Zohaib *et al. (IJECI)* 2024

International Journal for
Electronic Crime Investigation

DOI: https://doi.org/10.54692/ijeci.2024.0801185

(IJECI)
ISSN: 2522-3429 (Print)
ISSN: 2616-6003 (Online)

Research Article

Vol. 8 issue 1 Jan-Mar 2024

A Comprehensive Study for Malware Detection through Machine Learning in Executable Files

Zohaib Ahmad[1], Ahsan Wajahat[2] and Muhammad Salman Pathan[3]

[1] Faculty of Electronics and Information Engineering, Beijing University of Technology, Beijing, China.

[2] Faculty of information Technology, Beijing University of Technology, Beijing, China.

[3] School of Computer Science, National University of Ireland, Maynooth: IE.

Correspondence Author: ahmedzohaib03@gmail.com

Received: December 18, 2023; **Accepted:** January 07, 2024; **Published:** March 15, 2024

ABSTRACT

Two methods are frequently used to analyze malware and start specimens: static analysis and dynamic analysis. Following analysis, distinct characteristics are retrieved to distinguish malware from benign samples. The detection capacity of malware is contingent upon the effectiveness with which discriminative malware characteristics are retrieved through analysis methods. While conventional approaches and techniques were used inadvertently, machine learning algorithms are now utilized to classify malware, which can deal with the complexity and velocity of malware creation. However, even though a few research papers have been published, recent classifications of signature, behavioral and hybrid machine learning is not introduced well. Based on this demand, we provide a comprehensive analysis of malware detection using machine learning, as well as address the different difficulties associated with building the malware classifier. Finally, future work is addressed to build an effective malware detection system by addressing different malware detection problems.

Keywords: Machine learning, Static analysis, API calls, Ransomware, obfuscation technique malicious software, Dynamic analysis.

1. INTRODUCTION

Despite major improvements in computer security methods and their continual growth, the malware remains the primary threat in cyberspace [1, 2]. Malware investigators analyze malicious samples using techniques from several domains, including program analysis and network analysis, to acquire a better knowledge of their behavior and evolution [3-6]. The first computer malware, known as the brain, was developed in the 1980s and a lot of computers were infected. At that time the speed of malware creation was not very fast because it was the peak of computerization at the time. However, as time passes, a thousand new types of malware are created each day [7]. Due to widespread development of malware nowadays, the most recent malware is significantly more targeted, covert zero-day, and persistent than classic malware, which was open, broad, and performed just once [8-10]. Furthermore, today's malware is quite sophisticated, with the primary goal of exploiting computer system flaws. To get around malware identification and analysis systems, malware authors utilize a variety of obfuscation procedures [11]. Malware authors also used encryption and encoding techniques to create complicated harmful programs such as metamorphic , polymorphic, and packed malware, is extremely difficult to analyze and identify [12-14].

The spreading vectors, which are mentioned in Table 1, are typically used to propagate malware from one system to another. The battle between malware creators and analysts continues. Both sides are creating new methodologies and techniques for malware detection systems concurrently, while the other is building malicious software to breach the detection system to target computer and network resources. The malware researcher analyses knew malware intending to prevent an assault on the computer system [15, 16]. Malware is spotted using one of two approaches: signature-based and behavior-based identification. While signature-based malware recognition methods are quick and effective, obfuscated software easily evades them [17-19]. Behavior-based approaches, on the other hand, outperform obfuscation. The behavior-based approach takes a long time. Not only have methods for detecting malware based on behavior and signatures been developed, but also many hybrid tactics that incorporate the recompenses of both. Hybrid detection methods are intended to overcome the concerns associated with both signature-based and behavior-based methods for detection. Zero-day malware detection is thought-provoking as such malware makes use of the recent susceptibilities that have not yet been discovered [19, 20]. Crackers aim to find vulnerabilities in new software and exploit them to breach the software's security. Since the first malware assault on a computer system, a defense mechanism has been built [21, 22]. Machine learning offers a potential answer to this problem by allowing developers to create malware classifiers that can detect new virus and it’s variant [23, 24]. Various machine learning-based strategies based on supervised and unsupervised algorithms have been suggested in the literature [16, 24].

Two main aspects arise after the evaluation of the proposed machine-based detection methods. For the testing of malware, the first stage is the development of classification algorithms for the classification device and the second step is the extraction of malware using a dynamic and static approach. These two variables affect the precise classification of malware. Both the NB, DT, SVM, and ensemble classification technology RF and Ada boosting have been utilized and enhanced for classification training. Classifying ensembles usually provide better results [25]. The benefits and limitations of each categorization algorithm include. In addition, the representation of the feature greatly changes the detection rate of the classifier. One needs a far more reliable automated malware detection technology. Some academics have created automated cognitive analytic methods for addressing the extremely disastrous zero-day malware that can also resist malware assaults. Continuous study of malware is crucial to update techniques for detecting new malware patterns and behavior and variants in existing malware.

Malware authors also used encryption and encoding techniques to create complicated harmful programs such as polymorphic, metamorphic, and packed malware, which is extremely difficult to detect and analyze [12-14]. The spreading vectors, which are mentioned in Table 1, are typically used to propagate malware from one computer system to another. The battle between malware creators and analysts continues. Both sides are creating new methodologies and techniques for malware detection systems concurrently, while the other is building malicious software to breach the detection system to target computer and network resources. The malware researcher analyses knew malware intending to prevent an assault on the computer system [15, 16]. Malware is spotted using one of two methods: signature-based detection or behavior-based detection.

While signature-based malware recognition methods are quick and effective, obfuscated software easily evades them [17-19]. Behavior-based approaches, on the other hand, outperform obfuscation. The behavior-based approach takes a long time. Not only have methods for detecting malware based on behavior and signatures been developed, but also many hybrid tactics that incorporate the advantages of both. Hybrid detection methods are intended to overcome the issues associated with both signature-based and behavior-based methods for detection. Zero-day malware detection is thought-provoking because such malware makes use of the recent susceptibilities that have not yet been discovered [19, 20]. Crackers aim to find vulnerabilities in new software and exploit them to breach the software's security. Since the first malware assault on a computer system, a defense mechanism has been built [21, 22]. Machine learning offers a potential answer to this problem by allowing developers to create malware classifiers that can detect new virus and it’s variant [23, 24]. Various machine learning-based strategies based on supervised and unsupervised algorithms have been proposed in the literature [16, 24]. Two main aspects arise after the evaluation of the proposed machine-based detection methods.

For the testing of malware, the first stage is the development of classification algorithms for the classification device and the second step is the extraction of malware using a dynamic and static approach. These two variables affect the precise classification of malware. Both the NB, DT, SVM, and ensemble classification technology RF and Ada boosting have been utilized and enhanced for classification training. Classifying ensembles usually provide better results [25]. The benefits and limitations of each categorization algorithm include. In addition, the representation of the feature greatly changes the detection rate of the classifier. One needs a far more reliable automated malware detection technology. Some academics have created automated cognitive analytic methods for addressing the extremely disastrous zero-day malware that can also resist malware assaults. Continuous study of malware is crucial to update techniques for detecting new malware patterns and behavior and variants in existing malware.

Table 1 : Methods for Spreading

Methods for Spreading	Features
Drive-by Download	Unintentional drive-by downloads point to a malware infection that causes damage to users in a variety of ways. Cybercriminals steal and gather personal information and get all the account credentials via drive-by downloads. For example, their banking information such as usernames and passwords may also include Trojans or exploit kits that may be used to spread other malicious targets.
Vulnerability	A vulnerability is a security hole in a device's or program's software that allows an invader activity to insert malware into the system. It may be a fault arising due to programming in an application or device software, design fault, or some other form of inbuilt flaw. A very successful WannaCry (2017) ransomware exploited Windows 7's vulnerability to encrypt millions of users' files. This malware exploited an existing weakness in the Windows 7 OS SMB. The user who mended the susceptibility did not affect them, while the remaining lost their data. It is a perilous kind of vector of propagation that is very hard to deal with. An invader discovers a weakness in the OS or application and attempts to create malware to exploit the existing defect to do the most harm. As a result, to handle various types of malwares, the user must update the system periodically.
Backdoor	A backdoor denotes any technique that enables allowed and unauthorized operators to utilize regular security mechanisms to increase high-level user access and to a device, network, or software program (aka root-access). Cyber thieves will exploit a loophole when the information is stolen, further software is installed, and the machine is hijacked.
Removable Drives	Nowadays, removable drives are available on both flash discs and hard discs. These are the most prevalent malware distribution techniques for each device. Despite the existence of anti-malware software on the infected computer, it enables infection propagation and connects the infected device to the mobile drive. Users should always take care while transferring data between computers using flash devices. It can transmit any kind of malware, including viruses, worms, and ransomware.
Homogeneity	The setup of similar OS software connected via the same network becomes the source of worm virus spreading from one machine to another.

1.1. Contributions

The current state and evolution of malware detection systems are discussed in this research study.

1. Many classifications technique for machine learning is explored and compared.
2. Recent classifications of signature, behavioral and hybrid machine learning are explored. It shares with its advantages and limitations a fraction of the data in the proposed malware detection systems.
3. The present study has covered some important parameters that influence malware classifier performance. A hybrid model for malware detection has been presented utilizing machine learning is also provided. Finally, the paper was completed, and the topic of the future directive was discussed.

1.2. Scope Overview

The emphasis of this study is based on the detection of malware using ML techniques, and to create the executable files system's automated smart malware detection. This article examines the work suggested for the identification of executing files and provides information on current research on malware revealing via numerous characteristics and methods.

1.3. Evolution

There is a complete review of machine learning malware identification methods. Owing to significant variances in the number of data sets used, ML algorithms, and valuation processes, the detection technique provided are very difficult to compare properly. The results of suggested ML-based malware classificatory are nevertheless equated and presented with certain in this research.

1.4. Orgnization

The rest of the paper is systematized as follows. Section 2 introduces systems for malware research. Section 3 debates the machine learning methods employed to categorize malware. In Section 4 static, dynamic, and hybrid, the analysis of different approaches is provided. Section 5 assesses the result of the examined papers, and the many criteria for malware classification are explained in section finally it illustrates the possible breadth and concludes the study.

2. MALWARE ANALYSIS

Different malware data samples are investigated to obtain results that can be utilized to detect them. Static and dynamic studies are two basic malware research methodologies that are illustrated below.

2.1. Static Malware Analysis

In the study, features are gathered without running the malware sample. Following the analysis by the extraction of several static features such as N-grams, hash value, strings, opcodes, and PE header information. The development of malware revealing software (antiviruses, intrusion detection systems etc. is based on these characteristics [26, 27]. Security analysts examine malware samples either by using reverse engineering or not. The malware files are dismantled and converted into assembly language code, which is then used to test the malware sample during the coding stage. Some of the most widely used IDA Pro, Ollydbg, WinDbg, and capstone disassemblers are among the most widely used disassemblers and debuggers [33, 34]. An examination of the assembly code is performed to discover the processing route of a malicious operation file, pattern and structure. This information may be used to detect new or variant malware. To study the assembly language code to identify the execution functions is a time-consuming procedure. The use of code obfuscation practices makes the forecaster's work even more difficult. Malware authors employ a variety of methods to escape malware inspection, including code encryption, code reordering instructions, and dead code insert techniques [28, 29]. Static analysis techniques are defined briefly in Table 2, which is followed by a discussion of the methodologies.

Table 2: Static analysis tools

Tool name	Description
PeView [44]	The 32-bit Portable Execution File (PE) structure and content of files, as well as the Component Object File Format, may be viewed quickly and simply with this tool (COFF). This PE/COFF file reader reads headers, sections, directories, import tables, and tables for exporting, and resource data numerous files (such as EXE, OBJ, DLL, DBG and LIB) as well as information for resource purposes.
PEid [45]	Used to test if the malware is hideous if the packer tool is used (for example NSPACK, UPX, etc.). The creators of malware are increasingly utilizing antivirus methods to disguise the true malicious code. The packaged malware may be classified using PEid.

CFF Explorer [46]	This provides the executable file with complete header information and metadata. In this application, you may view more detailed information about executable files.
PsFile [47]	This useful tool shows details about the system's open files. Analyzing if the computer device is remotely controlled is very beneficial. It can send details about opened files to the remote computer on the local machine.
Accesschk [48]	To evaluate the degree of security of the equipment, Accesschk is utilized. It contains information on access rights, including whether a group or user can write, read, or accomplish, among other things, registry keys and files.
IOC Finder [49]	(IOC) Finder is a free utility that collects host system data and reports IOC presence. IOCs are open-standard XML documents that assist incident responders in capturing a variety of threat information.
Radare [50]	Radare is a comprehensive set of tools for reverse engineering. This utility is available on a wide range of platforms including Windows, Linux, Android, and MacOS. On the file system, Radare can also do forensics.
Yara [51]	The Yara tool is employed in executable files to match a string. To recognize the malware file, such string signatures may be used in malware analysis. The Yara tool has the capability of matching a certain string pattern contained inside a binary file. Other file formats, such as PDFs, Word documents, and other similar documents may be matched using this feature.
SS Deep [52]	Executed fuzzy hash values needed to verify malware variants may be calculated in this utility. The fuzzy hash contains more potential to compact with malware variations, unlike the simple hash value.
Disassemblers [44]	Disassemblers are then employed if a more comprehensive static analysis is needed. IDA Pro is a well-known and widely used disassembler. To effectively carry out the reverse engineering task, a new Ghidra disassembler was constructed. The assembly code that is transformed from the executable file, which is then examined manually to determine the functionality of the malicious software.

