spreadsheet is opened. The virus produces a file called PERSONAL.XLS if it doesn't already exist in the starting path. Laroux is a module found in this file.
27	OS2 AEP		Virus	The first virus to infect OS/2 executable files.	The viruses either deleted the file, wrote themselves to the file's location, or used the companion virus approach.
28	Win95.Boza		Virus	Bizatch is another name for this virus. It was the first virus for Windows 95, infecting files in the current directory. When it is run, it infects up to three files with around 3,192 bytes of code attached to them.	This virus spreads by adding its code to other files on the device or network. Some of the programs may halt working correctly.

TABLE 11. *(Continued.)* **The description of malwares with their threat strategies.**

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
29	AOL Trojan	1997	Trojan	Messages attempting to distribute the Trojan target AOL Instant Messenger users.	The user's account has been suspended and will be terminated in 72 hours, according to the notification. The user must download a crucial AIM update to fix this issue. The "update" is a dangerous Trojan.
30	mIRC Worm		Worm	A worm spreads over Internet Relay Chat (IRC) networks, message boards, or chat rooms by distributing infected files or webpages through IRC channels; since the IRC network is linked to hundreds of channels, it is vulnerable to worm attacks.	This worm can propagate by connecting to the IRC network or by dropping detailed scripts into the IRC user directory. The installed script will force the user to transmit a copy of the infected file to other users in the same channel whenever the infected user signs into the IRC server and connects to any channel.
31	Esperanto Virus		Virus	The first virus runs on several processors. The virus may infect files on computers running on Mac, Microsoft Windows, DOS, and x86 processors.	If there isn't a running copy of Esperanto in memory when Esperanto is executed on a DOS or Windows platform, it enters memory-resident mode. It infects .com, .exe, NewEXE, and Portable EXE files as they are run.
32	Autostart	1998	Worm	A Macintosh worm that only existed once. It originated in China and made it into several CDs from software providers preinstalled.	A file called "DB" is started by the worm when a disc infected with this worm is mounted on a power Mac computer. It is running on QuickTime 2.0 or later. It is a concealed program file, and "????" indicates the creator. Within the Extensions folder, it duplicates itself. It alters the file's name to "Desktop Printer Spooler," a concealed file. The computer is then restarted via Autostart.
33	Cross		Virus	A virus also known as Hopper can contaminate Word documents with .vbs and .html files.	The virus examines to see if the file is already infected before running. It adds its code to the .vbs and .html files. It instantly compromises the normal.doc template if it is opened from a .doc file. When a .doc file is closed, an infection takes hold. It just comments out the portions of itself that are particular to HTML when infecting .vbs and .doc files. It disables MS Office's VirusProtection feature.
34	CIH		Virus	An extremely deadly virus for Microsoft Windows, sometimes known as Chernobyl or Spacefiller, exclusively affects Windows 95, 98, and ME. A code remark inspired the name.	The virus becomes resident on a system when a CIH-infected file is run because it infects all executable files that are accessible. Due to the way CIH infects files, the infected files are frequently the same size as the uninfected ones. The virus begins by looking for long stretches of vacant or unused space in any big file to accommodate its code. If the available space is insufficient, CIH will attempt again and hunt for a location with the adequate overall capacity to accommodate its code in specific-sized chunks. It will act in a typical way of an infection if this check fails. It will sign up as a driver to elude simple cleaning procedures.
35	ExploreZip	1999	Worm	A worm for bulk emails and the first worm to be packed using a program like UPX. The worm's body changes with each subsequent replication, yet it remains visible. It still weighs about 210,432 bytes, making it a relatively huge worm even after compression. Following that, much smaller worms like Navidad were produced.	Zipped_files.exe is the name of the attachment. When ExploreZip is run, a notice states that the zip archive is invalid. Although the OK button is always in the language that the infected machine is configured to, the message is always in English. The worm duplicates itself as Explore.exe or _setup.exe in the System folder.

TABLE 11. *(Continued.)* **The description of malwares with their threat strategies.**

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
36	Happy99 Worm		Worm	A Microsoft Windows email/newsgroup worm that exhibits some characteristics of a virus and trojan. It was authored by Spanska and published in the 29A virus magazine's fourth issue. The author of Happy99 calls it a "sympathetic hitchhiker who utilizes your internet connection to travel and thanks you for the trip with a tiny animation," even though Happy99 is wild and without a destructive payload.	This worm is a specific type of virus that replicates its copies. It sends itself through the Internet as an attachment in e-mail messages rather than infecting disc files as its primary objective.
37	Melissa		Virus	One month before CIH's payload was published, a highly hazardous macro virus swept East Asia and caused hundreds of millions of dollars in damage. It was one of the earliest viruses to become well-known.	The email containing the virus has the subject "Important Message From [email address of the account from which the malware was delivered]." The real email address that sent it will be shown as the "sender." "Here is the paper you requested... don't show anybody else ;-)," reads the message's body. The list.doc attachment has 80 usernames and passwords for pornographic websites. All Word documents, by default, utilize the Normal.dot template, which is infected.
38	ILOVEYOU		Worm	The worm is referred to as LoveLetter or LoveBug on occasion. The worm spreads in text source form and is a text script software. Because hackers may alter the worm's programming, there are several iterations of the original worm.	The email contains an attachment that, when viewed, causes the message to be sent to everyone in the recipient's Microsoft Outlook contact book. The email has the subject line "ILOVEYOU" and contains the worm.
39	Pikachu	2000	Worm	A Microsoft Windows worm that spread over email and was thought to be the first worm targeted towards users in their teens and younger years since it was named after the Pokémon character. Visual Basic 6.0 was used to write it.	This worm spreads via Microsoft Outlook email by attaching the file PikachuPokemon.exe to email correspondence. The worm is an executable Win32 PE file roughly 32 KB in size. A poorly sketched Pikachu appears as the icon. Before deleting any files in the Windows and Windows system folders, the worm overwrites the original C: AUTOEXEC.BAT file with its deletion instructions.
40	Nimda Worm		Worm	One of the first Windows worms that could operate automatically without the user even reading the infected email. It is also the first to alter websites so users can download copies of themselves. Additionally, it contains a virus that infects executable files.	The Nimda worm may spread to other computers and networks through five distinct routes, including through an email, infected website, local network, server, and file infection.
41	Sadmind Worm	2001	Worm	A web worm that may alter web pages on Microsoft IIS servers running Windows NT 4.0, 2000, XP, and Solaris systems. It first surfaced just before the CodeRed worm, and because both of them were from China, they could be connected. It exploited security holes that had been fixed by Sun Microsystems and Microsoft for more than a year, underscoring the significance of constantly installing system updates as soon as they become available.	Sadmind generates IP addresses to locate new computers to infect. Each address is checked to verify if a portmap service is available and listening on port 111. It searches for systems that fit these criteria and determines whether they are running the sadmind remote administration service.
42	Sircam Worm		Worm	A Microsoft Windows mass-mailer worm said to have originated in Mexico is well-known for its capacity to attach unknown documents to its emails and transmit them along with the worm, possibly disclosing sensitive, private, and even humiliating information. It mostly spreads by email but is also network-aware.	Depending on the language used by the sender, Sircam appears in an email that might be either in English or Spanish. It includes an attachment with two file extensions: one for a document file of some description and the other for an executable. The attachment may be longer than the worm itself since it contains a real Word, Excel, or Zip file previously on the system the worm was on. It will accept a file it added to itself on the prior machine, along with the name and the initial extension. Three options are available for the initial extension:.doc, .xls, and .zip. There are four options for the final extension: .bat, .com, .lnk, and .pif.

TABLE 11. *(Continued.)* **The description of malwares with their threat strategies.**

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
43	KLEZ	2002	Worm	The worms caused the most significant damage in history, costing around $19.8 billion. It is particularly renowned for its capacity to forge sender line email addresses and for infecting the machine of the recipient even when they only preview or read the message without downloading or running the attachment.	Klez may infiltrate a system via email or network sharing. The worm employs forged email addresses that focus on the "From" section.
44	Mylife Worm		Worm	A potentially harmful email worm for Windows	Three variations of the ransomware exist a girl holding a flower, a parody of Bill Clinton, and an ox with a sinister message. When a user clicks an attachment, all emails and, within 45 minutes, all system files on the user's computer are deleted.
45	Beast Trojan		Trojan	The malware is called a Remote Administration Tool, or "RAT," which is used most frequently in the hacker community and refers to a backdoor trojan horse that runs on Microsoft Windows. It was created in Delphi and quickly gained popularity for its distinctive characteristics. From 95 through 10 versions of Windows are susceptible to infection.	One of the earliest trojans to offer a reverse connection to its victims, Beast grants the total attacker control of the infected machine once it is up and running. The traditional client-server architecture is used, in which the server infects the victim while the attacker operates the client. The trojan is safe until it is opened; after then, it injects its code into other apps. The three files must be deleted in safe mode with System Restore disabled on a Windows XP computer to clean the system.
46	SQL Slammer	2003	Worm	A worm that cost around $1 billion to repair. The Microsoft Windows operating system is impacted. The root of the issue was the buffer overflow flaw in Microsoft's SQL Server and Desktop Engine database systems.	David Litchfield, who had first identified the buffer overflow vulnerability that the worm exploited, built the proof-of-concept code that was used to present the worm at the Black Hat Briefings. A little code primarily generates random IP addresses and sends itself to those addresses.
47	Sobig Worm		Worm	The worm first surfaced a little more than two weeks before Slammer. With a reported total cost of $37.1 billion in damage, it was one of the most destructive worms of its time. Additionally, it has broken records for its capacity for spreading, notably the volume of emails received with it attached.	The email that contains Sobig has the sender address "big@boss.com." There are four potential topics, including Re: Films, Regarding Sample, Document, and Sample: This is that Sample
48	Blaster Worm		Worm	An online worm also goes by the name Lovesan. With extensive Distributed Denial of Service (DDoS) assaults that caused hundreds of millions of dollars' worth of damage in the late summer of 2003, it wreaked havoc. Blaster's most popular moniker comes from the msbast.exe program it deposits in the Windows System folder. Its second moniker, Lov(e)san, is derived from the worm's "I LOVE YOU SAN" phrase. Several publications also refer to this worm as Poza.	The system will receive code that exploits a DCOM RPC vulnerability (described in Microsoft Security Bulletin MS03-026) from the Blaster worm on an already infected computer coming through TCP port 135. There is an 80% chance that the worm will send exploit code specific to Windows XP and 20% that it will be specific to Windows 2000. The RPC subsystem will fail if the exploit code does not match the system. On Windows XP and Server 2003, this causes a system reboot. In Windows 2000 and NT 4.0, this causes the system to be unresponsive.
49	Mydoom Worm	2004	Worm	A Microsoft Windows worm that reportedly caused $3k more damage than Sobig. Thus, making it the most destructive worm ever unleashed. It also broke records for dispersal power.	Mydoom may be distributed by email or Kazaa file sharing. The worm must be downloaded from an infected computer on the Kazaa network to be distributed via Kazaa. Mydoom may also appear in an email address with a fake sender address and various subject lines.
50	Santy Worm		Worm	A Microsoft Windows worm	It spreads the worm by exploiting a flaw in the famous phpBB discussion forum software. It also used Google's search engine to locate susceptible servers. It does not infiltrate computers used by end users.