2.2. Dynamic Malware Analysis

When using this procedure for investigation malware the files of malware are processed and the resulting malware running time behavior is captured and analyzed. There are many types of runtime behavior including file system, processing execution, modification to registry key, and network activity [30-33]. Dynamic inspection differs from static examination since it relies on the connection amid malware and the windows operating system. To ensure system security, the execution of malware samples is always carried out in a simulated environment, since if the malware program's file to be run direct on the host machine, it would cause damage to the operating system [34-37]. The program Virtual Box or VMware, which allows you to create a virtual environment on a computer, is called virtualization software. There are a variety of behaviors that can be observed when a malware file is executed, including the formation of new program's files, the removal of a system file, modification of a registry key, creation of new log entries, API calls, visiting of URLs, the installation of malware, and the transmission of information to the command-and-control scheme. The following steps determine if the file is benign or dangerous based on its contents. Dynamic analyses may be used to investigate files that were not properly deconstructed or evaluated by static analysis. Table 3 provides a high-level summary of the various dynamic analysis techniques.

Table 3: Dynamic analysis technologies

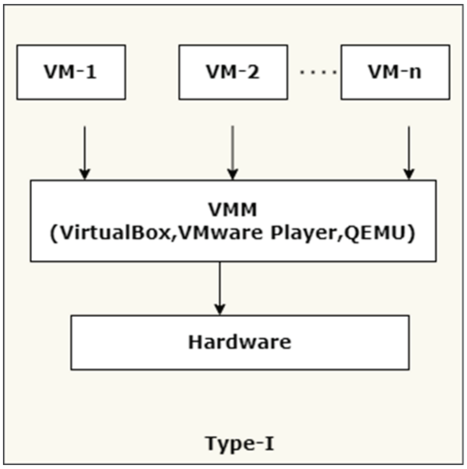
Name of Tool	Features
Process Explorer [9]	The system's task and control manager is called Process Explorer. It functions similarly to a Windows task manager by offering comprehensive details on the active system processes.
ProcMon [57]	The tasks performed by running processes are recorded by ProcMon (Process Monitor). It collects all operating of file system, registry updates, memory activity, and network operations system calls.
Wireshark, Tshark [58]	Network traffic analysis is used by Wireshark and Tshark. These tools can internment all inward and departing packets from the system to the outer world. Different system characteristics including port addresses, URLs, and data streams are collected and subsequently analyzed for malware file classification.
TCPdump [59]	Network data analysis in real-time may be accomplished via the use of this command-line application. It is a commonly used packet analyzer that shows in real-time the TCP/IP packets are sent and received from a computer system.
TCPview [60]	Windows networking tool that displays system statistics from every endpoint in UDP and TCP.
Regshot [61]	To capturing the registry modifications caused by the sample, the tool Regshot is utilized. The registry state of the Windows operating system will be captured both before and after the malware program sample is implemented. These both states may be equated to identify which changes have been made by the sample file that has been run.
Memoryze [62]	Memoryze is a command-based forensic memory tool that may be used to examine digital memories. There may be a full memory dump. Once the malware has been run, the memory dump is deleted from the computer. The memory dumps of rootkit malware are analyzed to extract different features such as the processes that are currently executing, strings, and the process that is being concealed from view.
Volatility [63]	Volatility is a cutting-edge framework for memory dump analysis that is constantly evolving. It is written in Python and has been built for various operating systems so that it may be used simply (Linux, Windows, and Mac OS). Advanced memory dump features like processes, registry keys, DLL-injected libraries, and strings may all be recovered with this tool, among other things.
Redline [64]	Redline is a portable manager that automatically collects information to examine the IOC. It is designed to be portable (Indicator of Compromise). Safety analysis software is used to examine several Windows components, including memory, the file system, the network, and registry entries.
Inetsim [65]	The network simulator is the component that imitates internet services to spread malware for virtual internet communication, as previously explained. It is Inetsim that supplies the malware with its virtual network environment, allowing the malware to perform its functions properly throughout the dynamic analysis process. Consider the possibility that inetsim may respond to such a query and that the malware will be permitted to continue to execute if the malware tries to connect to the remote device.
Fake DNS [44]	The FakeDNS software responds to DNS queries by generating a response. FakeDNS also resolves the virus's DNS requests, allowing the malware to handle them in real-world situations once the DNS queries have been resolved. Using the inetsim and FakeDNS programs, you may simulate virtual networks to better understand how the virus behaves.
Apate DNS [66]	ApateDNS, improved form of FakeDNS that incorporates a graphical user interface. When compared to the Fake DNS tool, Apate DNS is easier to set up and evaluate DNS responses.
Sandboxes [67]	For automated malware analysis, a variety of sandboxes, such as Cuckoo, Anubis, Panda, Limon, Parsa, and others, are utilized. These tools are composed of a variety of embedded analytical tools, such as those mentioned above. To remove the peculiarities of various groups, sandboxes and other tools are used in conjunction with one another, such as tcpdump for network traffic, the Cuckoo sandbox, volatility of memory dump, and so on.

2.3. Dynamic Analysis Framework

The dynamic analysis framework is configured to record the behavioral features of the executable files using hypervisor Type-1 and Type-2 [38, 39]. Type-1 hypervisors are recognized as bare-metal hypervisors because they handle and monitor guest computers directly on the hardware computer. Nutanix AHV, AntsleOs, VMware ESXi, XEN, Oracle VM Server Microsoft Hyper-V, are the patterns of type-1 hypervisors. Fig. 1 (a) displays the type-1 hypervisor architecture. Type-2 hypervisor run within the guest machine tools including Virtual machine ware, Virtual Box and Microsoft Hyper-V are employed to provide a dynamic analysis environment for type 2. VMware Player, Virtual Box, VMware Workstation, QEM, and so on are examples of type-2 hypervisors. Fig. 1 (b) demonstrates the type-2 architecture. In malware execution, the abstraction level is provided by these figures. Both kinds are based on the benefits and limitations stated in Table 4. The difference between methods for dynamic and static analysis based on their merits and demerits is given in Table 5.

Table 4: Hypervisor comparison for dynamic analysis

	Type-1	Type-2
Architecture	Virtualization of hardware	Most deprived code that works on low-end hardware
Detection methods	CPU Background performance low but not zero overhead	VMs have similar difficulties as emulators, although they may be more transparent.
Benefit	Close to hardware negligible overhead	Easy to use introspection and state control
Drawback	Lower introspection ability but measurable	Conceived for transparency compatibility
Protected	It is protected	little, if a host issue that affects the whole OS, like hypervisor,
Scalability	The scalability may be improved.	Less, reliant on host OS



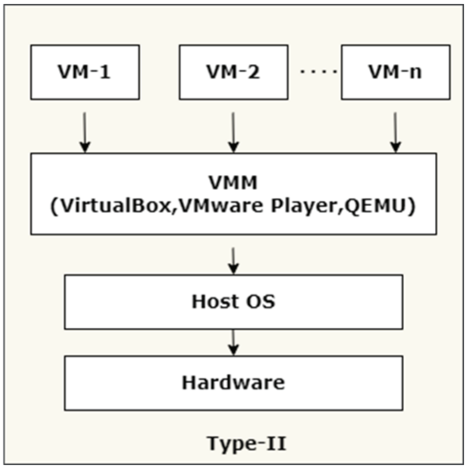


Fig. 1. (a) Hypervisor Type-I architecture

(b) Hypervisor Type-II architecture

Table 5: Brief comparison of static and dynamic analytical approaches

	Static	Dynamic
Methodology	The study is performed without the need to run the files in question.	Analysis takes place with running the files
Benefits	Quick and little time-consuming	Robust for handling obscure methods
Drawbacks	Unable to determine new malware, obscure techniques may simply be circumvented.	Time-consuming and complicated

3. MACHINE LEARNING FOR MALWARE DETECTION

In detecting and grouping malware, machine learning is becoming extremely helpful. There has been a great deal of effort in the literature to categorize benign and malicious files. ML technology offers additional options and flexibility to construct a more precise model by allowing for more properties of a malware and a benign file [26, 40]. In the realm of computer security, ML delivers a variety of options for the revealing of malware, infiltration, and harmful URL identification. Malware samples are examined, and the data collected for the training of the classifier is utilized. Fig. 2 displays the machine learning framework for the detection of benign and malignant files. This figure illuminates the simple building of ML classifiers, as well as the construction of classifiers in other problem areas. First step is feature extraction. After that feature representation and selection are completed. Last step is classification methods are employed to train malware classification models. Another advantage of employing ML in malware revealing is that it may be used to develop a model for identifying previously undiscovered malware. There is reasoning that it consists of many methods that may be used to create several malware algorithms to better identify malware. Furthermore, the following perks of employing ML algorithms to detect malware are some of its many uses.

- Current anti-viruses and sandboxing technique can be ousted.
- Automatically extracts samples of malware.
- The detection of unknown variants can be further generalized.
- It has the potential to decrease human efforts and time spent studying malware.

Many researchers have already carried out extensive research, demonstrating their high level of reliability regarding detecting malware.

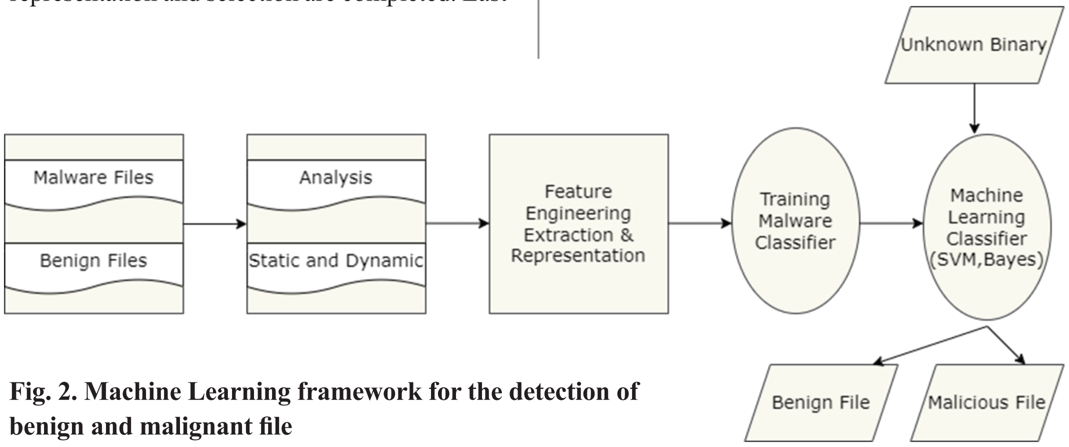

Unknown Binary
Malware Files
Benign Files
Analysis
Static and Dynamic
Feature Engineering Extraction & Representation
Training Malware Classifier
Machine Learning Classifier (SVM,Bayes)
Benign File
Malicious File

Fig. 2. Machine Learning framework for the detection of benign and malignant file

3.1. Implementing ML Challenges in Malware Identification

There are two main hurdles to implement ML in malware revealing, discussed in the subsequent sections.

3.1.1. High Computation Cost

Machine learning must be trained and upgraded to overcome the first barrier to using it in malware classifiers. Malware indicators must be restructured frequently to be effective. Computer views, where computer education has successfully been used, unlike other NLP domains, however, the classifiers often need to be re-trained to identify new and mutated viruses in this domain. According to the finding of the research everyday thousands of new viruses are created, and malware changes its behavior in a matter of hours or minutes. In contrast to other approaches, ML is thus more expensive and complex. As a result, while using machine learning for malware revealing, we must think in a different way.

3.1.2. Adversarial Machine Learning

In the field of computer-based malware categorization, adversarial machine learning is a major concern. The use of machine learning technologies by malware writers to escape malware detectors is critical for the development of unfavorable machine learning. In addition, Kolosnjaji et al. (2016) [41] pointed out that it was feasible to circumvent the machine-learning revealing method provided by the authors [42] with the aid of an intelligent escape assault. It is a fact that, apart from machine learning, there is no other technique available for detecting the most recent and greatly complicated malware. The development of malware is also moving at a breakneck speed. Now the Question is how we can hold these tests to employ machine learning in cyber security Realm. We may choose to minimize the dimensionality of the dataset to reduce training costs since machine learning algorithms take longer to learn from a dataset that includes more data characteristics than necessary. Consequently, only the most useful and discriminating malware features may make use of function selection and size reduction techniques to accomplish this. The second problem of an adversary's machine learning may be fixed by creating fusion malware classifiers. Fusion classifiers may utilize both dynamic and static information in their classification. It is possible to train some classification algorithms once the attributes have been extracted. A single machine learning method may be used to bypass the malware detector; however, the malware detector ensemble may be more resistant to unfavorable machinery learning than a single ML algorithm. Machine learning algorithms such as the vector support machines (SVM), Naive Bayes (NB), Random Forest (RF), and Decision Tree (DT), as well as set algorithms such as Random Forest (RF), and others in the classification literature have been used to train classification models. Machine learning techniques such as k-Nearest Neighbors (KNN) and others are also used to train classification models (ADA). ML classification methods are briefly discussed in Table 6, which includes a brief contrast of the different techniques.

Table 6: Description of classification methods for machine learning

ML Methods	Depiction	Benefits	Drawbacks
NB [43]	The likelihood to each class and the dependent possibility of every database within each class are computed using the NB method. To predict the probability of a class occurring in each data instance collection, the cumulative likelihood of the class is estimated by measuring the class chance and conditionally likelihood of every backing instance of data. NB can be utilized for both multi-class and binary classification.	It is quite easy to use and comprehend the NB classification method. It can operate well with non-relevant data. Furthermore, a tiny dataset may be used in the classifier.	The major disadvantage of the NB classification is that when the data features in training data are correlated, it is poorly performed. According to NB, data components should be autonomous.
K-NN [79]	The K-Nearest Neighbor (K-NN) classification technique divides the instance input into classes based on the class labels of the k-instances that are closest to the input instance. It is anticipated that the class of the input instance will correspond to the class of the majority. If it is necessary to catch the class label of the instance of input from the adjacent K occurrences, distance measures such as Euclidean, Manhattan, Hamming, and Minkowski will be employed.	KNN is simple to construct as novel instances with well-defined class labels, and can be restructured at a low rate. There are no assumptions on the data in the KNN algorithm. Search space is more resilient; thus, the data set is not linearly separable.	The major fault of the k-NN method is that, does not work when a data set is spread randomly. It is moreover uncertain to choose a suitable value for k.
SVM [89]	A hyper plane is used to split data instances into various classes in the data set entry by the SVM algorithm. A point vector in a two-dimensional space input may be seen to divide the instance of input data into two benign and the binary class. For correctly categorizing classes, usage of kernel functions in SVM classification training is essential. SVM classifiers utilize linear, radial, and poly kernel features often.	SVM is the most promising technique for classification since it provides high accuracy while yet being simple to use. SVM is capable of handling large datasets with multiple dimensions. It may also categorize separable non-linear data. Each issue has a regularization parameter and kernel function.	When the penalty parameter is set to a high value, the training time for SVM becomes very lengthy (C). Furthermore, the selection of the C value involves a balancing act between a test error and a training miscalculation.
LR [81]	As a parametrical binary classification technique, Logistic Regression is used to categorize data and divide it into groups. To build a logistic regression classifier, LR acquires the quantities from the training samples. In a qualitative response model, the likelihood ratio (LR) is employed to estimate the empirical parameter value.	It is less difficult to analyze and less complicated to use. The independent variables do not need to have identical variances, nor must a normal distribution with equal variances be used. In addition, since there are no linear connections between the independent and dependent variables, it may be used to cope with non-linear effects.	On average, the accuracy of LR's predictions is low. It has a lot of undesirable qualities that must be dealt with.

DT [90]	A DT is built by calculating the information achievement of each attribute in a dataset and applying that information gain to the decision tree classification. The root is the most important feature for obtaining knowledge. After then, the other transforms into a leaf of the plant's root. DT that has been created is then used to forecast the classes. Each node within the decision tree (not leaf) verifies the function, the function value correlates to the branch of the decision tree and a leaf-node is a class mark. The model has two division functions: information gain and Gini index. The model is trained.	High dimensional datasets and bright data may be handled using DT classifiers. In contrast to KNN and SVM classifiers, it operates in the white case. It is feasible to do a trained interpretation. This enables the trained model to be analyzed in depth. DT classification also has a high workout pace.	A minor modification of the dataset may lead to a big change in the decision tree structure, which will make the model unstable. It does not work properly because of the limited amount of data characteristics.
ANN [73]	Artificial Neural Networks (ANNs) represent standard techniques that help identify decision limits while reducing error rates in the same manner that human brains do.	ANN may be used to model a non-linear dataset with a high number of input characteristics. ANN may be utilized to solve virtually any issue, especially the optimum problem.	Over-fitting may be a problem for ANNs. The weights for training data may not be computationally costly for additional data sets, even though they are of the same demographics as the training data.
RF [91]	Random Forest is a machine learning method that makes use of bagging to improve performance. DT produces a single decision tree, while RF creates multiple decision trees based on separate sub-sets of the dataset with replacement, each of which is different from the previous one. The result of the RF is determined by the votes of each tree.	Since the RF learning machine algorithm randomly chooses various subsets, it is immune from data set variations and therefore minimizes the danger of overfitting. This provides the best results in categorization. It may provide excellent results, but the dataset varies, as opposed to the decision tree.	The training pace is sluggish. The number of classifiers that must be trained to create a powerful classifier is directly proportional to the strength of the classifier. Unlike decision-making book since many decision-making bodies are designed to produce a strong modal outcome. Because of this, it is challenging to assess the consequences.
Boosting Algorithms [67]	Alternative approach to ML is to boost processes that are used to train many weak classifiers sequentially. Without a replacement for sub-sets of data, weak classification is learned linear. The average of all weak graduates is then utilized to build a strong grading system. Another kind of algorithm is boosting machines. Boosting varies in many ways from bagging. Bagging subsets are chosen at random from the entire data samples, while the training subsets are designated from previously unselected data. Adaptive and Gradient boosting (GB) are two boosting approaches that are often utilized (AdaBoost).	Boosting techniques with any dataset works well. Boost techniques, like the RDF, address the variation in data characteristics. To train the week classifiers, several fundamental classification methods for machine research (SVM, KNN, DT) may be employed in conjunction with boost approaches before edifice the ultimate classifier.	Enhancing systems may be time-consuming and computer consuming. Increased algorithms on a real-time platform are tough to implement. A long and time-consuming computation is required for malware analysis since the classification method must be used to classify millions of malware samples.

4. MALWARE DETECTION TECHNIQUES

The objective is to detect and defend from harmful programs that could harm the computer systems or network properties. The input is evaluated in various instances so that malware samples are detected and classified into an appropriate family. It is necessary the use of specialized harmful file knowledge or expertise that accurately reflects the actual malware file behavior. Consequently, numerous malware program samples are evaluated utilizing dynamic, static and hybrid systems, depending on their complexity. It is then displayed properly to help with the continued training of the revealing system. According to authors [43], many distinct characteristics are used by several malware detection researchers, including Opcodes, strings, PE header information , API calls, ,Windows registry, files system accessing AV/Sandbox submissions, network Activities, and generated exceptions. Malware programmed files are assessed, and their characteristics are extracted and presented in an intermediate format prior to initiating malware detection. This intermediate form shows an important function in revealing. The false-positive rate will only be decreased if the collected data is optimal for malware revealing. The numerous malware revealing approaches presented are categorized into three primary classes, as mentioned below.

4.1. Signature-Based Malware Detection

This technique used to identify malware that is characterized by a certain file pattern and signature. It is a standard way of detecting known harmful files quickly, Compared to other techniques. Signature-based methods are often used in the development of antivirus software. One easy technique to construct the signature of malware program files by utilizing a hash algorithm such as message-digest algorithm (MD5), Secure hash Algorithm (SHA1) and others. Malware file signatures are generated and kept in the detection system database to check the unknown program file signature. It is assumed that if the signatures of the harmful file match, formerly it will be avowed as virus else the benign file. There is a problem with this because, if just one byte of a file code is modified, the signature of the program file will also be altered. It follows that a new signature must be created for each new malware variant and each upgraded malware variant. Only after that will a malware detector be able to identify this malware. In addition, new malware detection techniques based on signatures have been proposed, which make use of a variety of models, including program, graphic flow control, and mnemonic sequences. The malware data samples are investigated by means of a range of IDA Pro, Preview, PeStudio, and functional tools, among other tools, to determine if they are malicious. Previously mentioned standard and master learning techniques are utilized to train the malware detection system, which in turn is taught using the characteristics that were extracted. All the various methods to creating signature malware classifiers that are based on static characteristics is shown in Fig. 3. When it comes to developing malware detectors, there are many tools, static features, and methods to choose from. All the techniques discussed in this section based on signatures were proposed using this architecture.

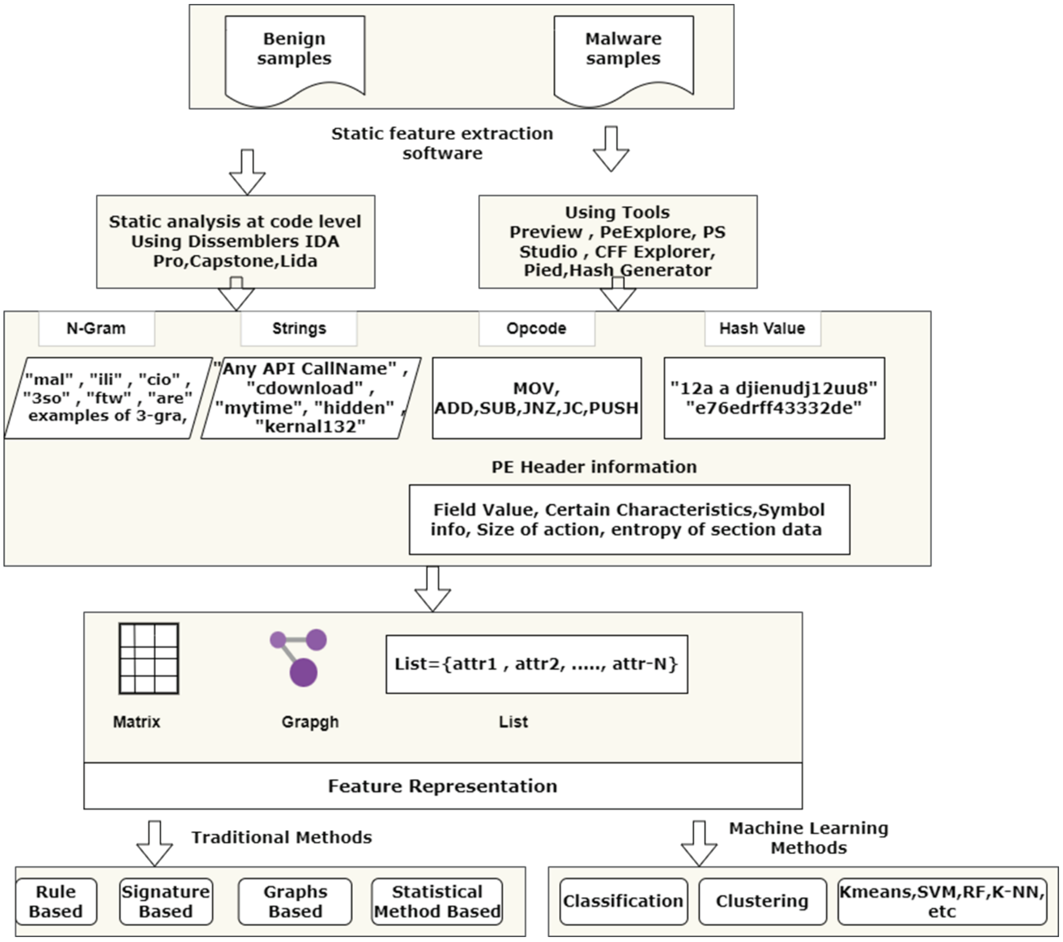

Benign samples
Malware samples
Static feature extraction software
Static analysis at code level Using Dissemblers IDA Pro,Capstone,Lida
Using Tools Preview , PeExplore, PS Studio , CFF Explorer, Pied,Hash Generator
N-Gram
Strings
Opcode
Hash Value
"mal" , "ili" , "cio" , "3so" , "ftw" , "are" examples of 3-gra,
"Any API CallName" , "cdownload" , "mytime", "hidden" , "kernal132"
MOV, ADD,SUB,JNZ,JC,PUSH
"12a a djienudj12uu8" "e76edrff43332de"
PE Header information
Field Value, Certain Characteristics,Symbol info, Size of action, entropy of section data
List={attr1 , attr2, ..., attr-N}
Matrix
Grapgh
List
Feature Representation
Traditional Methods
Machine Learning Methods
Rule Based
Signature Based
Graphs Based
Statistical Method Based
Classification
Clustering
Kmeans,SVM,RF,K-NN, etc

Fig. 3. The malware detection system architecture based on static extracted features

The assessment of numerous potential signature-based techniques is presented in this sector. Karnik et al. presented a malware revealing method where he uses the sequences of the function[53]. The sequence element represented the opcode group, and the function sequence was the hallmark of the malicious program file to distinguish the malware versions. The measure of cosine similitude has been calculated to deal with the obfuscation systems. Nevertheless, in this case, advanced obfuscation methods (equivalent instruction substitution) and packaged malware were unable to defeat the attack. In 2007, Bruschi et al [54] devised a method for categorizing malware using graphical representations [80]. According to the author, this method can control several fundamental obfuscation techniques. Binary files were constructed to match control flow charts to graphs of previously identified hazardous files and then executed. Two algorithms were utilized in this malware detecting technique. The first method searches for similarities between the two graphs of the binary program file B which is underneath assessment and, M. that previously recognized file malware. The B-file charts that had been reduced were then compared to the known

M-file charts in the second approach. Concerning the first and second methods, the author calculated the false positive rate as 4.4 % and 4.5 % correspondingly for 78 malware data samples that were established against these procedures. Nonetheless, this technique is incapable of dealing with zero-day malware. Based on the characteristics of an n-graph byte sequence, Zhang et al. suggested a technology for detecting and classifying malware using n-gram byte sequence properties [55]. A selection strategy removed pre-eminent bytes of n-gram that can indicate the malware program files. Subsequently, a classifier was constructed using the method of a stochastic neural network. For each classification, entailed of a series of malware detection decisions. Three malware courses have been occupied from the virtual VX Heavens database for training. The authors [56] presented an opcode-based machine learning methodology for detecting unknown malware program files. The Opcode have been arranged into the following sizes: 1 byte, 2 bytes, 3 bytes, 4 bytes, 5 bytes, and 6 bytes features.

In the n-byte function sets, four classification techniques were employed, including Decision Tree (DT), Naive Bayes, and Random Forest (NB) and Adaboost. According to the author, this method can considerably predict file maliciousness. Griffin et al. provided a technique for the identification of heuristic malware[68]. This proposed method generates a 48-byte sequence that was used as a string signature for identifying the malware strains. The authors employed the several module signatures for training the classifier instead of utilizing a single component signature. When compared to a single component signature, it has a higher probability of achieving high accuracy. On the other hand, it was unable to comprehend the effect of many component signatures during runtime. The one-sided perceptron method employed by Gavrilut et al. employed a single-sided perception of several machine learning algorithms to distinguish between benign and malicious [69]. This algorithm was intended to minimize the number of false positive rate. Firstly, they employed a basic partial perception cascade, now the unilateral partial perception system cascading demonstrated greater accuracy (88.79 %) compare to simple partial perceived cascades. The authors suggested a technique for malware recognition using the recovery of malicious file execution [70]. Malware flow control charts have been created using execution flow function calls. Graph matching has been performed to match the malware program files with the stowed CFG malware patterns. Malware patterns have been saved. The CFG was labeled harmful by a malware detector when it was included inside the CFG template. The issue is that obfuscation methods such as code rearrangement may escape malware detection by changing the real execution of the path of the malware. The authors created a text-based pattern matching technique for the development for the revealing of a malware system [71]. In this study the feed-forwards bloom filter was used to scan the whole collection of sample malware files.