TABLE 11. *(Continued.)* **The description of malwares with their threat strategies.**

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
51	Commwarrior		Worm	A Bluetooth worm that attacks the SymbOS operating system. When it first appeared, this virus spread quickly and was frequently utilized on other SymbOS viruses and derivatives, such as Doomboot. It was also one of the first mobile phone viruses to propagate over Bluetooth and MMS (Multimedia Messaging Service).	When Bluetooth sends data to a device, it asks the user to install it. If the user installs the worm, Bluetooth and MMS will be affected.
52	Zotob Worm	2005	Worm	A computer worm that takes use of security flaws in Windows 2000 and other Microsoft operating systems, such as the plug-and-play flaw MS05-039. It has been reported that this worm may propagate over TCP port 445 or Microsoft-ds.	The Rbot worm served as the ancestor of zotob. An infected machine may be made to restart itself repeatedly via Rbot. Every time a computer restarted, Zotob would reproduce itself, multiplying until each device had several copies of the file by the time it was deleted. The Blaster worm and this are comparable.
53	Zlob Trojan		Trojan	A trojan can infect computer users by posing as an Active X-based fake video codec, but it can also infect the host machine via malicious software.	After being installed, it shows popup adverts that seem like genuine Microsoft Windows warning popups, alerting the user that malware has been installed on their machine. When these pop-ups are clicked, a bogus anti-spyware application (like Virus Heat and MS Antivirus) that contains the trojan is downloaded. Reference name="tm"/
54	Brontok Virus	2006	Virus	An email worm and virus that targets the Microsoft Windows operating system. When specific words are found in a window's title bar (such as "Registry"), one typical indication is an automated reboot of the machine.	Brontok transfers itself to the user's application data directory when it is launched. After that, it configures itself to run with Windows by adding a registry entry to the HKLMSoftwareMicrosoftWindowsCurrentVersion registry key. Open the registry key. It alters Windows Explorer settings and disables the Windows Registry Editor (regedit.exe). The "Folder Options" option from the Tools menu is removed, making it harder for users to retrieve hidden files where it is hidden. Additionally, the Windows firewall is disabled. In certain variations, the computer restarts when a window is discovered that has specific strings (such as "application data") in the window title. When an address entered into Windows Explorer is partially blanked out, it might be frustrating for the user. It sends itself to email using its own mailing engine.
55	Stration Worm		Worm	An e-mail worm from the computer worm family, also known as Stratio and Warezov. It can infect Microsoft Windows systems, disabling security features and spreading to other machines via e-mail attachments.	The Stration worms use social engineering to infect the target computer by arriving in an email that appears to be a report from a mail server informing the recipient that their computer is infected due to an unpatched security flaw in Windows and offering as an attachment a purported fix, which is the worm program itself, in somewhat broken English. Some subsequent iterations of the worm disseminated using Skype and instant messenger conversation notifications that contained a URL pointing to the worm.
56	Storm Worm		Worm	A backdoor phishing Trojan horse that attacks machines running on Microsoft operating systems.	The worm sends an email with the subject line, "230 dead as storm slams Europe," to discuss a recent weather tragedy. The virus installs the wincom32 service when an attachment is opened, injects a payload, and transmits packets to locations encoded inside the malware itself.
57	ZeuS Trojan	2007	Trojan	Malware that is installed on Microsoft Windows. It was first used to steal data from the US Department of Transportation but didn't catch on until 2009. It has also installed Cryptolocker on occasion. It was regarded as one of the most successful, infecting millions of machines globally. It has appeared in several frauds.	It is spread by phishing and drive-by downloads.

TABLE 11. *(Continued.)* **The description of malwares with their threat strategies.**

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
58	Conficker	2008	Worm	The other names for this worm are Downadup, CONFLICKER, or Kido, and a Microsoft Windows worm that has received much media attention and might have originated in either Ukraine or China. The media overhyped it, claiming it would carry a hugely destructive payload. While this never occurred, it is impressive, given the number of PCs allegedly affected.	Conficker infects a new system by receiving malware that takes advantage of the MS08-067 flaw. An RPC request with exploit code that uses a buffer overflow vulnerability to download and run the worm will be sent to the target machine. It will be downloaded as a .jpg file from an HTTP server with the worm installed on the infected computer.
59	Koobface Worm		Worm	A malware that targeted the Windows operating system and reported to target Facebook to propagate through infected wall postings.	Koobface hides via the VBInject trojan. If a download is permitted, Koobface will operate local web and IRC servers, allowing it to join a botnet, modify the DNS, and do various other tasks. These additional features could be installed from the first download or from different files that might be installed later.
60	TDL3	2009	Rootkit	An infection that is incredibly clever and has infected millions of computers worldwide. The TDL3 Rootkit, similar to the original TDSS Rootkit, may tamper with Internet surfing and search results, trigger sporadic crashes and "blue screens of death," and render a computer system unresponsive and unstable.	The TDL3 Rootkit gives hackers access to your computer so they may use it as a botnet node or launch malware attacks against it.
61	Kenzero		Trojan	A malware that spreads over peer-to-peer networks and is designed to track its victims' online activity.	Kenzero targets victim machines that download files from peer-to-peer networks (P2P). The malware locates the victim's surfing history after the file is opened and posts it online. Users can then see the files.
62	W32. Dozer		Worm/ Trojan	It was designed to delete data from infected machines and stop them from rebooting.	Worm releases Trojan. Dozer causes a distributed denial of service (DDoS), and W32. Mydoom. A@mm, the W32.Dozer component is responsible for email transmission. These parts work together to carry out DDoS assaults and distribute via email.
63	Stuxnet	2010	Worm	A computer worm that aimed at the Iranian nuclear plant to harm that nation's uranium enrichment program and stop President Mahmoud Ahmadinejad from developing a nuclear weapon.	Stuxnet is deployed against industrial systems rather than trying to steal sensitive information such as credit card information or passwords. The centrifuges self-destruct, as a result, causing significant damage.
64	Waledac Botnet		Botnet	A botnet that was primarily used as email spam. It is also known as Waled and Waledpak. Microsoft eliminated the botnet in March 2010.	Accepting directives from a remote server is conceivable. Commands also provide instructions on what to do.
65	Kelihos	2011	Botnet	A botnet that goes by the name Hlux is mainly used for spamming and bitcoin theft.	The Kelihos botnet is a peer-to-peer botnet, meaning each botnet node could function as the network's central command and control server. Traditional non-peer-to-peer botnets rely on a few servers for all the nodes' guidance and "work"; if these servers are taken down or deleted, the botnet will no longer get advice and will subsequently shut down. Peer-to-peer botnets aim to reduce that danger by enabling each peer to submit commands to the botnet, making it more challenging to take it down.
66	Duqu		Worm	A worm that operates on Windows. It resembles the Stuxnet and searches for data that might be utilized in an assault on industrial control systems. The known components are attempting to acquire information, not to do harm.	Duqu requires a thorough and time-consuming installation procedure. Duqu uses a specifically created Microsoft Word document to arrive. The Word document includes a zero-day kernel vulnerability, allowing attackers to covertly install Duqu on the machine without the user's knowledge. The installer and the exploit shellcode may be separated into two pieces to demonstrate the installation procedure as simple as possible.

TABLE 11. *(Continued.)* **The description of malwares with their threat strategies.**

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
67	Morto Worm		Worm	A worm that spreads over Remote Desktop Services on Windows computers by brute-forcing the server's login credentials.	The payload includes a worm propagation routine that uses accessible Remote Desktop Protocol (RDP) Services to infect other systems. This affects machines linked to the local subnet and internet-based RDP services available to the public. A hardcoded password list will be used to brute force the administrator login, and following a successful login, infection will start.
68	Medre		Worm	A worm that takes operational data. The worm gathers AutoCAD files, including drawings. ACAD/Medre.A is capable of being utilized for industrial espionage.	ACAD/Medre.A gathers data pertaining to the AutoCAD program. The worm gathers drawings from AutoCAD (*.dwg) files stored in information repositories.
69	Flame	2012	Virus	A virus that targets Windows-based PCs.	Flame has a "Kill" command that removes all virus traces from the computer. Through LAN or USB, it can spread to other computers. Keyboard input, screenshots, network activities, and Skype chats may all be recorded.
70	Shamoon		Virus	A computer virus that affected current 32-bit NT kernel versions of Microsoft Windows was found by Seculert. The malware appears to have been designed for cyber warfare and became known years later as the "largest hack in history." due to the virus's destructiveness and the expense of both the assault and recovery, it has been noticed that its behavior differs from that of other malware attacks. An infected machine can transmit Shamoon to other devices connected to the network.	Once a machine has been infected, the virus keeps a list of the files from places on the system, uploads them to the attacker, and then deletes them. Finally, the virus overwrites the computer's master boot record, rendering it useless.
71	Hesperbot		Trojan	The Trojan may connect to remote sites to deliver or receive orders from the attacker and log user keystrokes to steal sensitive information.	Despite being a new type of malware, ESET identified it as "Win32/Spy.Hesperbot" and gave it the name "Hesperbot." The security company calls it a "potent banking Trojan" because it could log keystrokes, take screenshots and videos, set up a remote proxy, and even set up a covert VNC server on the infected system.
72	CryptoLocker	2013	Ransomware/ Trojan	A well-known Microsoft Windows ransomware that spreads over email and is regarded as one of the original ransomwares. The executable file for CryptoLocker is included in a ZIP file that is attached to an email message. By utilizing Windows' built-in feature of masking file extensions from file names, this executable file has a disguised filename and icon that looks like a PDF.	The most common way this virus propagates is through emails sent to business email accounts that pose as customer support-related communications from FedEx, UPS, DHS, etc. The computer becomes infected when the zip attachment in these emails is opened.
73	Windigo	2014	Backdoor	A group of malicious programs designed to form a sophisticated network of botnets that can spread spam, reroute Web traffic, and infect users' machines with malware all while concealing the whereabouts of the hackers carrying out the assaults.	The main tools used by Windigo to steal login information, compromise web servers, and reroute traffic are Linux/Ebury and Linux/Cdorked backdoors. cPanel, a well-known web hosting control panel platform, and kernel.org have become notable victims of Windigo.
74	Bashlite	2015	Botnet	The malware infects Linux systems (DDoS) to launch distributed denial-of-service attacks. It was once known as Bashdoor; however, this name is now used to describe the malware's attack technique. It has been used to launch 400 gigabits per second assaults. Bashlite was created to quickly cross-compile to various computer architectures and is written in C.	For command and control, Bashlight employs a client-server architecture. The communication protocol is a lightweight variant of Internet Relay Chat (IRC). Most variations only have a single command and control IP address hardcoded, even though it allows numerous command and control servers. Using a built-in dictionary of popular usernames and passwords, it spreads through brute force. The virus establishes connections to random IP addresses and makes login attempts; it then reports any successful logins to the command-and-control server.
75	Linux Wifatch		Virus	A Linux virus that patched WiFi routers after infecting them and turning them into nematodes.	The virus will advise the user to upgrade firmware and change passwords in a message displayed after launching it. It refreshes definitions and functions similarly to an antivirus program. This uses its peer-to-peer network and removes any remaining virus remains.