Two kinds of outputs were generated: (i) malicious file matched and (ii) a subset of signatures. The Signature subsets were mined from the signature data base that was required to recognize the malware program files. A check reduces bloom filter-induced false-positive outcomes. It deals with the two problems. Firstly, the enormous database of signatures is easy to manage by decreasing those using subsets of signatures. The issue is that obfuscation methods such as code rearrangement may escape malware detection by changing the real execution of the path of the malware. The author created a text-based form matching technique for the development of a malware system identifications [71]. In this study the feed-forwards bloom filter was used to scan the whole collection of sample malware files. Two kinds of outputs were generated: (i) malicious file matched and (ii) a subset of signatures. The Signature subsets were mined from the signature data base that was needed to recognize the malware files. After that, a check is carried out to reduce the number of false-positive outcomes caused by the bloom filters. It deals with the two difficulties. Firstly, the enormous database of signatures is easy to manage by decreasing those using subsets of signatures. Secondly, a single-bit vector is used to manage the difficulty of memory scaling. Text-based detection matching method has not produced useful results that this text-based revealing model does not offer the high false positive outcomes. The authors have suggested a method for packaging detection framework to tackle obfuscation malware [45]. The primary goal of this technique is to detect malware that has been packed. Malicious sample have been evaluated utilizing static analysis for collecting the more refined executable attributes. Then, using a two-class vector machine support approach, a malware detector was trained to identify malicious code. Veeramani and Rai have built a system for malware detection utilizing the appropriate API calls[26]. The IDA Disassembler was used in this method to extract the malicious API calls. Unpacking tools were employed before the virus was disassembled to investigate packed malware. The malware program files and Windows system 32 are evaluated statically to excerpt the appropriate API for these both classes. SVM classifier training followed API call extraction. Even though various methods exist to deconstruct packaged malware, every infection is hard to detonate. Obfuscated malware makes API call extraction difficult. Consequently, there is high chances of the false positive results.

The authors in [72] Extended Moskovitch et al. [73] work, which was built employing opcode pattern set of features. Four sizes of 50, 100, 150, and 200 opcode n-gram sequences were created from malware program samples. Eight different classifiers were trained. Logistic regression, Artificial Neural Networks (ANNs), Vector Support, Naive Bays, Random Forestry, DT, and Boosted Naive Bays are some of the methods that are used to train the opcode patterns. In the study, it was discovered that the accuracy was above 96 %, and the false-positive rate was just 0.1 %. The API graph dependent malware revealing solution was proposed by Elhadi et al. [74]. In this method for each malware file to a data-based API calls graph were design. Apparently, a malware database consisting of data type API graphs was created. The analysis used the Longest Common Subsequence (LCS) technique to equal the unknown file's resemblance to the saved malware graph. However, in the trial, just 85 malware samples were utilized, which led to a detection rate of 98 %. In particular, The authors suggested the malware system for the portable executable program file of windows type PE header information [43]. This approach was utilized to train the model based on file metadata. The results of the experiments suggest that the executable metadata may be used to discriminate between benign and malicious. On the created attribute of the portable program executable header three important machine learning algorithms were used. Decision Tree classification beat NB and logistic regression classification. The authors joined the n-gram structure with the statistical examination with the malware-detection ML method [75]. The goal was to produce the co-operative architecture to identify the emerging malware i.e., metamorphic, those are difficult to identify by employing the statistical method of the n-gram model. The Markov blanket approach has been employed to choose the features. It has been employed because it minimizes feature size. In the next step, the Hidden Markov Model (HMM), which achieved an accuracy of 90 %, was trained in the resultant function sequences.

Srndic et al. submitted a study on ML algorithms for static analysis for the classification of malware samples [76]. The virus developer now uses portable document format (PDF) and shock wave flash (SWF) files to incorporate the executable scripts to harm system resources. This research has evaluated 40,000 SWF and 440,000 PDF files. To identify possibly dangerous programs included inside PDF and SWF files, this approach was employed in conjunction with other methods. When it comes to malware detection, there are many factors to consider. Kim et al. present a technique for PE malware header detection using the ML algorithm [77]. The main goal of this approach is to advance the revealing ratio as equated to preceding methods of malware revealing that focused on the PE header data of executable files. The portable executable header features of malicious and benign program files were observed in both cases (PE header information). From there on three ML procedures were applied to the retrieved feature: Gradient Descent (GD), SVM and Classification and Regression Tree (CART). To put the technique through its paces, almost 27,000 malwares and 11,000 benign files were used. This system has an accuracy rate of 98 % and a false positive rate of 0.2 %. This method, on the other hand, will not function if the actual PE header was obfuscated. Narra et al. developed a malware detection clustering approach that is competitive in terms of efficiency with SVM [3]. In their prior research, they have only applied the clustering technique like k-means, the maximizing of expectations with HMM. In the research, clustering techniques have been trained with 7800 malware program samples without HMM and effects were equated with the SVM classification, those have also been trained in the same data samples. It's problematic to regulate that many clusters are in the malware dataset, leading to a hit and a testing solution. Raff et al. developed byte n-gram-type approaches and examined the shortcomings of earlier n-gram strategies, those relied on the n-gram function [42]. The characteristics for Elastic Net Regularized logistic regression model training have been selected and the multi-byte identification has been analyzed. This approach led to the discovery of three significant faults in the prior n-gram methods. First challenge is how the earlier corpora was created the overestimate the detection precision, the second challenge was most n-grams have only been retrieved from string features and the third challenge was, n-gram feature has overpowered the classifier.

Searles et al. enhanced the previous work the usual malware detection graph program using control flow technique [93]. In this approach, similarities were found between CFGs retrieved from binary files using the Shortest Path Graph kernel (SPGK). In conjunction with a similarity matrix, the SVM method was then employed to enhance the precise classification. Various parallelization methods were assessed to lessen computational costs or to boost up the classification. It intimates a better accurateness on 22000 binary files in contrast to the 2 gram and 3-gram model. But it is quite difficult to cope with the enormous sizes of CFG. No such solutions have been familiarized to deal with this method. The authors utilized malware detection characteristics such as CPU utilization, network traffic, and swap usage [2]. This way, spyware known as APTs (Advanced Persistent Threats) detected. The findings further showed that the obfuscated malware can also be detected. The classifiers are trained on the self-organization feature map to minimize the problem of over fitting and produced great outcomes 7% to 25% higher than older methodologies. The authors have presented a technique for malware detection that analyses execution files using static analysis tools and extracts elements such as DLL import, hex dumping, and assembly code [34]. These structures have been applied to paragraph vectors via the training of SVM and k-NN algorithms, which have been used to analyze the data. Approximately 3600 malware samples were used in the trial, which resulted in a 99 % accuracy in detection. But simple obfuscated method can avoid the proposed method. The graph-based malware revealing method was enlarged by authors [94]. In the current methods each node represents a single system call freely however in the methodology every single node in the graph signifies a group of systems calls for related kinds for the compilation of the system dependence graph (ScD-graph). It is also possible to detect mutated malware using this technique, which involves categorizing a weighted directed network, also known as a group relationship graph (such as oligomorphic and metamorphic). The main issue with graphics detection techniques is the difficulty in matching graphs for similarity. Same-similarity and NP-similarity metrics have been proposed as potential solutions to this problem.

There have been 2631 malware samples utilized in the development of the anticipated model that comprises 48 malware families. The detection rate for the proposed model was 83.42 % when using the data from this dataset. The suggested system was assessed on 10568 binary program files, with an accuracy rate of 98.7 %, and it was shown to be effective. The authors [96] created SVM type classifiers, which are now widely used. Making use of the many features provided by the executable PE header. The static PE header features were split into 54 categories. A total of 500,000 malware data samples collected from the Vxheaven and Virusshare repositories were utilized to train the SVM classification system. There was a significant flaw in this method in that the proposed model was not explored in depth before it was implemented. In addition, there is no clear description of how to extract the static PE features of hidden malware from its source code.

Table 7 confirmations the complete analysis of the methods evaluated based on the signature. It also compares the methods suggested after a study of the signature-based malware classifiers. Further, this also summarizes the malware features, classification methods, and performance metrics utilized to develop the suggested strategies. In Real-time Detection Scenarios, signature-based methods are advantageous for reduced overhead and runtime. Some suggested methods have claimed greater than 99 % accuracy in malware detection. These methods used several malware features and showed their efficiency in the classification of dangerous and benign files. When combined with malware detection features, the representation methodologies show an important part, as demonstrated by Burn AP et al. (2017) [2] and the authors [78], both of whom are using API calling to improve detection rates but have used various machine learning and representation methodologies to do so. However, methods relying on signatures have some drawbacks, such as the inability to identify new and disguised malware. While we think that these methods may identify unknown malware, especially those which rely on heuristic signatures, they cannot detect malware that has been disguised. This is because the malware detector, which is based only on the static features of known malware, is rendered ineffective. Compared to traditional malware detection approaches, the presence of machine learning in signing-based methods has resulted in significant improvements in malware prediction (Fig. 2.). Using machine learning techniques, the authors in [42], [79], and [73] have obtained more accuracy than the research by authors in [80] and [75].

4.2. Behavior-Based Malware Detection

In the behavioral approach, malware is detected based on malicious activities carried out during execution. APIs, browser events, and system events, as well as network events, are all examples of feature kinds of behavior that have been specified [37, 81, 82]. These qualities are classified into three groups in the context of behavior methods: File activities, registry activities, and network activities. The entire process description to construct behavioral malware detection systems is illustrated in Fig. 4. New malicious Files can be identified by employing this malware detection because malware files somehow share harmful behaviors. Consequently, the malware detection system is trained to recognize similar behavior to identify new malware or variants of current malware that has been introduced. Similar activity, thus, is utilized to educate the system for malware detection to detect new malware or recognized malware variation. The behavioral method also provides a strategy for controlling the obfuscated malware. The obfuscation tactics utilized by makers of malware can dodge signature-based techniques. Behavior-based technology is trained in two ways. Anomaly denotes a malfunction that malicious files perform. When a file displays an aberrant behavior other than usual file stored behavior, the file is declared as a malicious file. The logic behind the malware is the abnormality (abnormal operation performed by malware). Malware detectors are being trained in the system to identify abnormalities. Benign files are examined using static or dynamic analysis. For the categorization of benign files, the usual activities of the benign files are used as a guide. Anomaly and benign analysis are the process in which malware data are analyzed alongside innocuous ones. Because both normal and dangerous activities are performed, it is desirable to distinguish between benign and malware behavior rather than using an anomaly-based approach. The detector must be trained for a longer time as compared to when using an abnormal technique. Behavioral malware detection techniques that use heuristics perform much better. Machine learning is much more important than traditional malware detection methods when it comes to detecting highly complex malware.

Table 7: A complete analysis of the signature- based malware detection techniques

Author(s)	Input	Data Source/ Number of malwares samples	Outcomes
Wang and Wu [45]	Portable Executable (PE) Header	1056-Packed Malware and3789-Unpacked Malware and Benign Files /Vxheaven and PCHome Malware Repositories	TPR-94.54%
Veeramani [26]	Application Programming Interface calls	214- Malware and 300-Benign Files Vxheaven Malware Repository	Accuracy-97%
Gavrilut [69]	Application Programming Interface calls	16437-Benign Files, 12817-Malware	Accuracy-88.78%
Shabtai [73]	N-gram Opcode sequence	7688-Malware and 22735-Benign Files / Vxheaven Malware Repository	Accuracy-96% TPR-95%, FPR-0.1%
Elhadi [74]	Data Dependent Application Programming Interface Graph	85-Malware Files / Vxheaven Malware Repository	Accuracy-98%
Markel [43]	Portable Executable Header	42003 benign files, 122799 malware files.	Accuracy-97% (Tree CART) 94.5% (LR)
Pechaz [75]	N-gram	1207-Malware and 194-Benign Files / Vxheaven Malware Repository	Accuracy-90%
Srndic [76]	Strings, API Calls	440000-PDF and 40000-SWF files / Virus total Malware Repository	Accuracy-99%(PDF)95%(SWF)
Wang [80]	Opcode Sequence	11665-Malware, 1000-Benign Files / Vxheaven Malware Repository	Accuracy-88.75%
Huda [78]	Application Programming Interface calls	Vxheaven Malware Repository and using Honeypot	Accuracy-96.84%
Kim [93]	Portable Executable Header	271095-Malware, 9773-Benign Files / Vxheaven Malware Repository	Accuracy-99%
Narra [3]	Opcode Sequence	7800-Malware Files / VirusShare dataset	Accuracy-98%
Raff [42]	Byte N-gram	400000-Malware and Benign Files / /VirusShare Malware Repository and Open Malware, MS Window	Accuracy-97.4%
Liu [19]	N-gram opcode, Image representation	20000-Malware Files	Accuracy-95.10%
Searles [117]	Control Flow Graph	22000-Malware Files	19% more accuracy than n-gram model
Nagano and Uda [79]	DLL import, hexdump and assembly code	3600-Malware Files / MWS 2016 Malware Dataset	Accuracy-99%
Nikolopoulos and Polenakis [96]	System Call Dependency graph	2631-Malware Files	Accuracy-83.42%
Le [97]	Greyscale images	10568-Malware Files	Accuracy-98.8%