TABLE 11. *(Continued.)* **The description of malwares with their threat strategies.**

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
76	Locky	2016	Ransomware	According to reports, a Windows macro trojan virus program and ransomware email worm have caused 4000 new infections each hour and around 100,000 per day, with most attacks occurring in Germany and the Netherlands.	Locky is spread by emails that look like bills or by using exploit kits on compromised websites. The Word document will download the Locky ransomware executable and start the encryption process when the user double-clicks it, enables macros, or runs the Javascript file.
77	Mirai		Botnet	Distributed denial of service (DDoS) assaults are the most destructive. Linux-based computer systems can be infected with this malware that converts them into "bots" that can be remotely controlled and used as part of a botnet in sophisticated network assaults. It primarily targets online consumer electronics, including wireless routers and remote webcams.	Internet-connected Mirai-infected machines continually search the internet for IoT devices' IP addresses. Private networks, addresses assigned to the US Postal Service, and the Department of Defense are among the IP Address ranges listed in a chart included with Mirai that it would not infect.
78	Wanna Cry		Ransomware	The original name is WanaCrypt, Wana Crypt0r, or Wana Decrypt0r, the famous ransomware worm on Microsoft Windows. It has caused chaos at airports, banks, universities, hospitals, and many more establishments and makes	Numerous attack vectors, such as worms and trojans, can cause infection. The program will extract an embedded file into the same folder as said executable when a computer contracts WannaCry. A password-protected zip folder containing several files utilized by WannaCry is the embedded resource.
79	Xafecopy Trojan	2017	Trojan	The targeted operating system is Android. It was said that the Trojan was included in several programs, most frequently in battery optimizers.	Xafecopy poses as a helpful app, frequently a battery optimizer. It works by clicking on online pages that use the WAP billing system, a type of mobile payment system that is paid straight to the mobile account. Based on the Ubsod family, the virus operates on Android devices that support WAP via a GPRS or 3G wireless connection.
80	Ransomware		Ransomware	Another name for this malware is encryptor virus, crypto trojan, lock virus, cryptovirus, or crypto worm, which renders personal data on a computer inaccessible in some way while requesting a ransom for its recovery, hence the term. Although the subject of crypto virology predates the word "ransomware," it is frequently used to characterize such harmful software.	Sending the recipient an email message with a specially constructed file or program attached allows for the execution of this form of extortion assault. The malware encrypts several files on the victim's PC if they open or execute the attachment. The victim is then presented with a ransom letter. The victim cannot open the encrypted data without the proper decryption key. The cracker may (or may not) transmit the decryption key, enabling decryption of the "kidnapped" data that are taken once the ransom required in the ransom letter has been paid.
81	Tanatos	2018	Worm	The other name for this worm is Bugbear which targeted the Microsoft Windows platform by releasing a backdoor/keylogger malware that may provide a hacker access to several components of the compromised machine. The preview pane of an unpatched system is where the worm may spread to a machine. It may also send emails stored on an infected system to a random email address. Additionally, it was prone to giving networked printers information that made them produce nonsense.	Tanatos can enter a system via network or email. When it comes to an email attachment, it employs a few sophisticated techniques to avoid being immediately recognized as a worm. The worm's network broadcast is considerably stranger than usual.
82	Titanium APT		Backdoor	A very advanced backdoor malware APT, developed by PLATINUM, a cybercrime collective. Due to the use of encryption and fileless technology, none of the files in the file system can be identified as malicious. The ability of software to imitate well-known programs is another trait that makes identification difficult.	A Trojan backdoor is then deployed at the end of a lengthy process that involves dropping, downloading, and installation phases in the Titanium APT. A significant portion of the sequence is cleverly concealed from detection, notably by stenographical concealing data in a PNG picture.
83	Swarm Virus	2019	Virus	An artificial swarm malware may exchange acquired information, accelerate the trial-and-error process, and use the specialization in swarm intelligence.	A virus that uses certain characteristics of swarm systems, or swarm algorithms, found in nature. An antimalware system can integrate the swarm behavior pattern.

TABLE 11. *(Continued.)* **The description of malwares with their threat strategies.**

No.	Name of the virus	Year	Type of malware	Description	Threat strategy
84	Shlayer		Trojan/ Adware	Ransomware infections can be used, along with other malware infestations and trojans that steal passwords. Various software cracking tools and the Adobe Flash Player installation is common guises used to conceal it.	Creates pop-up advertising that isn't truthful, free software installations (bundling), deceptive Flash Player installers, and torrent file downloads.
85	Ghost	2020	Ransomware	Since every file is encrypted, opening the file requires paying a ransom. A ransomware infection can be installed alongside other malware infestations and trojans that steal passwords.	The user cannot open files previously accessible by the user. These files now have a new extension, such as my.docx.locked. Your computer shows a message requesting a ransom. To release user files, cybercriminals demand a ransom payment, which is often made in bitcoins.
86	Snugy		Backdoor	This malware allows backdoor access to the hacked Exchange server while communicating with the actors via control (C2) channels and various commands. Snugy is a CASHY200 backdoor version.	To execute instructions on the compromised server, the Snugy backdoor leverages a DNS tunneling channel, which enables an actor to discover the system's hostname and execute commands.
87	Clop		Ransomware/ Trojan	The targeted platform for this ransomware is Microsoft Windows which MalwareHunterTeam discovered. It is also found in the CryptoMix family. The word clop in Russia means bug. The strategy of this ransomware is to attack large networks rather than single machines.	It will disable antivirus programs like Windows Defender and Malwarebytes. Then, it terminates several Windows services and processes. It closes all open files to prepare them for encryption. Windows Defender may be turned off by configuring different Registry entries, which turn off features including antispyware detections, real-time protection, behavior tracking, sample uploading to Microsoft, cloud detections, and Tamper Protection.
88	Gameover Zeus	2021	Botnet/ Trojan	A peer-to-peer botnet constructed from parts of the previous ZeuS virus. It is thought that the Cutwail botnet was used to propagate it.	Transactions can be completed without a centralized "Command and Control" server. Zeus Gameover can construct separate servers to deliver sensitive data without using centralized ones. It takes all of your money by gaining access to your private bank account information. In essence, it is impossible to find the stolen data.
89	Cryptojacking		Crypto	A cyber threat hides on a computer or mobile device and takes advantage of the device's resources to "mine" cryptocurrency forms of virtual money.	Utilize a person's computer resources to assist in "mining" cryptocurrencies like Bitcoin. Hackers are attempting to install cryptojacking software on computers and mobile devices to aid in the mining process and significantly slow down the user's device because mining demands a lot of computational power to produce new cryptocurrency.
90	Onyx	2022	Ransomware	The malware was written using the .net programing language, which has a method where Getprocess AP returns a list of processes that are currently operating on the host. When this malware is executed on the system, it first checks the process name and process ID to see if the malware instance is already operating. The new instance will not be executed if the malware instance is already operating.	After encrypting files, this ransomware changes their filenames by inserting the.ampkcz suffix. The "readme.txt" ransom note is dropped into each encrypted directory by this ransomware when it has finished encrypting the target device.
91	Raas		Ransomware	Ransomware as a Service (RaaS) is a business model in which ransomware operators pay affiliates to initiate ransomware attacks established by operators. They are widely available on the dark web, offered in the same manner as commodities on the regular web.	RaaS providers execute advertising campaigns and maintain websites that are identical to those of your own business. They tweet regularly and have videos and white papers.

TABLE 12. Quality assessment score of the selected studies.

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
P1	1	1	1	1	1	1	1	1	1	1	10
P2	0.5	1	1	0.5	0.5	1	1	1	1	0.5	8
P3	1	1	1	1	0.5	1	1	1	1	0.5	9
P4	1	1	1	0.5	1	1	1	1	1	1	9.5
P5	0.5	1	1	0.5	0.5	1	1	1	1	0	7.5
P6	1	1	1	1	1	1	1	1	1	0	9
P7	1	1	1	1	0.5	1	1	1	1	1	9.5
P8	1	1	1	1	0.5	1	1	1	1	1	9.5
P9	1	1	1	1	0.5	1	1	1	1	1	9.5
P10	1	1	1	1	0.5	1	1	1	1	1	9.5
P11	0.5	1	1	1	0.5	1	1	1	1	0	8
P12	1	1	1	1	1	1	1	1	1	0.5	9.5
P13	1	1	1	1	0.5	1	1	1	1	1	9.5
P14	1	1	1	1	1	1	1	1	1	0	9
P15	1	1	1	1	0.5	1	1	1	1	1	9.5
P16	1	0.5	1	0.5	1	1	1	1	1	1	9
P17	0.5	1	1	1	0.5	1	1	1	1	0	8
P18	0.5	1	1	1	0.5	1	1	1	1	0	8
P19	0.5	1	1	1	0.5	1	1	1	1	0	8
P20	1	1	1	1	0.5	1	1	1	1	1	9.5
P21	0.5	1	1	1	0.5	1	1	1	1	0	8
P22	0.5	1	1	1	1	1	1	1	1	1	9.5
P23	0.5	1	1	1	0.5	1	1	1	1	0	8
P24	1	1	1	1	1	1	1	1	1	0	9
P25	1	1	1	1	0.5	1	1	1	1	1	9.5
P26	0.5	1	1	1	0.5	1	1	1	1	0	8
P27	1	1	1	1	1	1	1	1	1	0.5	9.5
P28	0.5	1	1	1	1	1	1	1	1	0	8.5
P29	1	1	1	1	1	1	1	1	1	0	9
P30	0.5	1	1	1	1	1	1	1	1	1	9.5
P31	1	1	1	1	1	1	1	1	1	1	10
P32	1	1	1	1	1	1	1	1	1	0.5	9.5
P33	1	1	1	1	0.5	1	1	1	1	1	9.5
P34	0.5	1	1	1	1	1	1	1	1	1	9.5
P35	0.5	1	1	1	1	1	1	1	1	1	9.5
P36	1	1	1	1	1	1	1	1	1	1	10
P37	0.5	0.5	1	1	0.5	1	1	1	1	0	7.5
P38	0.5	1	1	1	0.5	1	1	1	1	0	8
P39	1	1	1	1	1	1	1	1	1	1	10
P40	0.5	1	1	1	1	1	1	1	1	1	9.5
P41	0.5	1	1	1	0.5	1	1	1	1	0	8
P42	1	1	1	1	1	1	1	1	1	0.5	9.5

TABLE 12. *(Continued.)* **Quality assessment score of the selected studies.**

ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
P43	1	1	1	1	1	1	1	1	1	1	10
P44	1	1	1	1	1	1	1	1	1	1	10
P45	0.5	0.5	1	1	0.5	1	1	1	1	0	7.5
P46	1	1	1	1	0.5	1	1	1	1	0	8.5
P47	1	1	1	1	1	1	1	1	1	1	10
P48	1	1	1	1	1	1	1	1	1	0.5	9.5
P49	0.5	1	1	1	1	1	1	1	1	1	9.5
P50	1	1	1	1	1	1	1	1	1	1	10
P51	0.5	1	1	1	0.5	1	1	1	1	0	8
P52	0.5	1	1	1	0.5	1	1	1	1	0	8
P53	1	1	1	1	1	1	1	1	1	1	10
P54	0.5	1	1	1	0.5	1	1	1	1	0	8
P55	0.5	1	1	1	0.5	1	1	1	1	0	8
P56	0.5	1	1	1	0.5	1	1	1	1	0	8
P57	0.5	1	1	1	0.5	1	1	1	1	0	8
P58	0.5	1	1	1	0.5	1	1	1	1	0	8
P59	0.5	1	1	1	0.5	1	1	1	1	0	8
P60	0.5	1	1	1	0.5	1	1	1	1	0	8
P61	0.5	1	1	1	0.5	1	1	1	1	0	8
P62	1	1	1	1	1	1	1	1	1	1	10
P63	0.5	1	1	1	0.5	1	1	1	1	0	8
P64	1	1	1	1	1	1	1	1	1	1	10
P65	1	1	1	1	1	1	1	1	1	1	10
P66	0.5	1	1	1	0.5	1	1	1	1	0	8
P67	0.5	1	1	1	0.5	1	1	1	1	0	8
P68	1	1	1	1	1	1	1	1	1	0	9
P69	1	1	1	1	1	1	1	1	1	1	10
P70	1	1	1	1	1	1	1	1	1	1	10
P71	0.5	1	1	1	0.5	1	1	1	1	0	8
P72	1	1	1	1	1	1	1	1	1	1	10
P73	0.5	1	1	1	1	1	1	1	1	1	9.5
P74	0.5	1	1	1	0.5	1	1	1	1	0	8
P75	0.5	1	1	1	0.5	1	1	1	1	0	8
P76	1	1	1	1	1	1	1	1	1	0	9
P77	1	1	1	1	1	1	1	1	1	1	10

TABLE 13. Comparative study of MLA on malware detection.