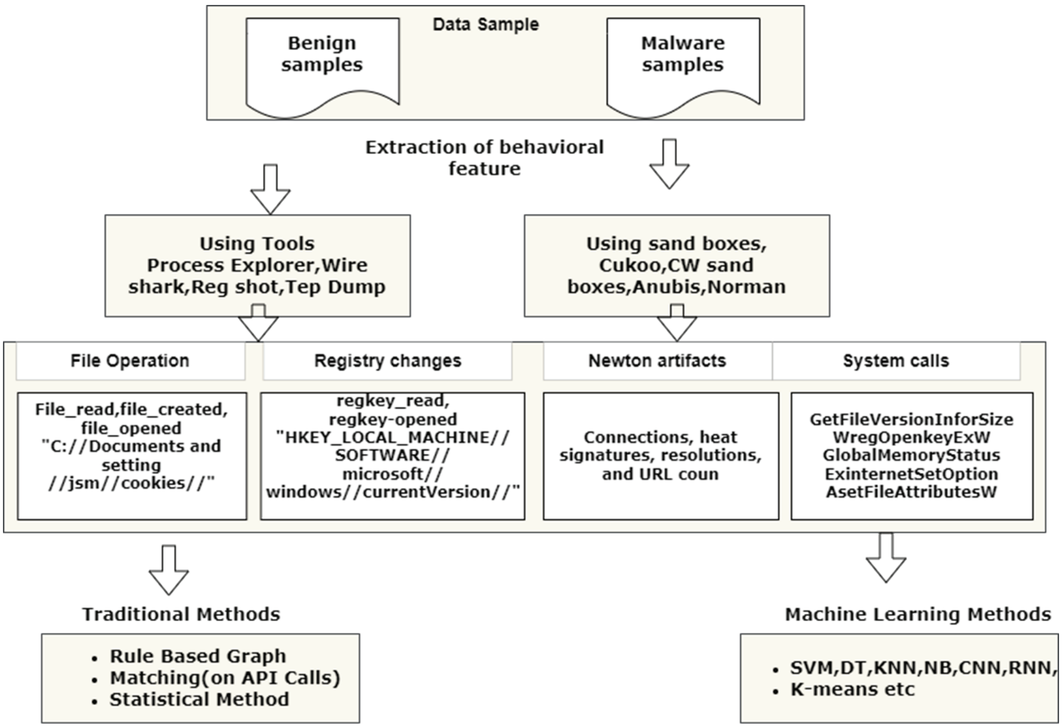

Data Sample
Benign samples
Malware samples
Extraction of behavioral feature
Using Tools Process Explorer,Wire shark,Reg shot,Tep Dump
Using sand boxes, Cukoo,CW sand boxes,Anubis,Norman
File Operation
Registry changes
Newton artifacts
System calls
File_read,file_created, file_opened "C://Documents and setting //jsm//cookies//"
regkey_read, regkey-opened "HKEY_LOCAL_MACHINE// SOFTWARE// microsoft// windows//currentVersion//"
Connections, heat signatures, resolutions, and URL coun
GetFileVersionInforSize WregOpenkeyExW GlobalMemoryStatus ExinternetSetOption AsetFileAttributesW
Traditional Methods
Rule Based Graph
Matching(on API Calls)
Statistical Method
Machine Learning Methods
SVM,DT,KNN,NB,CNN,RNN,
K-means etc

Fig. 4. Scheme of behavioral malware detection system architecture

The proposed behavior-based malware detection different algorithms are explained in this section. Bailey et al. provided an interaction malware detection technique with system services that included System API calls, special addresses, and functions [83]. Malware files behave in a variety of ways, which may be classified into many categories. A malware file, for example, may include a virus, a Trojan horse, spyware, or worms. Moreover, malware is also categorized according to the information retrieved in various classes and subclasses. The Malware Instructions Set (MIST) was suggested by Trinius et al. have described a new Malware Instruction Set (MIST) representing malware behaviors. CW sandbox used to analyze the malware samples[84] . The XML report is created by Sandbox and after that it is converted into a MIST format. In addition, machine learning and data mining have helped to improve the efficiency of malware analysis while simultaneously reducing the number of reports. Rieck et al. proposed an autonomous malware detection framework to detect malware class variants [84]. The framework divided into two steps. The first step uses the clustering method to make malware types comparable. The second step classifies or assigns the unknown malware file to the discovered classes. MIST representation was utilized to speed up the process of clustering and classification. Vasilescu et al. have made use of the Cuckoo sandbox to get dynamic analysis [85]. The generated report includes API calls to the system, log entries, and portable binary running information, among other things. The malware detector is trained with these extracted characteristics to identify zero-day malware. A data-dependent API graph-based malware detection method was developed by Elhadi et al. [76]. In the first stage, the clustering technique is used to create malware classes that have comparable characteristics to one another. In the second step, the unknown malware file is categorized or allocated to one of the recognized malware classes, depending on the properties of accelerate the clustering and classification process. Hegedus et al. utilized behavioral characteristics for malware identification to two classifiers k-nearest neighbor and random forest [86]. The technique proposed operates in two phases. In the first phase, the random projection is used to reduce the dimensionality and duration of the variable space. The index of Jaccard was used to measure the similarity of malware traits. Then the second phase is to utilize, detect and categories malware samples with Virus Total's K-nearest neighbor classifier. According to Mohaisen et al. they have used the malicious filename in a virtual environment to remove behavioral devices [81]. After that, malware samples are identified based on file transactions, network activity, registry key changes, and memory operations to determine their origin. SVM was used to categorize the data, and it was quite effective. Logic regression and hierarchical clustering methods were used to split Characteristics. A special version of this format, known as the MIST representation, was created to Malware into families that had similar characteristics, with each family having its own set of attributes. The experiment was conducted out on both medium and big datasets (each with 400 samples), and the results were compared (115,000 malware samples). The article implies 98% accuracy in the classification of malware. According to Ghiasi et al. proposed a new malware recognition approach using CPU registry entries [87]. In the monitored environment, this approach extracts API calls with dynamic analysis by running binary files. A similarity between two binary files was then calculated based on the content of their register. Four Machine learning methods are utilized include Random Forest, Bayesian Logistic Regression, and Bayesian Logistic Regression with Bayesian Logistic Regression. To train the detection model, regression, SMO, and J48 have all been used. The article declares well-regulated time to match existing patterns efficiently. In the Cuckoo sandbox method, Pirscoveanu et al collected the contextual features of binary files. A random forest algorithm was developed for the categorization system [65]. Ki et al. presented a technique of malware identification that relied on the API call sequences [108]. The study shows that dynamic analysis has been proven to be more effective in obtaining malware behavioral features. it is observed that all the malwares contain Sequence alignment technique has been used to address redundant or non-relevant code insertion in malware. In the testing step, if unknown file calls are matched to stored API patterns for the extracted APIs, the file is declared malicious else the file is benign. Pan et al. proposed a malware classification system based on the BPNN model [88].

The HABO system is designed to collect runtime functions. The essential elements of the report were extracted, for example reading foreign memory, creating mortexes, creating the process, or modifying registry entries. The model was developed utilizing the method of the Back Propagation neural network. Narayanan et al. developed a supervised machine learning techniques to build the classification of malware [92]. Polymorphic malware was managed using the suggested method as pictures capable of capturing small changes while maintaining the general structure. Three classifiers have been trained on the provided dataset using the KNN, ANN, and SVM methods. Cho et al. presented a technique employing the API call sequence [93]. API calls have been extracted by running malware samples in cuckoo sandbox for the construction of the API sequences. 150 samples from 10 malware families were trained and 87 % accuracy was obtained. Mira et al. presented a research in which they built an API-based malware detection model that was trained in two algorithms: the Longest Common Subsequent (LCSS) and the Longest Common Substring (LCSS) (LCS) [90]. The multi-process execution behavior of malicious files was presented by Bidoki et al. [57] . Malware may disseminate its activities over a wide range of legitimate operations, but the overall effect is always negative. It is necessary to use the improved learning method during the training phase, and every API request must be gathered during the detection phase. The execution rules of all processes have been merged to determine whether a binary file is harmful. A graphical call system technique was proposed by Ming et al. (2017) [94] that generates an invisible malware problem system call-based dependency diagram using a graphical call system. According to the author, all variants of the same malware have the same semantics; the only difference is in the malware's syntax. A technique was trained on 5200 malware files and then tested on 960 malware files before being released. The model that was proposed was 97.30 % accurate. According to Wagner et al. (2017) [95], malware may be visually identified by its activities. This system was created to monitor the visual pattern of malware activities using knowledge-assisted visual analysis. Mao et al. (2017) [113] established a methodology wherein they assess, based on their usefulness, the importance of system subjects. This allows us to create a network for security dependencies that gives us an insight into the value of system object security. Enhanced DT It was decided to utilize Amazon Web Services to host the new cloud-based design, which provides for greater scalability. To train the classifier, 150,000 malware samples and 87,000 benign samples were used. Using the improved DT algorithm, the detecting system obtained a 99 % accuracy rate in its detection.

Ding et al. proposed the idea of graph-based malware detection [31]. Instead, then creating a behaviour graph for each malware, the author proposes a standard graph for every malware family. Dynamic taint analysis technique was used for building the behavior graph. The highest weight parameter subgraph was used to compare the graph of an unknown file to the graphs of each malware family that had previously been created. Stiborek et al. have presented a technique for capturing malware behavior by running malware in a sandbox environment [99]. The sandbox has a series of names and resources for each malware sample. The term for this approach to the issue framing is multiple instance learning. Machine learning is used to samples of varied sizes of malware; a report on many instances of learning explains several ways to deal with the issue of sample size. To address the issue of sample size fluctuation, a vocabulary approach was employed. The method suggested was developed in a big 11,2115 binary and obtained a precision rate of 95.4 %. Ghafir et al. presented an advanced persistent threat detection approach. [100]. To train the classifier, the author utilized SVM, KNN, and group methods. This method successfully generated the accuracy of 84.8% in APT prediction.

Run-time features were used by Alaeiyan et al. to construct a malware detection algorithm [101]. Using the Parsa sandbox, this method has also discovered evasive malware. The given method was tested on 1100 malware samples and was shown to be accurate to 97.9% of the samples. For the Windows platform, Xiaofeng et al. created an API that uses sequence-based malware classifiers.[102]. The cuckoo sandbox was used to eliminate dynamic API calls from the application. Following the training of two classifiers. A model for categorizing malicious traffic was proposed by Arivudainambi et al. in their study of network traffic analysis [103]. They integrated PCA to handle sophisticated anti-network traffic analyses. The approach presented was evaluated via the Noriben, Cuckoo, and Limon sandboxes by running 1000 malware samples. This technique has a 99% accuracy rate in terms of malware identification. Yucel et al. presented a techniques for creating executable memory file images [104]. A total of 123 malware samples from various families were collected. It was discovered that malware samples ran similarly when run in virtual machines, and 3D memory snapshots were created for comparison. It showed that various families of malware had varying rates of similarity, such as 0.99 for Marina Botnet, 0.99 for Rex Virus, and 0.886 on average. Rabbani et al. presented a model for the detection of malicious behavior conduct in network traffic using a Probabilistic Neural Network (PNN) [105]. The vector featured IP, TCP, UDP, CON, jitters, and other network capabilities. A modified version of this technique was developed by combining the PSO (Particle Swarm Optimization) algorithm with the PNN algorithms, which resulted in a malicious traffic detection rate of 96.5 %.

Table 8 compares all the behavioral techniques that were examined. There is now more research being conducted on behaviorally based techniques of malware identification. This is because signature-based techniques are incapable of dealing with emerging and zero-day malware. Using runtime features, these approaches have outperformed signature-based accuracy solutions in terms of accuracy. Many researchers, including Elhadi et al. [106], Pan et al. [88], Ali et al. [67], and others, have made use of dynamic API calls. Alaeiyan et al. [101] have used file, registry, and network activities in the training of malware classifiers using supervised classification techniques, as have Stiborek et al. [99], Paketas et al. and Stiborek et al. [99]. For instance, Ghafir et al. [101] have been developing their models with several runtime characteristics. The detection rate is considerably greater for behavior methods as compared to signature based. The methods also suggested to claim that new and obscured malware may be predicted. Historically, it has taken a long time to extract the runtime function from conventional behavioral methods; however, the use of machine-learning algorithms has sped up the process, allowing the proposed model to make use of more data and larger malware samples to train and test the malware classification system. However, the implementation of these suggested methods presents certain problems and difficulties. The proposed methods are tested and verified using a range of malware samples. Furthermore, classifiers vary in techniques for training. Before prediction, high processing time and the running duration of malware samples are obstacles to applying the behavioral technology to a real system. The advantages of signature and Behavior methods, described in the next section. In addition, hybrid approaches have been put out and will be examined in the section.

Table 8: A Complete analysis of the behaviors-based malware detection techniques

Author(s)	**Input**	**Data Sources/ Number of malwares samples**	**Outcomes**
Elhadi [106]	Data Dependent Application Programming Interface Graph	416-Malware and 98-Benign Files / Vxheaven Malware Repository	Precision 98 %
Mohaisen [81]	Application Programming Interface calls and Network Activities	115000-Malware Files / Antivirus companies	Precision 99.5 % Recall 99.6 %
Ghiasi [87]	Contents of Registers, Application Programming Interface Calls	850-Malware and 300-Benign Files	Precision 95.9 %
Pirscoveanu [65]	Window Application Programming Interface calls	42000-Malware Files / VirusShare and Virustotal Malware Repositories	Precision 98 %
Ki [118]	Application Programming Interface Call Sequence	23080-Malware Files / VirusTotal Malware Repository and Malica Project	Recall 98.8 % F1-Score 99.9 %
Pan [88]	Application Programming Interface calls	13600-Malware Files / Kafan Forum	Precision 98 %
Nayaranan [92]	Representation of Malware in form of images	10868-Malware Files / Kaggle Microsoft Malware Dataset	Accuracy 96.6 % (Linear KNN)
Cho [93]	Application Programming Interface call sequence	150-Malware Files / Vxheaven Malware Repository	Precision 87 %
Mira [90]	Application Programming Interface call sequence	13600-Malware / VirusSign, SAMI, CSDMC Malware datasets /	Precision 99 %
Mao [119]	System objects	7257-Malware Files / Vxheaven, MALICA and Virustotal Malware Repository	Precision 93.92 % FPR 0.1 %
Bidoki [57]	Application Programming Interface Calls	378-Malware and 500-Benign Files / Vxheaven Malware Repository	Precision 91.66 %
Ming [94]	Dependency Graph	5200-Malware Files	Precision 97.30 %
Wagner [95]	Sequence of Application Programming Interface calls	8847-Malware and 1460-Benign Files / Vxheaven Malware Repository	Precision 95.25 %
Pektas and Acarman [98]	Registry, Network, File System, Application Programming Interface call sequence	17900-Malware / Virusshare Malware Repository	Precision 92.5 %
Ali [115]	Run-time features	150000-Malware and 87000-Benign Files / Malware Repository of Nettitude	Precision 99 %

Stiborek [99]	Behaviour artefacts	112115-Malware Files	Precision 95.4 %
Alaeiyan [101]	Behavioral features using Parsa sandbox	1700-Malware and 1700-Benign Files / Virusshare Malware Repository	Precision 97.9 %
Xiaofeng [102]	Application Programming Interface call Sequence	1430 Malware and 1352 benign Files / Virusshare and Virus Total Repositories	Precision 96.7 %
Arivudainambi [103]	Network Artefacts	1000-Malware Files	Precision 99 %
Rabbani [105]	Network features	677789-Benign Files and 22211-Malware	Precision 96.5 %
Namavar [4]	Behavioral features	18831-Malware Files / Vxheaven and Microsoft Kaggle	Precision 99.65 %
		Datasets	
Yucel [104]	Memory Images	123-Malware Files / Virusign Malware Dataset	Precision 99.5 %

4.3. Hybrid Malware Detection

Behavior-based and signature-based techniques have some advantages and disadvantages. Consider that several researchers have suggested ways to hybrid malware detection that incorporate the advantages of both static and dynamic malware techniques. In this part, we discuss and compare the research of different methods for hybrid malware detection using various criteria. Rabek et al. suggested a malware detection technique for detecting harmful files that were obfuscated [107]. In addition to the dynamics, the information gathered includes all the function names, addresses, and return addresses of system calls. Malware files have been executed to record runtime activity in a controlled dynamic environment. The program is classed as malicious when calling the same previously saved system calls (which represents the malware file). If, however, any unnecessary system calls are included in the code, the malware creator fails. The graph detector for network worms was suggested by Collins et al. [108]. Hosts in a network were represented as nodes and connections as edges. The approach mimicked a network to learn about the behavior of worms. It addresses worms exclusively, not Trojan horses, viruses, or other malware. The following are the different methods used for detecting hybrid malware. To minimize false-positive reactions, Mangialardo and others [109] suggested that the FAMA framework address faults in both static and dynamic analysis methods. For extracting static characteristics, IDA pro was utilized and Cuckoo's sandbox for capturing behavioral functions. The features gathered were then used to train the random forest and C5.0 algorithms, which were both developed by the researchers. The tests showed that the unknown file can be classified as benign or dangerous with 95.75 % accuracy. Shijo et al. presented an integrated malware detection approach that was based on machine learning [110]. Following the deconstruction of the binary files, information in the form of readable string information was obtained. It also takes care of any superfluous printed strings that were used to conceal the code. Once the binary files had been run, they were subjected to dynamic analysis in the Cuckoo Sandbox. This included file system-relevant API calls, change registries, and the extraction of a specific process and memory addresses. The Random Forest and Support Vector Machine were used to train the malware classifier. Edem et al. suggested a malware detection methodology that is automated [111].

Several data-mining techniques have been used in integrated with the author's malware investigation. A clustering technique called k-means clustering was used to group malware samples that had similar behavior together to make the analysis easier and more useful. The malware samples were analyzed both statically and dynamically from the outset. The IDA Pro and OllyDbg tools, as well as the CW Sandbox, were utilized to extract the static characteristics from the code during the static analysis. An XML report on the malware sample activity is generated by the CWSandbox. The data mining method was then utilized to process both static and behavior characteristics. The enhanced malware detection method for malware categorization using the SVM algorithm was proposed by Okane et al. [112]. Unlike others, this method utilizes the runtime trace as the tracking feature of the application to train the detection system. The Support Vector Classifier was trained after the extracted functions were reduced to smaller feature sets utilizing opcode filtering methods, which resulted in the reduction of the retrieved functions to smaller feature sets. Nauman et al. introduced the concept of tridimensional decision-making in the context of malware detection systems [113]. All the previously proposed techniques were accompanied by a binary file, which might be malware or benign. Practically, the detector is unable to correctly classify all the examined files. Some complicated malware is wrongly categorized, such as Stuxnet or extremely unique files. In this instance, detectors are more likely to produce false positive or negative findings. Thus, the strategies of fered address such viruses with three types of accepted, rejected, and delayed decisions. Two approaches to malware are proposed: (i) Rough theoretical rough game sets (ii) rough theoretical information test. The disadvantage of this method is that it does not offer a solution for the handling of malware that has not been detected yet. By combining various static and dynamic analytic methods with component-based frameworks, Kaur et al. have created a hybrid methodology for identifying malware that may be used to detect a wide range of threats [20]. Initially, the Hybrid Framework was developed to automatically identify zero-day malware that mimicked the destructive behaviors of existing malware. The malware detector is trained by extracting static and dynamic characteristics from malware samples, which are then used to identify malicious code. This technique captures a broad variety of static properties, such as the hash value, PE header information, string values, and dynamic actions, such as process activities, file operations, storage operations, and network activities. This method makes it possible to identify new malware and classify it according to its static characteristics.

Kolosnjaji et al. proposed an improved semi-supervised malware detection approach that incorporates both dynamic and static malware analysis results to improve the performance of categorization and classification

[41]. During the processing of extract characteristics of dynamic and static analysis, various methods were employed, which are distinct from prior malware detection approaches. To categorize static findings, the semi-supervised propagation method has been utilized and the dynamic reports, which found hidden semanticized characteristics in malware files, have been statistically modeled. Above all, it provides an online dynamic malware classification system for non-parametric techniques. Damodaran et al. proposed a hybrid approach for training a classifier, in which an Opcode sequence or an API request was utilized to train the classifier in parallel [114]. Like earlier hybrid techniques, binary data is used to extract both static and dynamic features from it. The Hidden Markov technique was then used in the classification process, both dynamically and statically. This approach only produced good results for a few malware types. Current methods of obscuring may also avoid the static analytical procedure of malware identification using malware tactics. Pfeffer et al. suggested MAAGI for malware detection (Malware Analysis and Attributed using Genetic Information) [115]. In this context, the genetic algorithm has been used to the comparability features of malware. Malicious samples are processed in static and dynamic sandboxes to collect malware features. Static analysis was carried out using PEid and IDA pro tools while dynamic analysis tools were utilized on Symon and Introvert. The foundation of the MAAGI is founded on the notion that biological behavior and malware behavior share a great deal. The outcome was the creation of the malware detection framework using the artificial intelligence algorithm, which showed promising results. It is expected that it will improve collaboration between cyber defense and artificial intelligence groups in the long run. Huda et al. suggested a semi-supervised machine learning method that would automate information regarding unknown malware in the sensing system using previously tagged and unlisted data, which would be implemented in a sensor network [89]. Other techniques do not have the advantage of automatically updating the database of the detection system, which makes this one stands out since it does not need external help. This technology makes use of k-means clustering with reverse document frequency as the distance metric, word frequency as the distance metric, and the Support vector machine technique to categorize binaries to extract cluster information, all of which are used to extract cluster information. Huda et al. [116] proposed a hybrid method that combines the inclusion of wrapper filters with the selection of characteristics to get the best possible results. The author has selected the maximum and minimum characteristics in this research After that, several MR+SVM, MRED+SVM and Fisher+SVM machine learning methods were employed to train the model that provided 99,49 % precision utilizing. Table 9 offers a comprehensive summary of the updated techniques for hybrid malware detection, as well as their advantages and disadvantages. It also presents the results of the investigation into hybrid malware ratings. Huda et al [116] achieved the highest accuracy in malware detection with a 99.49 % detection rate. These methods were proposed as a means of bridging the gap between static and dynamic malware detection by integrating the benefits of both approaches.

Table 9: A complete analysis of the hybrid malware detection approaches

Author(s)	Input	Data sources / No. of samples	Results
Mangialardo [109]	Application programing interface call sequences	131073-Malware Files / Virus Share Malware Dataset	Precision-95.75%
Shijo [110]	Printable strings information (PSI) and Application programing interface calls	997-Malware and 490-Benign Files	Precision -98.7% (SVM), 97.68% (RF)
Edem [111]	Signature and Window Application programing interface calls	1143-Malware Files / 1143-Malware / Malware Sample taken from MWanalysis.org	
Okane [120]	System Application programing interface calls	350-Malware, 300-Benign / Vx heaven Malware repository	Precision -86.3%
Nauman [113]	System Calls	UNM Application Dataset	Precision -92.51%
Kolosnjaji [41]	Portable Executable Header Information and	2000-Labelled Malware and 15000-Unlabelled Malware Files / Virus total Dataset	Precision-90%
	Application programing interface Calls		
Damodaran [121]	Portable Executable information, Application programing interface Calls	745-Malware and 40-Benign Files / Vx heaven Dataset	Precision -98%
Pfeffer [115]	Application programing interface calls	8336-Malware and 128-Benign Files / MIT Lincoln Lab Malware Dataset	Precision -86%
Huda [89]	Printable Strings, Imports, Procedure Call Graph (PCG)	967-Malware Files / CA Technologies VET Zoo Malware Dataset	Precision -93.83%, FPR-0.144%
Huda [116]	Runtime activities	2000-Malware and 1500-Benign Files / Malware sample from CA Technologies VET, ZOO, Offensivecomputing.net and Vx heaven Repositories	Precision 99.49%

5. Discussion and analysis of proposed malware detection techniques

Authors	Type		ML Algorithm								Features						
	SA	DA	DT	SVM	KNN	NB	LR	ANN	RF	ADA	MN	PEH	AC	PSI	RC	IM	RTF
Ali [67]		✓	✓	✓						✓			✓				✓
Gavrilut [69]	✓							✓					✓				
Ghafir [100]		✓		✓	✓				✓	✓							✓
Ghiasi [87]		✓											✓		✓		
Huda [89]	✓	✓	✓	✓		✓			✓				✓	✓			
Huda [116]	✓	✓		✓													✓
Huda [78]	✓			✓									✓				
Ki [118]		✓											✓				
Kim [77]	✓		✓	✓								✓					
Kolosnjaji [41]	✓	✓							✓	✓		✓	✓				
Le [97]	✓							✓								✓	
Liu [19]	✓								✓		✓		✓			✓	
Mangialardo [109]	✓	✓	✓						✓				✓				

Mao [119]		✓															✓
Markel [43]	✓		✓			✓	✓					✓					
Mohaisen [81]		✓	✓	✓	✓		✓						✓				✓
Nagano and Uda [79]	✓			✓	✓						✓		✓				
Narra [3]	✓			✓							✓						
Nauman [113]	✓	✓											✓				
Nayaranan [92]		✓		✓	✓			✓								✓	
Okane [120]	✓	✓		✓									✓				
Pan [88]		✓						✓					✓				
Pfeffer [115]	✓	✓											✓				
Pirscoveanu [65]		✓											✓				
Raff [42]	✓						✓				✓						
Searles [117]	✓			✓													
Shabtai [72]	✓		✓	✓	✓	✓	✓	✓	✓		✓						
Shijo [110]	✓	✓		✓					✓				✓	✓			
Srndic [76]	✓			✓									✓	✓			
Stiborek [99]		✓		✓					✓								✓
Veeramani [26]	✓			✓									✓				
Wagner [95]		✓			✓								✓				
Wang [80]	✓										✓						

Table 10: Analyzing three kinds of malware revealing methods: ML algorithms and malware characteristics

The statistical examination of ML revealing techniques is covered in this part of the paper. Because there are three different types of malware detection techniques. These methods may be classified into a variety of categories, including analytical approaches, conventional or machine-based learning systems, deep learning systems, computer technologies, and mobile malware detection technique. We have examined the strategies of malware detection designed for Computer malware. Table 10 summarizes research into methods of malware detection based upon algorithms and the features of input. It shows which algorithms have been greatly utilized and employed in past and present and which characteristics are static and dynamic.