ID	Ref.	Platform	Data Set	Modelling Tools	Software Used	Proposed Model (Development)	Classification Method	Analysis Type	Machine Learning Algorithm	Detection Accuracy Rate
P1	[125]	Android applications	Fisher iris, Forensic glass, Japanese credit, Pima Indian Diabetes, M0Droid	Bagging-DT algorithm, Android Package Kit (APK), Dalvik bytecodes, AAPT2	Android OS, Virtual Machine, MATLAB R2016	LinRegDroid: Detection of Android Malware Using Multiple Linear Regression Models-Based Classifiers	Signature-based	Static	DT KNN	92.99 89.15%
P2	[126]	Android applications	Malgenome, Maldroid	API calls, Python	Android OS	Using Machine Learning to Identify Android Malware Relying on API calling sequences and Permissions	Behavior-based	Dynamic	KNN NB SVM DT	98.3% 98.7% 100% 100%
P3	[127]	Android applications	Google Play Store, MalDroid, DefenseDroid	API calls, Reverse Engineered(Jadx-GUI), APK, Dalvik Bytecodes, Python 3.8.12, Androguard	Windows 8	Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms	Signature-based	Static	SVM DT NB KNN	92% 90.12% 88.7% 89.5%
P4	[51]	Network system	CTU-13 (CTU University, Czech Republic)	Python, Scikit-learn (SMOTE, SMOTEENN, ROS), flow-based feature selection, protocol/structure independent	Weka, Jupyter Notebook	Multilayer framework for botnet detection using machine learning algorithms	Behavior-based	Dynamic	KNN	92.20%
P5	[52]	Computer System	Elastic Malware Benchmark for Research 2018 (EMBER2018)	Python, Scikit-learn, AdaBoosted CatBoost, AdaBoosted LightGBM and Optimized LightGBM	Not mentioned	Empirical Measurement of Performance Maintenance of Gradient Boosted Decision Tree Models for Malware Detection	Signature-based	Static	KNN DT NB	88% 91% 88%
P6	[53]	Cloud based service	University of New Mexico (UNM) & University of California (Bare cloud)	Python, LibVMI, DRAKVUF, binary particle swarm optimization (BPSO)	Virtual machine (Ubuntu Linux), Anaconda Navigator	VMShield: Memory Introspection-based Malware Detection to Secure Cloud-based Services against Stealthy Attacks	Behavior-based	Hybrid	N-gram & Meta-heuristic	81.23% to 99.91%
P7	[54]	Network system	25-DGA & UMUDGA	Kullback-Leibner divergence, Jaccard Index, pseudo-random strings	Not mentioned	Algorithmically generated malicious domain names detection based on n-grams features	Signature-based	Static	N-grams	> 96%
P8	[55]	Computer System	Chinese security company (RiSing)	variational inference, neural networks, & stochastic gradient optimization	Not mentioned	Deep Generative Model for Malware Detection	Signature-based	Static	Bayesian & Gaussian	80% to 85%

TABLE 13. *(Continued.)* **Comparative study of MLA on malware detection.**

P9	[56]	Cloud environment	Data collection from VirusShare & VirusTotal sites	Cuckoo sandbox, principal component analysis (PCA), random forest, Chi-square, Python, & Scikit-learn	Not mentioned	Machine Learning based Malware Detection in Cloud Environment using Clustering Approach	Behavior-based	Dynamic	DT	93.60%
P10	[57]	Computer System (Windows)	Windows 10 ISO files	Python, Scikit-learn	VirtualBox (Ubuntu)	Using Dtrace for Machine Learning Solutions in Malware Detection	Behavior-based	Dynamic	DT	94%
P11	[58]	Android application	Kaggle	Not mentioned	Not mentioned	Malicious Application Detection in Android using Machine Learning	Signature-based	Static	DT	93.60%
P12	[37]	Android application	Omnidroid (by AndroPyTool), Android Malware Dataset (AMD), KuafuDet dataset, AndroZoo	Python, Dalvik bytecode, Android Application Package (APK), Androguard, AndroPyTool, TensorFlow	Android OS (Linux), Google Collab	Android Malware Detection Using Static Features and Machine Learning	Signature-based	Static	Gaussian DT SVM	83% 82% 81%
P13	[59]	Android application	DREBIN & AMD (by Arguslab)	Manhattan distance, Python, AndropyTool	Linux	DroidTKM: Detection of Trojan Families using the KNN Classifier Based on Manhattan Distance Metric	Behavior-based	Hybrid	KNN	97.83%
P14	[60]	Computer System	VirusShare, portableapps.com & Windows 7 Ultimate 32-bit directory	Cuckoo Sandbox, JavaScript Object Notation (JSON), Python Pefile, Principal Component Analysis (PCA)	VirtualBox	Forensic Malware Identification Using Naive Bayes Method	Behavior-based	Hybrid	NB	93% (static) 85% (dynamic)
P15	[61]	Android application	Google Play Store, Genome, Drebin & Koodous	Apktool, Python, Bag-of-Words Model, Natural Language Processing (NLP), Information Gain (I.G.)	Not mentioned	IPDroid: Android Malware Detection using Intents and Permissions	Signature-based	Static	NB SVM	92.54% 92.42%
P16	[62]	Computer System	Kaggle	Python, scikit-learn	Linux, VirtualBox, Anaconda3, Jupyter Notebook	A Knowledge-Domain Analyser for Malware Classification	Behavior-based	Dynamic	NB SVM	73.01% 75.97%
P17	[63]	Android application	CICInvesAndMal2019	Principal Component Analysis (PCA)	WEKA 3.8, Ubuntu (Linux)	A Static Feature Selection-based Android Malware Detection Using Machine Learning Techniques	Signature-based	Static	NB SVM DT	88.23% 91.26% 92.90%

TABLE 13. *(Continued.)* **Comparative study of MLA on malware detection.**

P18	[64]	Android application	DREBIN	eXtreme Gradient Boosting (XGBoost), Natural Language Processing (NLP), Python, scikit-learn, panda, numpy	Mint 18.3 Sylvia (Linux)	Evaluation of N-Gram Based Multi-Layer Approach to Detect Malware in Android	Signature-based	Static	N-grams	> 97%
P19	[65]	Computer System	VirusTotal & Virushare	Cuckoo Sandbox, cloud-based sandbox (SNDBOX), Markov model, TF-IDF, Java 8 Update 19, Python 2.7.15	Adobe Acrobat Reader DC 2019, Adobe Flash Player, Microsoft. NET Framework 4.7, Microsoft Office Standard 2010, WinRAR 5.61.	MALGRA: Machine Learning and N-Gram Malware Feature Extraction and Detection System	Behavior-based	Dynamic	N-grams	98.40%
P20	[66]	Computer System	Malware Data Science (small data) & Malshare (large data)	HashingVectorizer, Sklearn, Python, CUDA, Keras, Tensorflow, Gradient boosting	Ubuntu 18.04 (Linux)	TuningMalconv: malware detection with not just raw bytes	Signature-based	Static	N-grams (TuningMalconv)	99.03% (small data) 98.69% (large data)
P21	[67]	Cloud Computing	VirusTotal	Android 7.0 API, Quadratic Programming Problem (QPP)	Not mentioned	Enhanced Android Malware Detection: An SVM-Based Machine Learning Approach	Signature-based	Static	SVM	99.75%
P22	[68]	Computer System	Kaggle	Fireworks algorithm (FWA), Elitism distances,	Windows 10 OS	A Malware Detection Method Based on Improved Fireworks Algorithm and Support Vector Machine	Signature-based	Static	SVM	>80%
P23	[69]	Computer System	DREBIN	RESTful API, Monkey tool	Not mentioned	Malware Detection Based on Feature Library and Machine Learning	Behavior-based	Hybrid	SVM	94.15 = %
P24	[70]	Android application	API calls, Permissions, Intents and combination all of it	AXMLPrinter 2, Baksmali Disassembler, Python 3.7, MD5 hash algorithm, Avira Antivirus, Android Application package (APK)	Android OS	A Framework for Detection of Android Malware using Static Features	Behavior-based	Static	SVM KNN DT	91.96% 95.9% 92.94
P25	[36]	Android (IoT Services-network traffic)	Google Play, Android Malware Genome Project, https://virusshare.com	ten-fold cross validation method	Android emulator, Virtual machine, Wireshark	Bayesian model updating method based android malware detection for IoT services	Behavior-based	Hybrid	Bayesian	96%

TABLE 13. *(Continued.)* **Comparative study of MLA on malware detection.**

P26	[71]	Android application	HelDroid	Python, Scikit-learn, Pandas, NumPy, SciPy, Matplotlib	Jupyter Notebook	Exposing Android Ransomware using Machine Learning	Behavior-based	Hybrid	DT	99.08%
P27	[72]	Computer System	VirusShare, VirusTotal, VX Heaven	PEview	Windows OS	Comparison of malware detection techniques using machine learning algorithm	Signature-based	Static	DT SVM KNN	99% 91% 94%
P28	[73]	Cloud Computing	Malimg	Kullback-Leibler divergence (KL), Gaussian Mixture Models (GMM), Python, Tensorflow	Not mentioned	Malware Detection in Cloud Computing using an Image Visualization Technique	Image-based	Static	Gaussian Mixed	80%
P29	[40]	Computer System (Data Registry)	Windows Registry	Euclidean Distance, Java	Virtual machine, Weka 3.8.2	Clustering Analysis for Malware Behavior Detection using Registry Data	Behavior-based	Dynamic	K-Means	> 90%
P30	[74]	Network system	Kasperski & McAfee	Euclidean distance	Not mentioned	Detection System for Detecting Worms using Hybrid Algorithm of Naïve Bayesian classifier and K-Means	Behavior-based	Dynamic	K-Means NB	88% 81%
P31	[75]	Android application	DREBIN	Android Application Package (APK), Dalvik byte code, dex2oat, & Euclidean, Minkowski, Correlation, Jaccard, Hamming and Spearman distances	Android OS	A performance evaluation on distance measures in KNN for mobile malware detection	Behavior-based	Static	KNN SVM	99.2% 93.9%
P32	[76]	Computer System	Alexa 1M, Bader repo extended	Python, Boruta, R version 3.4.2	MacOS X 10.12.7	Detection of algorithmically generated malicious domain names using masked N-grams	Signature-based	Static	masked N-grams	98.91%
P33	[77]	Computer System	VirusTotal & Malicia	API calls., Levenshtein distance, packed cide, UPX, PECompact, VB.Net	Oracle VirtualBox, Drltrace, Windows 7 OS	A Dynamic Heuristic Method for Detecting Packed Malware Using Naive Bayes	Behavior-based	Dynamic	NB	>90%
P34	[78]	Android application	Xiaomi App Store, AndroidManifest files & dex files	API calls, androguard, Activity, BroadcastReceiver, Service, ContentProvider	Android OS	Android Malware Detection Based on Naive Bayes	Signature-based	Static	NB	87.18%
P35	[79]	Network system	Malicia-project	GPGPU	Windows OS	An efficient detection of malware by naive Bayes classifier using GPGPU	Signature-based	Static	NB	87%