- Static Analysis- (SA)
- Dynamic Analysis- (DA)
- API Calls, Imports, DLL Import- (AC)
- Image Representation of binary file- (IM)
- Register Content- (RC)
- Portable Executable Header- (PEH)
- Runtime features (FILE, Network)- (RTF)

In addition to the analytical category and the machine learning systems, malware types are the important thing in malware detection. For example, hash values, specific string data, opcodes, n-bytes, and registry changes, practice activities, system operations, and other system data. As exposed in the Table 10, most of authors used single-feature sets while many others used multiple different features for the classification of malware. Moreover, it affects malware detection performance how the characteristics are handled and displayed. Many techniques, such as API calls, runtime functions, opcodes, and n-grams, among others, have been tried; nevertheless, the processing and application of different methods have formed superior outcomes overall. Fig. 5 expressions the accuracy contrast to various static malware revealing techniques. The greatest accuracy of 99 % was obtained by [117] the authors in [77], and [27] using the SVM classification methods. Fig. 6 exhibits dynamic malware classifier's accuracy. The authors in [4], [104],and [17] some researchers have reached more than 99 % accuracy. The authors [116] generated the highest precision of 99. 49%, as shown in Fig. 7, utilizing MR+SVM and MRED+SVM systems. From the previous work we can elaborate that static and dynamic technology can produce a more precise malware revealing system with machine learning. Likewise, for constructing a model employing different malware traits a single classification technique is not suitable. However, SVM did better in static analysis than other systems. The algorithms of the dynamic examination collective also worked effectively. Some significant research problems arise after the analysis of different malware detection methods. There are several advantages and disadvantages to each method. For instance, signature-based malware detection methods may identify only known malware that has previously recorded its signature in the detector database. Malware detection systems based on a signature can be readily avoided by obscure methods. Malware detection technology that is based on behaviour may be used to remedy flaws of signature-based detection technique. However, behavioral methods require far more time and have a larger false positive rate than signature-based techniques. The identification of malware is an infinite process. By the day, it's becoming harder. More attackers create advanced malware which is rigid to identify though the number of computer users is growing. In addition to the intricacy of malware, malware is also a huge difficulty in stopping malware attacks. Therefore, as improvement increases, the fight between malware developers and security analyzers never ends. In the following part, we proposed a framework for addressing these malware detection challenges. It would not be a strengthen, but the maximal virus detection dimensions are included in this technique.

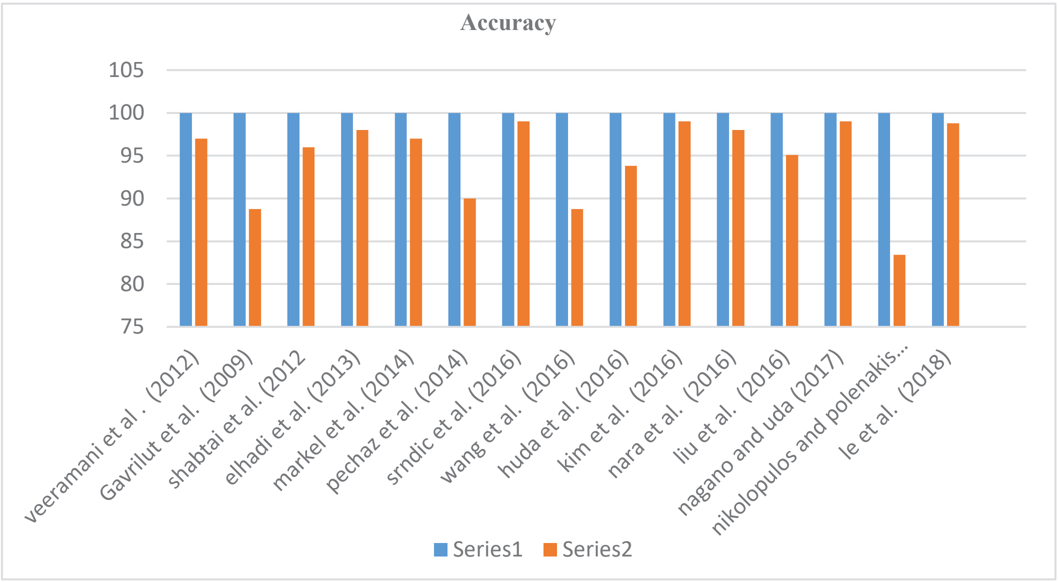


Fig. 5. Chart showing the accuracy contrast of static malware revealing methods

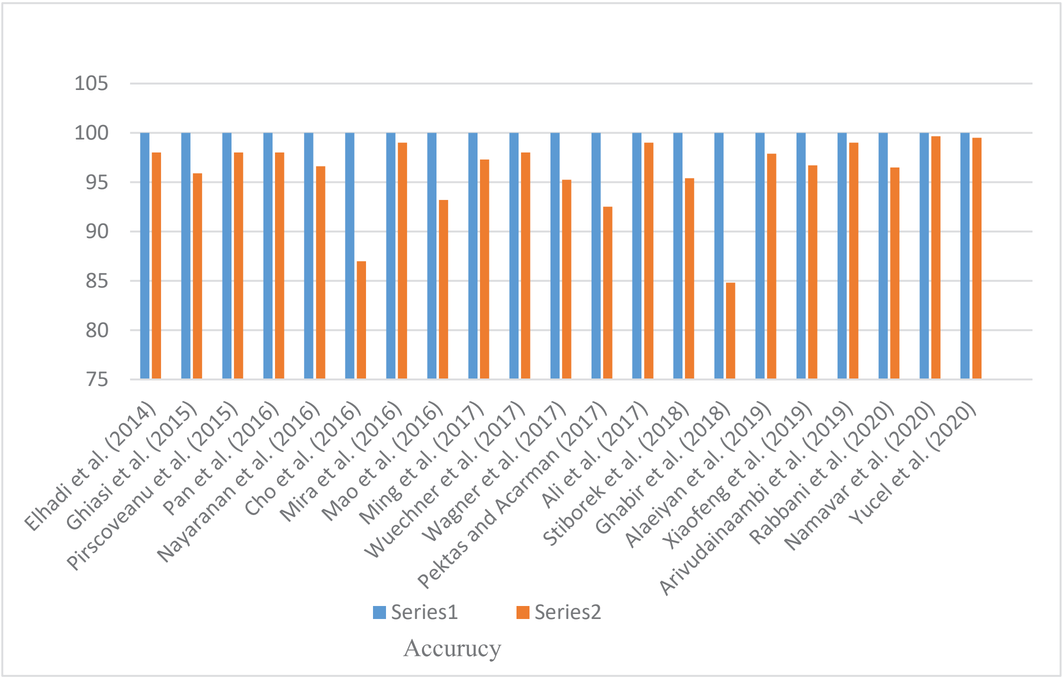


Fig. 6. Comparison Graph of Dynamic Malware Detection Techniques for Accuracy

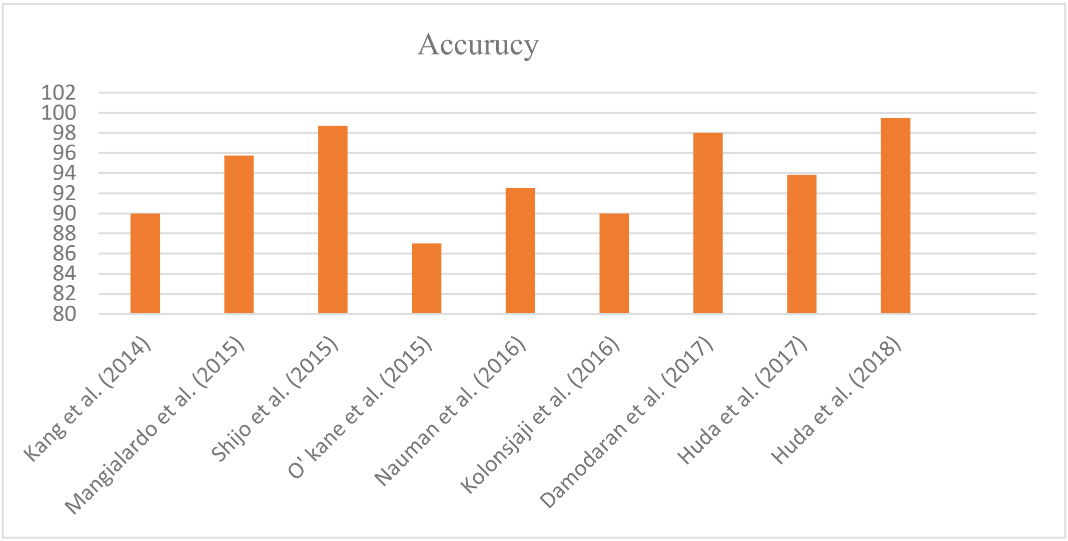


Fig. 7. Comparison Graph of Hybrid Malware Detection Techniques for Accuracy

6. IMPORTANT CRITERIA FOR MALWARE DETECTION SYSTEMS DEVELOPMENT

This section deals with several important elements of malware creation, providing the scientist insights into how to investigate this issue.

6.1. Handling Anti-Analysis Technique

The issue arises when malware is produced employing the method of antistatic or anti-dynamic analysis. In dynamic analysis, malware detects the analytical environment and hides or inhibits actual activity from occurring in this environment. The issue occurs when malware is produced using antistatic or dynamic analysis methods. The malware identifies the analytical environment and conceals or discontinues to run in that environment in dynamic analysis. As a result, throughout the development of the malware analysis system, analysts will be tasked with addressing this issue. In the case of static analysis, malware analysts must develop unpacking software and build a dynamic environment by addressing all the patterns of virtual control devices used by malware to control the analysis environment to do the static analysis. Here are a few indicators of how dynamic malware analysis may be configured for malware samples.

- The Default media access control address of virtual machine may be altered so that malware does not detect virtual machine with virtual machine standard and known media access control access address. Make a note of the names of virtual machines that seem to be the host systems. For example, Ahsan-pc, Sunny, and so on.
- Install all the essential program applications, including Microsoft Office, Adobe Reader, VLC, and others, to give your machine the appearance of a personal computer.
- Create subdirectories for documents in various folders, such as Documents, Desktop, Downloads, and Temp etc. Also, utilize the VM machine for many days to ensure that malware does not identify it as a new computer suitable for malware analysis and mark it as a target. Several documents are generated, as well as cache and other temporary files when accessing the internet after utilizing a virtual machine for personal work.

6.2. Analysis Tools and Environment Setup

Apart from the anti-analysis methodologies, the tools and kind of environment setup especially in dynamic analysis have a big impact on the accuracy of malware classifiers. In static analysis, several static extractor tools were utilized, including IDA Pro disassembler, capstone, Peid, PsStudio, and others. Statistical analyses are conducted on deconstructed files and disassembled malware files. Disassemblers extract more comprehensive characteristics which cannot be obtained with basic tools such as CFF Explorer, Psfile, IOC Finder, etc. The dynamic analytical analysis is influenced by the kind of architecture of the analytics environment. The hypervisor Type 1 is more robust than the hypervisor type 2 when it comes to anti-analytic methods. However, both have advantages and disadvantages that are addressed in Section 2. Thereby, the developer must select techniques for extracting the malware feature required while building an anti-malware system.

6.3. Data Samples for Malware Classifier Training

Samples of malware are gathered in the

academic literature from different sources. Certain sources like Virus Share enable scientists to download millions of malwares, including the latest. Vxheaven has been used to gather the most frequently used samples of malware and give labeled to the malware, but this malware repository has not been updated since 2010. In this manner, malware detectors need new malware samples. As we have found some research, repositories used for the formation of malware are Kaggle, Microsoft, Malware, contagio, the Zoo. The major problem is that the Malware Samples are not labeled with a particular database. Although certain internet systems can mark samples of malware such as VirusTotal and Hybrid Malware Analysis are often utilized. Labeling the malware samples one at a time is a time-intensive procedure, on the other hand. Malware samples from different categories may also be hard to include on a level playing field. A honeypot may also be used to acquire livemalware. Some of the suggested methods utilized honeypots to acquire payloads for malware in real-time, although most research documents do not disclose this. In addition, we may acquire malware samples of our suggested model from antivirus firms. This would be an excellent method of testing malware classifiers with actual malware samples.

6.4. Malware Characteristics Selection

The scalability of the suggested approach is a major issue. It mainly depends on the set of features in which benign and malicious files are classified. With the addition of features, the precision is improved, but the time to scan is increased, restricting malware detection systems in real-time applications. Malware selection is mainly used to distinguish between malware and benign files. Many studies have used API calls, PE headers, and file network activity.

6.5. Machine-Learning Algorithmic Frameworks Selection

Following a comprehensive examination of numerous methodologies, all kinds of machine learning algorithms were applied. Ensemble produces better accuracy than simple classification algorithms (SVM, DT, KNN), but these algorithms take more training time than simple algorithms. Classification methods for Machine Learning may be selected depending on malware size and variety. Another important factor in the accuracy of malware categorization is the parameterization of the algorithms that are used. This issue has not been addressed in depth by the approach that has been proposed. it is very important factor for the accuracy of the malware classifier.

6.6. Future Directives

The challenge now is how one can build a classification of malware that can deal with these problems. We understand now that malware analysts evaluate malware samples and continue to update the malware detection system to block attacks malware. Currently it's almost impossible for signature-based technique to detect new malware. However, the behavioral malware detection technique gives hope for detecting new malware. Therefore, we shall design a hybrid framework, not like hybrid methodologies previously offered. For implementing the suggested method, a two-layered architecture will be utilized. At first, signature-based malware detection is conducted, which, if it fails, will be utilized in second-level Behavior based analytical methods. In the first layer known and basic unbuffered malware may be readily detected, while in the dynamic analysis the malware can be predicted using runtime. When each new virus occurs, the database will be updated and utilized to anticipate future malware. Fig. 8 shows the framework of the malware detection system. This approach collects runtime characteristics by using both sandboxing automation (e.g., Cuckoo sandbox) and a variety of dynamical analysis tools, including Ollydbg, Regshot, Wireshark, and ProcMon, for collecting run time data. Furthermore, tools such as PExplorer, Peview, Peid, and IDA pro are used to extract static features such as strings, imports, and exports from a program. The algorithms for learning machines are then used for malware classification training, which is a kind of machine learning. Using runtime behavior, this approach has the potential to inherit the benefits of both signature-based and behavioral methods. It can successfully identify existing malware while also detecting new malware. We are going to use anti-obfuscation techniques to correctly analyses malware samples. It can be done only after the use of static and dynamic malware analyses to check for malware and benign samples. The approach that has been described is intended to bridge the gap that exists between signature and behavioral methods. Real-time malware detection will be made feasible by the development of a two-layer hybrid technique, which will be implemented in real-time. In addition, various algorithms are used to enhance the resistance of the hybrid model against the adversarial machine learning. The various types of malwares and their description is included in Table 11. A chronological examination of several well-known malwares is presented in Table 12.