TABLE 13. *(Continued.)* **Comparative study of MLA on malware detection.**

P36	[80]	Android IoT Services	Google Play Store & Chinese App Store	API calls, APK, Dex2jar, blockchain	Android OS	A multimodal malware detection technique for Android IoT devices using various features	Signature-based	Static	NB improved NB SVM KNN	90.5% 98% 95% 92%
P37	[81]	Computer System	VirusShare & AV-Test Engine (Windows portable executable files)	Cuckoo sandbox, genetic algorithm, JSON, API calls, Registry keys, windows system (directories, DLL, EXE)	Windows 7 OS, Ubuntu 16.04, Vmware, Virtual Box	Feature Optimization for Run Time Analysis of Malware in Windows Operating System using Machine Learning Approach	Behavior-based	Dynamic	SVM NB	81.3% 64.7%
P38	[82]	Computer System	DREBIN	Android Package Kit (APK), DEX, xxd, dexdump, aapt, scikit-learn	Android OS, Ubuntu Linux 14.04 LTS	A scalable and extensible framework for android malware detection and family attribution	Signature-based	Static	N-grams	99.2% (small data) 86.2% (large data)
P39	[83]	Android application	DREBIN, AndroTracker, M0DROID	Text mining, bag-of-words, APK & DEX files, ApkReader, BINARY & Augmented Normalized Term Frequency (ANTF) methods	Android OS, WEKA 3.6.1, OpenNLP	Adapting text categorization for manifest based android malware detection	Signature-based	Static	N-grams	94.0% to 99.3%.
P40	[84]	Computer System	RISS of ICL machine learning	Honeypot, Artificial neural networks (ANN)	Windows OS, Virtual machine	Ransomware prediction using supervised learning algorithms	Image-based	Static	SVM	88.20%
P41	[85]	Computer System	Endgame Malware BEnchmark for Research (EMBER)	Honeypot	Not mentioned	Malware detection using honeypot and machine learning	Signature-based	Static	SVM	90%
P42	[86]	Online social network	The Fake Project (collected from Twitter)	Dempster-Shafer-Theory (DST), Python 3.6	Mac OS	Detection of social botnet using a trust model based on spam content in Twitter network	Behavior-based	Dynamic	Bayesian	85%
P43	[87]	Computer System	MALICA (Real-world malware samples)	Temporal dependency network (TDN), Conditional probability distributions (CPDs), Loopy Belief Propagation (LBP), Apache Benchmark	Windows OS	Probabilistically inferring attack ramifications using temporal dependence network	Behavior-based	Hybrid	Bayesian	> 97%

TABLE 13. *(Continued.)* **Comparative study of MLA on malware detection.**

P44	[88]	Computer System	Clean and malware files are scrapped from the Internet	API Calls, Cuckoo Sandbox, Java, JSON, ReportHandler, Python	Windows 7 OS, Virtual Machine, Windows 7, Virtual Box, Weka, Adobe PDF Reader	A novel malware analysis framework for malware detection and classification using machine learning approach	Behavior-based	Hybrid	DT	100%
P45	[89]	Computer System	VX heavens & Windows OS clean files	PEiD	Windows 7 & 10 OS	Malware detection based on string length histogram using machine learning	Signature-based	Static	DT KNN	89% 79.14%
P46	[90]	Android application	Malware Genome Project & Android Malware Dataset (by Arguslab)	Markov Chain, Gaussian Dissimilarity (GD), Logarithmic Gaussian Dissimilarity (LGD), Python, Scikit-Learn	Android OS, Lollipop 5.1 (API 22), Linux OS	Sequencing system calls for effective malware detection in android	Behavior-based	Dynamic	GB	98%
P47	[91]	Computer System (Windows)	Windows PE files & PE parser extracts	API Calls, Value Difference Metric (VDM)	Windows OS	Malware detection using a heterogeneous distance function	Signature-based	Static	KNN	98.80%
P48	[92]	Android application	Google Android Market & virusShare.com	API calls, APK & DEX, Dalvik bytecodes, Python 2.7, Androguard	Android OS, Windows 10 OS	Quick and accurate android malware detection based on sensitive APIs	Behavior-based	Dynamic	KNN & DT	92%
P49	[93]	Android application	Android .APK files (by Malaysian Computer Emergency Response Team-MyCERT)	APK files, Dalvik opcode	Android OS	Android malware detection using machine learning on image patterns	Image-based	Static	KNN DT	80.69% 78.62%
P50	[33]	Android application	Kaggle	Relevance Frequency (RF), Euclidean distance	Android OS	New results on permission based static analysis for Android malware	Permission-based	Static	KNN NB	91% 84%
P51	[94]	Computer System	Github, CNET Download, PE file headers	Python, Opcode	Not mentioned	Malware Detection using Opcode Trigram Sequence with SVM	Signature-based	Static	Linear SVM	98%
P52	[95]	Computer System	Windows 7, Windows XP operating systems, & Cygwin executable files, VXHeavens	Snort sub-signature, Chi-square, CFsSubset, Principal Components, InfoGainAttribute, GainRatioAttribute, Hexdump	Linux Ubuntu 14.04, WEKA	Accuracy improved malware detection method using snort sub-signatures and machine learning techniques	Signature-based	Static	N-grams	> 99.78%

TABLE 13. *(Continued.)* **Comparative study of MLA on malware detection.**

P53	[96]	Computer System	VirusTotal, Windows-based executable files, VX Heaven, Open Malware, VirusSign and VirusShare	Hex Editor (HxD), C#.NET	Windows OS, Virtual machine	Signature-based malware detection using sequences of N-grams	Signature-based	Static	N-grams	100%
P54	[97]	Computer System	Mac OS X malware	Python, Euclidean distance, Principal Component Analysis (PCA), Synthetic Minority Over-sampling Technique (SMOTE), Quadratic Programming (QP), Kernel Smooth (KS)	Mac OS X	Intelligent OS X malware threat detection with code inspection	Behavior-based	Dynamic	SVM	>91% (existing data) >96% (new data)
P55	[98]	Computer System	Real Ransomware	API calls, Cuckoo Sandbox, Python 2.7	Windows 7 OS, Virtual machine, Ubuntu 16.04 LTS	Detecting ransomware using support vector machines	Behavior-based	Dynamic	SVM	97.48%
P56	[99]	Android application	AndroidManifest.xml	Application programming interfaces (APIs), aapt, APK files,	Windows XP, Android OS	A SVM-based malware detection mechanism for android devices	Signature-based	Static	SVM	99%
P57	[100]	Cloud Computing	Microsoft Malware Classification Challenge	Cuckoo Sandbox, Message Digest 5 (MD5), Zero-day attacks	Windows 10, Virtual machine	A Zero-Day Resistant Malware Detection Method for Securing Cloud Using SVM and Sandboxing Techniques	Behavior-based	Dynamic	SVM	93.80%
P58	[101]	Wireless network	VirusShare.com	API calls, Jimple, apktool,	Android OS, Soot	A Dynamic and Static Combined Android Malicious Code Detection Model based on SVM	Behavior-based	Hybrid	SVM	94.38%
P59	[102]	Android application	DREBIN	Active Learning, Expected error reduction, API calls, DroidCat, aapt, K-Best, Python 3.6	Android OS, Oracle	Android malicious application detection using support vector machine and active learning	Behavior-based	Dynamic	SVM	>90 %
P60	[103]	Android application	DREBIN, Android Malware Genome Project	APKtool, Python, scikit-learn, SMOTE	Android OS	Android malicious application classification using clustering	Signature-based	Static	SVM DT NB	92.29% 97.59% 81.01%

TABLE 13. *(Continued.)* **Comparative study of MLA on malware detection.**

P61	[104]	Android application	M0Droid	APKTool, Windows PowerShell, Python	Android OS, Virtual Machine, Weka Explorer	Application of machine learning algorithms for Android malware detection	Behavior-based	Hybrid	SVM KNN	79.08% 80.50%
P62	[105]	Computer System	Windows executables files (https://msdn.microsoft.com)	API calls, One Side Class Perceptron (OSCP), Python, scikit-learn, Sequential Minimal Optimization (SMO)	Virtual machine	Adjusting SVMs for large data sets using balanced decision trees	Behavior-based	Dynamic	SVM SVM + DT	84.02% 89.16%
P63	[106]	Android application	GNOME project	fuzzy c-means clustering (FCM-ANFIS), Adaptive neuro-fuzzy inference system (ANFIS), Android APK, API calls, root mean square error (RMSE)	Android OS	Android malware classification based on ANFIS with fuzzy c-means clustering using significant application permissions	Behavior-based	Dynamic	Adaptive NF (ANFC)	91%
P64	[107]	Computer System	Downloaded event/activity data (from Symantec)	Python 2.7.3, Marmite	Ubuntu Linux 12.04	Marmite: spreading malicious file reputation through download graphs	Signature-based	Hybrid	Bayesian	94%
P65	[108]	Android application	Malgenome project & Google playstore	APK	Android OS, Linux, dalvik virtual machine (DVM)	Android malicious application detection using permission vector and network traffic analysis	Behavior-based	Hybrid	DT	95.56%
P66	[109]	Android application	Android Malware Genome Project & Google Play Store	Monkey, Python 3.4.0, APK	Android OS, Linux, Virtual Machine (VM), WEKA	Malware detection in android based on dynamic analysis	Behavior-based	Dynamic	DT	85.00%
P67	[110]	Computer System	VirusShare (www.virusshare.com),	IDAPro code analyzer	Virtual machine, Ubuntu 12.04(Linux), Windows XP & 7, Weka C4.5, Cuckoo Sandbox	RansHunt: A support vector machines based ransomware analysis framework with integrated feature set	Behavior-based	Hybrid	DT NB	95% 93.73%
P68	[111]	Computer System	Windows Portable Executables (PE32), VirusShare, VirusTotal API, Peframe	Hybrid Intelligence, Self-Organizing Map (SOM), fuzzy rules, CaptureBat, WinDump 3.9.5, RapidMiner	Ubuntu 14.04, Virtual Box 5.0.20, Windows 7, Weka 3.7.13	A deep neuro-fuzzy method for multi-label malware classification and fuzzy rules extraction	Behavior-based	Hybrid	Deep NF	69.44%

TABLE 13. *(Continued.)* **Comparative study of MLA on malware detection.**

P69	[39]	Android application	GNOME project	Adaptive neuro-fuzzy inference system (ANFIS), Dynamic evolving neuro-fuzzy inference system (DENFIS), APK, API calls, Apktool	Android OS	An improved Android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (EHNFC) and permission-based features	Behavior-based	Dynamic	Evolving Hybrid Neuro Fuzzy (EHNFC)	90%
P70	[112]	Computer System	KDDCUP' 99	Open Set Recognition, User to Root Attacks (U2R), sigmoid	Not mentioned	Open set intrusion recognition for fine-grained attack categorization	Signature-based	Static	Gaussian SVM	91.1% 90.1%
P71	[113]	Mobile application	VirusShare.com, Google official market	Android Application Package (APK), AXMLPrinter 2, dex2jar, jd-core	Android 6	A detecting method for malicious mobile application based on incremental SVM	Behavior-based	Dynamic	Incremental SVM	90.50%
P72	[114]	Computer System	MS Word files & MS Excel files	Term Frequency Inverse Document Frequency (TFIDF)	Microsoft Office, Visual Basic for Applications (VBA)	Automated Microsoft office macro malware detection using machine learning	Signature-based	Static	KNN	96.30%
P73	[115]	Android application	Programs & Real-world application	Androidetect, Code injection, Binder, Vector construction, Hook technology, API calls, C/C++, Java	Android OS	Machine learning-based malicious application detection of android	Behavior-based	Dynamic	NB DT	82.5% 86%
P74	[116]	Computer System	Ahmadi, Sami, Virussign & CSDMC	Run Length Encoding (RLE), API calls, Java	Windows OS	Improving malware detection time by using RLE and N-gram	Behavior-based	Dynamic	N-grams	95%
P75	[117]	Computer System	Malimg	Gabor Wavelet, GIST, DWT	Windows OS	Malware class recognition using image processing techniques	Image-based	Static	SVM KNN	98.88% 98.84%
P76	[118]	Computer System	vx_heaven & dynamic link libraries (DLL)	operational codes (OpCode), Graph embedding, Application Programming Interfaces (API), Dynamic Link Libraries (DLL)	MATLAB	Graph embedding as a new approach for unknown malware detection	Signature-based	Static	SVM KNN DT	95.62% 94.83% 92.9%
P77	[119]	Cyber-physical systems	CA Technologies VET Zoo & publicly available data sources	API calls, HookMe, Euclidian & Minkowski distances, Python	Virtual Machine, Oracle VM Virtual Box, Cuckoo Sandbox	Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabelled data	Behavior-based	Hybrid	SVM NB	100% (dynamic) 84.8% (dynamic)

TABLE 14. List of datasets.

No.	Related Work	Dataset Type
1	[53]	UNM
2	[53]	Barecloud
3	[95]	Cygwin executable files
4	[55]	RiSing
5	[85], [52]	EMBER
6	[54]	25-DGA
7	[93]	Android .APK files
8	[83]	AndroTracker
9	[37]	AndroZoo
10	[70]	API calls, Permissions, Intents and combination all of it
11	[81]	AV-Test Engine
12	[119]	CA Technologies VET Zoo
13	[80]	Chinese App Store
14	[63]	CICInvesAndMal2019
15	[94]	CNET Download
16	[116]	CSDMC
17	[51]	CTU-13
18	[78]	Dex files
19	[107]	Downloaded event/activity data
20	[118]	Dynamic link libraries (DLL)
21	[94]	Github
22	[71]	HelDroid
23	[74]	Kasperski
24	[112]	KDDCUP'99
25	[97]	Kitmos
26	[61]	Koodous
27	[37]	KuafuDet
28	[97]	LaoShu
29	[97]	MacVX
30	[66]	Malshare
31	[66]	Malware Data Science
32	[88]	malware files are scrapped from the Internet
33	[74]	McAfee
34	[100]	Microsoft Malware Classification Challenge
35	[114]	MS Word & MS Excel files
36	[37]	Omnidroid
37	[96]	Open Malware
38	[94]	PE file headers
39	[91]	PE parser extracts
40	[111]	Peframe
41	[60]	portableapps.com
42	[115]	Programs & Real-world application
43	[119]	Publicly available data sources
44	[98]	real Ransomware
45	[84]	RISS of ICL machine learning
46	[97]	SMOTE
47	[86]	The Fake
48	[54]	UMUDGA
49	[57]	Windows 10 ISO files
50	[60]	Windows 7 Ultimate 32-bit directory
51	[89]	Windows OS clean files
52	[111]	Windows Portable Executables
53	[40]	Windows Registry
54	[97]	WireLurker
55	[78]	Xiaomi App Store
56	[37], [59], [90]	Android Malware Dataset (AMD)
57	[78], [99]	AndroidManifest files
58	[76], [88]	CLEAN

TABLE 14. *(Continued.)* **List of datasets.**

59	[64], [69], [75], [82], [83], [102], [103], [59], [61]	DREBIN
60	[92], [113]	Google Android Market
61	[61], [36], [80], [39], [108]	Google Play Store
62	[62], [68], [33], [58]	Kaggle
63	[83], [104]	M0DROID
64	[77], [79], [87]	Malicia
65	[73], [117]	Malimg
66	[108], [39], [61], [36], [106], [109]	Malware Genome Project
67	[65], [60], [72], [81], [92], [96], [101], [111], [113], [56], [110]	VirusShare
68	[96], [116]	VirusSign
69	[65], [67], [72], [77], [96], [111], [56]	VirusTotal
70	[72], [118], [96], [89], [95]	VX Heaven
71	[105], [96]	Windows executables files
72	[72], [91]	Windows PE file

ACKNOWLEDGMENT

The authors would like to thank the support of the student Michal Dobrovolny for consultations regarding application aspects.

REFERENCES

[1] J. Landage and M. P. Wankhade, "Malware and malware detection techniques: A survey," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 0181–2278, 2013.
[2] A. H. Galib and B. M. M. Hossain, "A systematic review on hybrid analysis using machine learning for Android malware detection," in *Proc. 2nd Int. Conf. Innov. Eng. Technol. (ICIET)*, Dec. 2019, pp. 1–6.
[3] N. Tarar, S. Sharma, and C. R. Krishna, "Analysis and classification of Android malware using machine learning algorithms," in *Proc. 3rd Int. Conf. Inventive Comput. Technol. (ICICT)*, Nov. 2018, pp. 738–743.
[4] M. Anshori, F. Mar'i, and F. A. Bachtiar, "Comparison of machine learning methods for Android malicious software classification based on system call," in *Proc. Int. Conf. Sustain. Inf. Eng. Technol. (SIET)*, Sep. 2019, pp. 343–348.
[5] G. Shanmugasundaram, S. Balaji, and T. Mugilan, "Investigation of malware detection techniques on smart phones," in *Proc. IEEE Int. Conf. Syst. Comput., Autom. Netw. (ICSCA)*, Jul. 2018, pp. 1–4.
[6] S. Naz and D. K. Singh, "Review of machine learning methods for windows malware detection," in *Proc. 10th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT)*, Jul. 2019, pp. 1–6.
[7] J. Senanayake, H. Kalutarage, and M. O. Al-Kadri, "Android mobile malware detection using machine learning: A systematic review," *Electronics*, vol. 10, no. 13, p. 1606, Jul. 2021.
[8] M. Ashawa and S. Morris, "Analysis of Android malware detection techniques: A systematic review," *Int. J. Cyber-Secur. Digit. Forensics*, vol. 8, no. 3, pp. 177–187, 2019.
[9] B. Kitchenham, "Guidelines for performing systematic literature reviews in software engineering," Dept. Comput. Sci., Keele Univ., Keele, U.K. Tech. Rep., 2007.
[10] D. Moher, A. Liberati, J. Tetzlaff, D. G. Altman, and G. Prisma, "Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement," *Ann. Internal Med.*, vol. 151, no. 4, pp. 264–269, 2009.
[11] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Inf. Softw. Technol.*, vol. 50, nos. 9–10, pp. 833–859, Aug. 2008.
[12] A. Nguyen-Duc, D. S. Cruzes, and R. Conradi, "The impact of global dispersion on coordination, team performance and software quality—A systematic literature review," *Inf. Softw. Technol.*, vol. 57, pp. 277–294, Jan. 2015.
[13] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, Jan. 2009, doi: 10.1016/j.infsof.2008.09.009.
[14] M. Orabi, D. Mouheb, Z. Al Aghbari, and I. Kamel, "Detection of bots in social media: A systematic review," *Inf. Process. Manage.*, vol. 57, no. 4, Jul. 2020, Art. no. 102250, doi: 10.1016/j.ipm.2020.102250.
[15] N. Milošević, "History of malware," 2013, *arXiv:1302.5392*.
[16] I. Zelinka and R. Šenkeřík, "Swarm intelligence in cybersecurity," in *Proc. Genetic Evol. Comput. Conf. Companion*, Jul. 2020, pp. 1313–1342.
[17] *HelpNetSecurity: A Brief History of Malware*. Accessed: Nov. 27, 2021. [Online]. Available: https://www.helpnetsecurity.com/2014/10/20/infographic-a-brief-history-of-malware/
[18] A. Qamar, A. Karim, and V. Chang, "Mobile malware attacks: Review, taxonomy & future directions," *Future Gener. Comput. Syst.*, vol. 97, pp. 887–909, Aug. 2019.
[19] (Apr. 2021). *McAfee: McAfee Labs Threats Report*. Accessed: Aug. 18, 2021. [Online]. Available: https://www.mcafee.com/enterprise/en-us/lp/threats-reports/apr-2021.html
[20] *AV-TEST Institute: Statistic of Malware*. Accessed: Aug. 18, 2021. [Online]. Available: https://www.av-test.org/en/statistics/malware/
[21] I. Kara, "A basic malware analysis method," *Comput. Fraud Secur.*, vol. 2019, no. 6, pp. 11–19, Jan. 2019.
[22] R. Tahir, "A study on malware and malware detection techniques," *Int. J. Educ. Manage. Eng.*, vol. 8, no. 2, pp. 20–30, Mar. 2018.
[23] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *Proc. Int. Conf. Broadband, Wireless Comput., Commun. Appl.*, Nov. 2010, pp. 297–300.
[24] M. Christodorescu and J. Somesh, "Static analysis of executables to detect malicious patterns," in *Proc. 12th USENIX Secur. Symp. (USENIX Secur.)*, 2006, pp. 169–186.
[25] E. Konstantinou and S. Wolthusen, "Metamorphic virus: Analysis and detection," Dept. Math. Roy. Holloway, Univ. London, London, U.K., Tech. Rep., RHUL-MA-2008-02, 2008.
[26] *Tesrex: Five Stages of Malware Attacks*. Accessed: Nov. 28, 2021. [Online]. Available: https://tesrex.com/article/5-stages-malware-attack/
[27] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Transl. Vis. Sci. Technol.*, vol. 9, no. 2, p. 14, 2020.
[28] *Analytics Vidya: Fundamental Knowledge of Machine Learning*. Accessed: Feb. 4, 2022. [Online]. Available: https://medium.com/analytics-vidhya/fundamental-omachine-learning-ada28afa1bd3/
[29] *Vinod Sharma's Blog*. Accessed: Nov. 28, 2021. [Online]. Available: https://vinodsblog.com/2018/03/11/the-exciting-evolution-of-machine-learning/
[30] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Comput. Surveys (CSUR)*, vol. 53, no. 2, pp. 1–33, 2020.
[31] M. Alazab, "Profiling and classifying the behavior of malicious codes," *J. Syst. Softw.*, vol. 100, pp. 91–102, Feb. 2015.
[32] R. Agrawal, "K-nearest neighbor for uncertain data," *Int. J. Comput. Appl.*, vol. 105, no. 11, pp. 13–16, 2014.

[33] D. Ö. Şahın, O. E. Kural, S. Akleylek, and E. Kiliç, "New results on permission based static analysis for Android malware," in *Proc. 6th Int. Symp. Digit. Forensic Secur. (ISDFS)*, Mar. 2018, pp. 1–4.

[34] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[35] B. Zhang, J. Yin, J. Hao, S. Wang, D. Zhang, and W. Tang, "New malicious code detection based on N-gram analysis and rough set theory," in *Proc. Int. Conf. Comput. Intell. Secur.*, vol. 2, Nov. 2006, pp. 1229–1232.

[36] F. Wu, L. Xiao, and J. Zhu, "Bayesian model updating method based Android malware detection for IoT services," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 61–66.

[37] A. Al Zaabi and D. Mouheb, "Android malware detection using static features and machine learning," in *Proc. Int. Conf. Commun., Comput., Cybersecur., Informat. (CCCI)*, Nov. 2020, pp. 1–5.

[38] W. Rhmann and G. A. Ansari, "Use of metaheuristic algorithms in malware detection," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 5, no. 6, pp. 1370–1374, Jun. 2017.

[39] A. Altaher, "An improved Android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (EHNFC) and permission-based features," *Neural Comput. Appl.*, vol. 28, no. 12, pp. 4147–4157, Dec. 2017.

[40] N. A. Rosli, W. Yassin, M. A. Faizal, and S. R. Selamat, "Clustering analysis for malware behavior detection using registry data," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 12, pp. 1–10, 2019.

[41] M. Al Ali, D. Svetinovic, Z. Aung, and S. Lukman, "Malware detection in Android mobile platform using machine learning algorithms," in *Proc. Int. Conf. Infocom. Technol. Unmanned Syst. Trends Future Directions (ICTUS)*, Dec. 2017, pp. 763–768.

[42] T. A. A. Abdullah, W. Ali, and R. Abdulghafor, "Empirical study on intelligent Android malware detection based on supervised machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 4, pp. 1–10, 2020.

[43] U. Baldangombo, N. Jambaljav, and S.-J. Horng, "A static malware detection system using data mining methods," 2013, *arXiv:1308.2831*.

[44] Y.-S. Yen and H.-M. Sun, "An Android mutation malware detection based on deep learning using visualization of importance from codes," *Microelectron. Rel.*, vol. 93, pp. 109–114, Feb. 2019.

[45] S. Sohrabi, O. Udrea, and A. Riabov, "Hypothesis exploration for malware detection using planning," in *Proc. 27th AAAI Conf. Artif. Intell.*, Jun. 2013, pp. 883–889.

[46] P. Vinod, R. Jaipur, V. Laxmi, and M. Gaur, "Survey on malware detection methods," in *Proc. 3rd Hackers' Workshop Comput. Internet Secur. (IITKHACK)*, Mar. 2009, pp. 74–79.

[47] I. A. Saeed, A. Selamat, and A. M. A. Abuagoub, "A survey on malware and malware detection systems," *Int. J. Comput. Appl.*, vol. 67, no. 16, pp. 25–31, Apr. 2013.

[48] G. Liang, J. Pang, Z. Shan, R. Yang, and Y. Chen, "Automatic benchmark generation framework for malware detection," *Secur. Commun. Netw.*, vol. 2018, pp. 1–8, Sep. 2018.

[49] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey," *Inf. Secur. Tech. Rep.*, vol. 14, no. 1, pp. 16–29, Feb. 2009.

[50] M. Goyal and R. Kumar, "The pipeline process of signature-based and behavior-based malware detection," in *Proc. IEEE 5th Int. Conf. Comput. Commun. Autom. (ICCCA)*, Oct. 2020, pp. 497–502.

[51] W. N. H. Ibrahim, S. Anuar, A. Selamat, O. Krejcar, R. G. Crespo, E. Herrera-Viedma, and H. Fujita, "Multilayer framework for botnet detection using machine learning algorithms," *IEEE Access*, vol. 9, pp. 48753–48768, 2021.

[52] C. Galen and R. Steele, "Empirical measurement of performance maintenance of gradient boosted decision tree models for malware detection," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, Apr. 2021, pp. 193–198.

[53] P. Mishra, P. Aggarwal, A. Vidyarthi, P. Singh, B. Khan, H. H. Alhelou, and P. Siano, "VMShield: Memory introspection-based malware detection to secure cloud-based services against stealthy attacks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 10, pp. 6754–6764, Oct. 2021.

[54] A. Cucchiarelli, C. Morbidoni, L. Spalazzi, and M. Baldi, "Algorithmically generated malicious domain names detection based on n-grams features," *Exp. Syst. Appl.*, vol. 170, May 2021, Art. no. 114551.

[55] Y. Fu and Q. Lan, "Deep generative model for malware detection," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Aug. 2020, pp. 2072–2077.

[56] R. Kumar, K. Sethi, N. Prajapati, R. Ranjan Rout, and P. Bera, "Machine learning based malware detection in cloud environment using clustering approach," in *Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2020, pp. 1–7.

[57] A. Mohan K. P., S. Chandran, G. Gressel, T. U. Arjun, and V. Pavithran, "Using dtrace for machine learning solutions in malware detection," in *Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2020, pp. 1–7.

[58] H. Soni, P. Arora, and D. Rajeswari, "Malicious application detection in Android using machine learning," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Jul. 2020, pp. 0846–0848.

[59] D. T. Dehkordy and A. Rasoolzadegan, "DroidTKM: Detection of trojan families using the KNN classifier based on Manhattan distance metric," in *Proc. 10th Int. Conf. Comput. Knowl. Eng. (ICCKE)*, Oct. 2020, pp. 136–141.

[60] B. Ramadhan, Y. Purwanto, and M. F. Ruriawan, "Forensic malware identification using naive Bayes method," in *Proc. Int. Conf. Inf. Technol. Syst. Innov. (ICITSI)*, Oct. 2020, pp. 1–7.

[61] K. Khariwal, J. Singh, and A. Arora, "IPDroid: Android malware detection using intents and permissions," in *Proc. 4th World Conf. Smart Trends Syst., Secur. Sustainability (WorldS4)*, Jul. 2020, pp. 197–202.

[62] O. P. Samantray and S. N. Tripathy, "A knowledge-domain analyser for malware classification," in *Proc. Int. Conf. Comput. Sci., Eng. Appl. (ICCSEA)*, Mar. 2020, pp. 1–7.

[63] A. Sangal and H. K. Verma, "A static feature selection-based Android malware detection using machine learning techniques," in *Proc. Int. Conf. Smart Electron. Commun. (ICOSEC)*, Sep. 2020, pp. 48–51.

[64] T. Islam, S. S. M. M. Rahman, M. A. Hasan, A. S. M. M. Rahaman, and M. I. Jabiullah, "Evaluation of N-gram based multi-layer approach to detect malware in Android," *Proc. Comput. Sci.*, vol. 171, pp. 1074–1082, Jan. 2020.

[65] M. Ali, S. Shiaeles, G. Bendiab, and B. Ghita, "MALGRA: Machine learning and N-gram malware feature extraction and detection system," *Electronics*, vol. 9, no. 11, p. 1777, Oct. 2020.

[66] L. Yang and J. Liu, "TuningMalconv: Malware detection with not just raw bytes," *IEEE Access*, vol. 8, pp. 140915–140922, 2020.

[67] H. Han, S. Lim, K. Suh, S. Park, S.-J. Cho, and M. Park, "Enhanced Android malware detection: An SVM-based machine learning approach," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2020, pp. 75–81.

[68] D. Dong, Z. Ye, J. Su, S. Xie, Y. Cao, and R. Kochan, "A malware detection method based on improved fireworks algorithm and support vector machine," in *Proc. IEEE 15th Int. Conf. Adv. Trends Radioelectron., Telecommun. Comput. Eng. (TCSET)*, Feb. 2020, pp. 846–851.

[69] D. Hu, B. Xu, J. Wang, L. Han, and J. Liu, "Malware detection based on feature library and machine learning," in *Proc. IEEE 3rd Int. Conf. Autom., Electron. Electr. Eng. (AUTEEE)*, Nov. 2020, pp. 205–213.

[70] M. Dhalaria and E. Gandotra, "A framework for detection of Android malware using static features," in *Proc. IEEE 17th India Council Int. Conf. (INDICON)*, Dec. 2020, pp. 1–7.

[71] O. B. Victoriano, "Exposing Android ransomware using machine learning," in *Proc. Int. Conf. Inf. Syst. Syst. Manage.*, Oct. 2019, pp. 32–37.

[72] N. S. Selamat and F. H. M. Ali, "Comparison of malware detection techniques using machine learning algorithm," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 16, no. 1, p. 435, Oct. 2019.

[73] F. Abdullayeva, "Malware detection in cloud computing using an image visualization technique," in *Proc. IEEE 13th Int. Conf. Appl. Inf. Commun. Technol. (AICT)*, Oct. 2019, pp. 1–5.

[74] O. M. Qasim and K. H. Alsadi, "Detection system for detecting worms using hybrid algorithm of Naïve Bayesian classifier and K-means," in *Proc. 2nd Int. Conf. Eng. Technol. Appl. (IICETA)*, Aug. 2019, pp. 173–178.

[75] G. Baldini and D. Geneiatakis, "A performance evaluation on distance measures in KNN for mobile malware detection," in *Proc. 6th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Apr. 2019, pp. 193–198.

[76] J. Selvi, R. J. Rodríguez, and E. Soria-Olivas, "Detection of algorithmically generated malicious domain names using masked N-grams," *Exp. Syst. Appl.*, vol. 124, pp. 156–163, Jun. 2019.

[77] E. M. Alkhateeb and M. Stamp, "A dynamic heuristic method for detecting packed malware using naive Bayes," in *Proc. Int. Conf. Electr. Comput. Technol. Appl. (ICECTA)*, Nov. 2019, pp. 1–6.

[78] J. Pang and J. Bian, "Android malware detection based on naive Bayes," in *Proc. IEEE 10th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Oct. 2019, pp. 483–486.
[79] S. K. Sahay and M. Chaudhari, "An efficient detection of malware by naive Bayes classifier using GPGPU," in *Advances in Computer Communication and Computational Sciences*. Singapore: Springer, 2019, pp. 255–262.
[80] R. Kumar, X. Zhang, W. Wang, R. U. Khan, J. Kumar, and A. Sharif, "A multimodal malware detection technique for Android IoT devices using various features," *IEEE Access*, vol. 7, pp. 64411–64430, 2019.
[81] A. Irshad, R. Maurya, M. K. Dutta, R. Burget, and V. Uher, "Feature optimization for run time analysis of malware in windows operating system using machine learning approach," in *Proc. 42nd Int. Conf. Telecommun. Signal Process. (TSP)*, Jul. 2019, pp. 255–260.
[82] L. Zhang, V. L. L. Thing, and Y. Cheng, "A scalable and extensible framework for Android malware detection and family attribution," *Comput. Secur.*, vol. 80, pp. 120–133, Jan. 2019.
[83] O. Coban and S. Ozel, "Adapting text categorization for manifest based Android malware detection," *Comput. Sci.*, vol. 20, no. 3, p. 383, 2019.
[84] U. Adamu and I. Awan, "Ransomware prediction using supervised learning algorithms," in *Proc. 7th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2019, pp. 57–63.
[85] I. M. M. Matin and B. Rahardjo, "Malware detection using honeypot and machine learning," in *Proc. 7th Int. Conf. Cyber IT Service Manage. (CITSM)*, vol. 7, Nov. 2019, pp. 1–4.
[86] G. Lingam, R. Ranjan Rout, and D. V. L. N. Somayajulu, "Detection of social botnet using a trust model based on spam content in Twitter network," in *Proc. IEEE 13th Int. Conf. Ind. Inf. Syst. (ICIIS)*, Dec. 2018, pp. 280–285.
[87] Y. Yang, Z. Cai, C. Wang, and J. Zhang, "Probabilistically inferring attack ramifications using temporal dependence network," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2913–2928, Nov. 2018.
[88] K. Sethi, S. K. Chaudhary, B. K. Tripathy, and P. Bera, "A novel malware analysis framework for malware detection and classification using machine learning approach," in *Proc. 19th Int. Conf. Distrib. Comput. Netw.*, Jan. 2018, pp. 1–4.
[89] S. K. Sawaisarje, V. K. Pachghare, and D. D. Kshirsagar, "Malware detection based on string length histogram using machine learning," in *Proc. 3rd IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT)*, May 2018, pp. 1836–1841.
[90] A. S. M. Ahsan-Ul-Haque, M. Shohrab Hossain, and M. Atiquzzaman, "Sequencing system calls for effective malware detection in android," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.
[91] M. Jureček and R. Lórencz, "Malware detection using a heterogeneous distance function," *Comput. Informat.*, vol. 37, no. 3, pp. 759–780, 2018.
[92] C. Zhao, W. Zheng, L. Gong, M. Zhang, and C. Wang, "Quick and accurate Android malware detection based on sensitive APIs," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, Aug. 2018, pp. 143–148.
[93] F. M. Darus, N. A. A. Salleh, and A. F. M. Ariffin, "Android malware detection using machine learning on image patterns," in *Proc. Cyber Resilience Conf. (CRC)*, Nov. 2018, pp. 1–2.
[94] A. I. Elkhawas and N. Abdelbaki, "Malware detection using opcode trigram sequence with SVM," in *Proc. 26th Int. Conf. Softw., Telecommun. Comput. Netw. (SoftCOM)*, Sep. 2018, pp. 1–6.
[95] B. M. Khammas, S. Hasan, R. A. Ahmed, J. S. Bassi, and I. Ismail, "Accuracy improved malware detection method using snort sub-signatures and machine learning techniques," in *Proc. 10th Comput. Sci. Electron. Eng. (CEEC)*, Sep. 2018, pp. 107–112.
[96] A. M. Abiola and M. F. Marhusin, "Signature-based malware detection using sequences of N-grams," *Int. J. Eng. Technol.*, vol. 7, no. 4, p. 120, Oct. 2018.
[97] H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K.-R. Choo, "Intelligent OS X malware threat detection with code inspection," *J. Comput. Virol. Hacking Techn.*, vol. 14, no. 3, pp. 213–223, Aug. 2018.
[98] Y. Takeuchi, K. Sakai, and S. Fukumoto, "Detecting ransomware using support vector machines," in *Proc. 47th Int. Conf. Parallel Process. Companion*, Aug. 2018, pp. 1–6.
[99] Y.-F. Lu, C.-F. Kuo, H.-Y. Chen, C.-W. Chen, and S.-C. Chou, "A SVM-based malware detection mechanism for Android devices," in *Proc. Int. Conf. Syst. Sci. Eng. (ICSSE)*, Jun. 2018, pp. 1–6.
[100] S. Kumar and C. B. B. Singh, "A zero-day resistant malware detection method for securing cloud using SVM and sandboxing techniques," in *Proc. 2nd Int. Conf. Inventive Commun. Comput. Technol. (ICICCT)*, Apr. 2018, pp. 1397–1402.
[101] J. Du, H. Chen, W. Zhon, Z. Liu, and A. Xu, "A dynamic and static combined Android malicious code detection model based on SVM," in *Proc. 5th Int. Conf. Syst. Informat. (ICSAI)*, Nov. 2018, pp. 675–801.
[102] B. Rashidi, C. Fung, and E. Bertino, "Android malicious application detection using support vector machine and active learning," in *Proc. 13th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2017, pp. 1–9.
[103] H. Rathore, S. K. Sahay, P. Chaturvedi, and M. Sewak, "Android malicious application classification using clustering," in *Proc. Int. Conf. Intell. Syst. Design Appl.* Cham, Switzerland: Springer: Cham, 2018, pp. 659–667.
[104] M. Kakavand, M. Dabbagh, and A. Dehghantanha, "Application of machine learning algorithms for Android malware detection," in *Proc. Int. Conf. Comput. Intell. Intell. Syst.*, Nov. 2018, pp. 32–36.
[105] C. Vatamanu, D. Teodor Gavrilut, and G. Popoiu, "Adjusting SVMs for large data sets using balanced decision trees," in *Proc. 20th Int. Symp. Symbolic Numeric Algorithms Sci. Comput. (SYNASC)*, Sep. 2018, pp. 223–229.
[106] A. Altaher and O. Barukab, "Android malware classification based on ANFIS with fuzzy c-means clustering using significant application permissions," *TURKISH J. Electr. Eng. Comput. Sci.*, vol. 25, no. 3, pp. 2232–2242, 2017.
[107] G. Stringhini, Y. Shen, Y. Han, and X. Zhang, "Marmite: Spreading malicious file reputation through download graphs," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, Dec. 2017, pp. 91–102.
[108] S. Kandukuru and R. M. Sharma, "Android malicious application detection using permission vector and network traffic analysis," in *Proc. 2nd Int. Conf. Converg. Technol. (I2CT)*, Apr. 2017, pp. 1126–1132.
[109] T. Bhatia and R. Kaushal, "Malware detection in Android based on dynamic analysis," in *Proc. Int. Conf. Cyber Secur. Protection Digit. Services (Cyber Secur.)*, Jun. 2017, pp. 1–6.
[110] M. M. Hasan and M. M. Rahman, "RansHunt: A support vector machines based ransomware analysis framework with integrated feature set," in *Proc. 20th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2017, pp. 1–7.
[111] A. Shalaginov and K. Franke, "A deep neuro-fuzzy method for multi-label malware classification and fuzzy rules extraction," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2017, pp. 1–8.
[112] S. Cruz, C. Coleman, E. M. Rudd, and T. E. Boult, "Open set intrusion recognition for fine-grained attack categorization," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Apr. 2017, pp. 1–6.
[113] Y. Li, Y. Ma, M. Chen, and Z. Dai, "A detecting method for malicious mobile application based on incremental SVM," in *Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, Dec. 2017, pp. 1246–1250.
[114] R. Bearden and D. Chai-Tien Lo, "Automated Microsoft office macro malware detection using machine learning," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 4448–4452.
[115] L. Wei, W. Luo, J. Weng, Y. Zhong, X. Zhang, and Z. Yan, "Machine learning-based malicious application detection of android," *IEEE Access*, vol. 5, pp. 25591–25601, 2017.
[116] F. Mira, W. Huang, and A. Brown, "Improving malware detection time by using RLE and N-gram," in *Proc. 23rd Int. Conf. Autom. Comput. (ICAC)*, Sep. 2017, pp. 1–5.
[117] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," in *Proc. Int. Conf. Data Manage., Anal. Innov. (ICDMAI)*, Feb. 2017, pp. 76–80.
[118] H. Hashemi, A. Azmoodeh, A. Hamzeh, and S. Hashemi, "Graph embedding as a new approach for unknown malware detection," *J. Comput. Virol. Hacking Techn.*, vol. 13, no. 3, pp. 153–166, Aug. 2017.
[119] S. Huda, S. Miah, M. M. Hassan, R. Islam, J. Yearwood, M. Alrubaian, and A. Almogren, "Defending unknown attacks on Cyber-physical systems by semi-supervised approach and available unlabeled data," *Inf. Sci.*, vol. 379, pp. 211–228, Feb. 2017.
[120] H. S. Anderson and P. Roth, "EMBER: An open dataset for training static PE malware machine learning models," 2018, *arXiv:1804.04637*.
[121] *LIEF: Library to Instrument Executable Formats*. Accessed: Jan. 1, 2022. [Online]. Available: https://lief-project.github.io/
[122] *The Malware Wiki*. Accessed: Nov. 29, 2022. [Online]. Available: https://malwiki.org/index.php?title=Main_Page
[123] N. Z. Gorment, A. Selamat, and O. Krejcar, "A recent research on malware detection using machine learning algorithm: Current challenges and future works," in *Proc. Int. Vis. Inform. Conf.* Cham, Switzerland: Springer, Nov. 2021, pp. 469–481.

[124] N. Z. Gorment, A. Selamat, and O. Krejcar, "Anti-obfuscation techniques: Recent analysis of malware detection," in *New Trends in Intelligent Software Methodologies, Tools and Techniques*. Clifton, VA, USA: IOS Press, 2022, pp. 181–192.
[125] D. Ö. Şahin, S. Akleylek, and E. Kiliç, "LinRegDroid: Detection of Android malware using multiple linear regression models-based classifiers," *IEEE Access*, vol. 10, pp. 14246–14259, 2022.
[126] E. Amer, A. Mohamed, S. E. Mohamed, M. Ashaf, A. Ehab, O. Shereef, and H. Metwaie, "Using machine learning to identify Android malware relying on API calling sequences and permissions," *J. Comput. Commun.*, vol. 1, no. 1, pp. 38–47, Feb. 2022.
[127] B. Urooj, M. Ali Shah, C. Maple, M. K. Abbasi, and S. Riasat, "Malware detection: A framework for reverse engineered Android applications through machine learning algorithms," *IEEE Access*, vol. 10, pp. 89031–89050, 2022.

NOR ZAKIAH GORMENT (Member, IEEE) received the bachelor's degree in computer science (major in software engineering) from Universiti Malaya (UM), in 2003, and the master's degree in software engineering from Universiti Putra Malaysia (UPM), in 2018. She is currently pursuing the Ph.D. degree with the Malaysia–Japan International Institute of Technology (MJIIT), Universiti Teknologi Malaysia (UTM).

She was a Multimedia Programmer with Dawama Sdn. Bhd., from 2003 to 2011, and a Software Developer with One Learning Solution Sdn. Bhd., from 2011 to 2013. She is also a Lecturer with the College of Computing and Informatics, Universiti Tenaga Nasional (Energy University—UNITEN), an education institute that is established by the Ministry of Higher Education, Malaysia. Her research interests include software engineering, artificial intelligence, machine learning, and cyber security.

Prof. Gorment is a member of the Malaysia Board of Technologies (MBOT) for graduate and professional technologies.

ALI SELAMAT (Member, IEEE) received the B.Sc. degree (Hons.) in information technology from Teesside University, U.K., in 1997, the M.Sc. degree in distributed multimedia interactive systems from Lancaster University, U.K., in 1998, and the Dr.Eng. degree from Osaka Prefecture University, Japan, in 2003.

He was the Chief Information Officer (CIO) and the Director of Communication and Information Technology with Universiti Teknologi Malaysia (UTM). He was elected as the Chair of the IEEE Computer Society, Malaysia Section, under the IEEE, USA. Previously, he was holding the position of Research Dean of the Knowledge Economy Research Alliance, UTM. He was a Principal Consultant in big data analytics with the Ministry of Higher Education, in 2010, a member of the Malaysia Artificial Intelligence Roadmaps, from 2020 to 2021, and a keynote speaker at many international conferences. He was a Visiting Professor with Kuwait University and few other universities in Japan, Saudi Arabia, and Indonesia. He is currently the Dean of Malaysia–Japan International Institute of Technology (MJIIT), an educational institute that is established by the Ministry of Higher Education, Malaysia, to enhance Japanese-oriented engineering education in Malaysia and Asia, with support from the Government of Japan through the Japanese International Cooperation Agency (JICA) and UTM together with 29 Japanese University Consortium (JUC). He is also a Visiting Professor with the University of Hradec Králové, Czech Republic, and the Kagoshima Institute of Technology, Japan. His research interests include data analytics, digital transformations, knowledge management in higher education, key performance indicators, cloud-based software engineering, software agents, information retrievals, pattern recognition, genetic algorithms, neural networks, and soft computing.

LIM KOK CHENG (Member, IEEE) received the Ph.D. degree in usability engineering from the Department of Computing, School of Engineering, Universiti Teknologi Malaysia.

He started his career as an IT Analyst Consultant in New Zealand, Singapore, and a Malaysian-based company. It is within this period of time that he obtained specific experience and field knowledge in software marketing, testing, and training. With clear interest in academics, he expanded his knowledge by continuing his master's degree, while at the same time conducting multiple research plus development works with the Ministry of Science, Technology and Innovation (MOSTI), in the midst of also tutoring in Universiti Tenaga Nasional (UNITEN), Malaysia, and Asia–Pacific University (APU). He joined UNITEN's permanent academic force as a Lecturer, in 2010 and has been contributing to the university till date, as a Lecturer with the College of Computing and Informatics. He is also working as the Head of Software Engineering Program, the Head of the External Relations Unit, UNITEN, and a Treasurer of the IEEE Malaysia Section Computer Chapter. He has accumulated 15 years of teaching and training experience, produced more than 100 student projects, and has won multiple awards in international innovation competitions.

ONDREJ KREJCAR received the Ph.D. degree in technical cybernetics from the Technical University of Ostrava, Czech Republic, in 2008.

From 2016 to 2020, he was the Vice Dean of Science and Research with the Faculty of Informatics and Management, University of Hradec Králové (UHK), Czech Republic, where he has been a Vice Rector in science and creative activities, since June 2020. He is currently a Full Professor in systems engineering and informatics with the Center for Basic and Applied Research, Faculty of Informatics and Management, UHK, and a Research Fellow with the Malaysia–Japan International Institute of Technology, University of Technology Malaysia, Kuala Lumpur, Malaysia. He is also the Director of the Center for Basic and Applied Research, UHK. At UHK, he is responsible for the Doctoral Study Program in applied informatics, where he is focusing on lecturing on smart approaches to the development of information systems and applications in ubiquitous computing environments.

Dr. Krejcar's H-index is 21, with more than 1800 citations received in the Web of Science, where more than 120 IF journal articles are indexed in the JCR index. He is also on the editorial board of the *Sensors* (MDPI) IF journal (Q1/Q2 at JCR), and several other ESCI-indexed journals. He has been a Management Committee Member Substitute for Project COST CA16226, since 2017. He has also been the Vice Leader and a Management Committee Member of WG4 at Project COST CA17136, since 2018. In 2018, he was the 14th Top Peer Reviewer in multidisciplinary in the world according to Publons, and a Top Reviewer in the Global Peer-review Awards 2019 by Publons. Since 2019, he has been the Chairperson of the Program Committee of the KAPPA Program, Technology Agency of the Czech Republic, and a Regulator of the EEA/Norwegian Financial Mechanism in the Czech Republic, from 2019 to 2024. Since 2020, he has also been the Chairperson of Panel 1 (Computer, Physical and Chemical Sciences) of the ZETA Program, Technology Agency of the Czech Republic. From 2014 to 2019, he was the Deputy Chairperson of Panel 7 (Processing Industry, Robotics, and Electrical Engineering) of the Epsilon Program, Technology Agency of the Czech Republic.