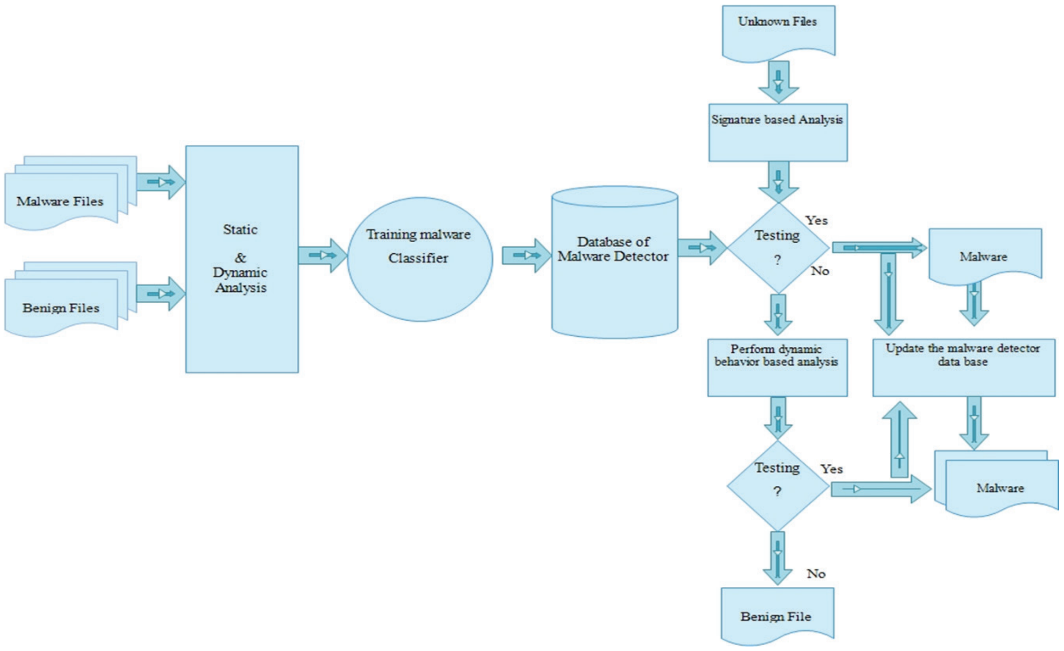

Unknown Files
Signature based Analysis
Malware Files
Benign Files
Static & Dynamic Analysis
Training malware Classifier
Database of Malware Detector
Testing ?
Yes
No
Malware
Perform dynamic behavior based analysis
Update the malware detector data base
Testing ?
Yes
Malware
No
Benign File

Fig. 8. The Proposed schematics of Hybrid Malware Detection Technique.

Table 11: Types of Malwares

Types of malwares	Description	Examples
Virus	More technologically speaking, a computer virus is a harmful code or software intended to alter the way a computer function and to propagate it from one computer to the next. A virus inserts or attaches to a legitimate application or document supporting macros to run its code. A virus may have unintentional or negative consequences in this process by corrupting or deleting information, such as damaging system software.	Common warrior, Creeper, Eliza, Elk cloner, and the Chernobyl Virus.
Worm	A computer worm is a kind of malware, which can be copied and extended across computers. Without human involvement, a worm may reproduce itself and does not need an association with a software application to do damage.	The Storm worm, SQL slammer, The Morris worm, Jerusalem worm, Dabber, and the Code Red II.
Trojan Horse	A Trojan horse is a kind of malware that masquerades as legitimate software, and it is also known as a trojan horse. Hackers and cyber thieves may enter a computer system from Trojans. Social engineering is sometimes used to convince people on their computers to download and execute Trojans.	I love you, Code red, Melissa, Sasser, Zeus, and Conficker.
Spyware	Spyware refers to a kind of software that is designed to steal personal or business information. It is done by performing a sequence of actions without the required user rights, and in certain cases, even in plain view of the user. Advertising, data collection, and changing the computer's user configuration settings are all common actions performed by spyware.	Coolwebsearch (CWS), Gator, Transponder, BlazeFind, Hot as Hell, and ISTbar.
Rootkit	A rootkit is hidden computer software that retains privileged computer access while shielding the presence of a machine. The phrase 'rootkit' consists of the combination of the term's 'root' and 'kit.' A rootkit was originally a collection of tools that enabled administrators to manage a computer or network.	Soni BMG Copy protection Rootkit, NT Rootkit.
Adware	Adware is a program that shows unwanted ads or advertising-supported applications. When pop-up advertisements are shown, your browser's homepage is altered, Spyware is installed, and pop-up advertisements are blasted, Adware programs will bombard your device with advertising. Adware is a term used to describe potentially harmful software.	1080 Solution Assistant, Altnet, Cool web search, Ads by Game Vance,
Bot	A malicious is a kind of harmful malware that infects the host system and establishes a connection with a central server. The server is used as a command and the control center is a botnet or network of infected computers and other devices.	Earth link spammer, cut wail, Storm, Grum, Kraken.
Ransomware	Ransomware is a malware kind that encrypts victims' data. In return for restoring access to data, the attacker then demands a ransom from the victim.	Wannacry, Bad Rabbit, Ryuk, Troldesh, Jigsaw.

Table 12: A chronological examination of several well-known malwares

Year	Malware Attacks
1986	First IBM-PC brain Sector Virus W
1987	The Jerusalem virus was found in Jerusalem and all executables on computers were infected and destroyed after it had just begun Friday the 13th.
1988	A virus of Ping Pong Boot sector was founded at Turin University in Italy.
1989	There is a Trojan AIDS. It requested urgent payment to be dropped.
1990	The chameleon virus was the first polymorphic virus to be created.

1999	An e-mail worm known as Happy99 emerges, hides modifications, and wants the computer user a good New Year. A new version is available. Outlook Express and Internet Explorer were impacted by Windows 95 and 98.
2000	More than a million PCs were infected with "I love you" dubbed as a love bug.
2001	Bad Trans was created to steal credit card information as well as passwords.
2003	Microsoft defects make spreading simple for Agobot and Bolgimo computer worms.
2004	Hacks may access the hard drive of the infected machines using the MyDoom (also Novang), the fastest mail and file-sharing computer worm.
2005	Cellular phone virus, Commwarrior-A propagated via text messages from mobile phones to mobile phones.
2006	The first malware to attack Mac OS X arrived as the low-threat Trojan called OSX/Leap-A.
2007	A Trojan horse called Zeus used a method called keystroke logging to steal bank sensitive information.
2008	The Koobface virus targets people who use MySpace and Facebook.
2009	In the United States and South Korea W32.Dozer Follows a serious cyber-attack.
2010	A Kenzero virus spreads the history of the browser online.
2011	Zeus and SpyEye have joined together to create a new method of attacking mobile phones to collect financial information.
2013	Cryptolocker one of the early ransomware programs crypto lockers had a large global impact and contributed to the rise of ransomware.
2014	Backoff malware infiltrates POS systems to steal information from credit cards.
2016	Cerber One of the most powerful ransomware threats. It's also one of the most common forms of crypto malware. Cerber infected more enterprise PCs than any other ransomware family at one point, according to Microsoft.
2017	WannaCry ransomware almost affected 150 countries including hospitals, banks, warehouses, telecommunication companies, and many other industries.
2018-2020	During that time, many crypto miners and ransomware, such as the COVID19 RAT, the Samsam ransomware, the cyborg ransomware, and the clop ransomware, were developed.

7. CONCLUSION

This study helps to identify malwares using runtime characteristics. It provides the knowledge fraction for harmful software analysis and detection procedures through machine learning algorithms. Malware is now well-known to be highly sophisticated and rapidly changing. Today, it is not only used to disturb users, steal information, and destroy user data but also to enforce its objectives for businesses and nations. The protection of data and resources is a key component in information technology. This is essential because computers or embedded digital devices in every area are utilized to execute activities more quickly and precisely without human involvement. The computer system is more susceptible to hacking because of its wide range of applications. This study covers the development of malware, present status, and techniques of detection. Two malware analysis techniques exist signature based, and behavior based. Signature-based technology has two major flaws that must be addressed. For the time being, signature-based techniques will not be able to identify new or previously unknown malware. Second, different types of malwares may readily outwit the system's detection mechanisms. Behavior-based technique can identify new variants of malware, dynamic techniques in malware-based methods are more robust than signature-based methods. The actual application of dynamic methods nevertheless remains rigid and time-intensive, whereas signature technology is quicker and more efficacious than dynamic techniques in identifying known malware. We spoke about malware methods that were suggested for machine learning to train the classification of malware in this research. This is because it includes a huge number of algorithms that may be used for a variety of malware features. In addition to its accessibility, machine learning algorithms provide many benefits over conventional malware classifications, such as the capacity to get information from file samples, rapidly detect, unexpected changes, and minimize the human work and time spent analyzing malware. In this article, malware detection methods are classified according to analytical methodologies such as static, dynamic, and hybrid techniques. Each method has advantages and disadvantages. These methods, however, produced encouraging results in the categorization of malware in a certain scenario. For example, static methods are quicker, with a reduced false-positive rate, but it is difficult to cope with the obfuscation technique while collecting static characteristics. In this respect, on the contrary, dynamic methods offer a beneficial although implementation in real-time is inconvenient. Finally, we addressed the two-layer malware detection framework with static and dynamic functions that can effectively and reliably detect the new and known malware.

REFERENCES

[1] H. Weijie, X. Jingfeng, W. Yong, H. Lu, K. Zixiao and M. Limin, “MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics”, Computers and Security, vol. 83, pp. 208-233, 2019.

[2] B. Pete, F. Richard, T. Frederick and J. Kevin, “Malware classification using self organising feature maps and machine activity data”, Computers and Security, vol. 73, pp. 399-410, 2018.

[3] N. Usha, T. F. Di, C. V. Aaron, A. Thomas and S. Mark, “Clustering versus SVM for malware detection”, Journal of Computer Virology and Hacking Techniques, vol. 12, pp. 213-224, 2016.

[4] J. A. Namavar, H. Sattar, D. Ali and C. R. Kim kwang, “An improved two-hidden-layer extreme learning machine for malware hunting”, Computers and Security, vol. 89, pp. 10-16, 2020.

[5] F. Massimo and P. F. Leaf, “An open-source cybersecurity training platform for realistic edge-IoT scenarios”, Journal of Systems Architecture, vol. 97, pp. 107-129, 2020.

[6] K. Afreen, Z. Swaleha and Al. S. Muaadhabdo, “An improved pre-processing machine learning approach for cross-sectional imaging of demented older adults”, International Conference of Intelligent Computing and Engineering (ICOICE), IEEE, pp. 1-7, 2019.

[7] P. Jakub, N. Q. Anh, B. Adrian, G. Jonathan and L. Y. Kubo, “a framework for automated efficacy testing of anti-virus behavioral detection with procedure-based malware emulation”, In Proceedings of the 13th International Workshop on Automating Test Case Design, Selection and Evaluation, pp. 37-44, 2022.

[8] B. Andrew, “How deception can change cyber security defenses”, Computer Fraud & Security, vol.5, no.1, pp. 12-14, 2019.

[9] G. Ekta, B. Divya and S. Sanjeev, “Malware analysis and classification: A

survey", Journal of Information Security, vol. 6, 2014.

[10] M. S. Mariam, S. M. Ali, Z. Si-Jing and Y. Hong-Ji, "Conceivable security risks and authentication techniques for smart devices: A comparative evaluation of security practices", International journal of Automation and Computing, vol. 13, pp. 350-363, 2016.

[11] B. Kang, F. Liu, Z. Yun and Y. Liang, "Design of an Internet of Things-based smart home system", International Conference on Intelligent Control and Information Processing, IEEE, vol. 2, pp. 921-924, 2011.

[12] A. Shahid, H, R. Nigel, T. Issa and S. Ibrahim, "A framework for metamorphic malware analysis and real-time detection", Computers and Security, vol. 48, pp. 212-233, 2015.

[13] H. Xin, "Large-scale malware analysis, detection, and signature generation", Doctoral dissertation, University of Michigan, 2011.

[14] R. Kalpika, A. R. Vasudevan, "Detection of Zeus Bot Based on Host and Network Activities", Communications in Computer and Information Science, vol. 746, pp. 978-981, 2017.

[15] E. Nabeil, E. Rashad, H. Alzubair and L. Fagen, "A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud", Journal of Systems Architecture, vol. 102, pp. 10-16, 2020.

[16] J. Daehee, J. Yunjong , L Sungman , P. Minjoon, K. Kuenhwan , K. Donguk and K. B. Byunghoon, "Rethinking anti-emulation techniques for large-scale software deployment", Computers and Security, vol. 83, pp. 182-200, 2019.

[17] M. Samaneh and G. Ali A, "Application of deep learning to cybersecurity: A survey", Neurocomputing, vol. 347, pp. 149-176, 2019.

[18] Z. Weizhe, W. Huanran, H. Hui and L. Peng," DAMBA: detecting android malware by ORGB analysis", IEEE Transactions on Reliability, vol.69, no.1, pp. 55-69, 2020.

[19] L. Liu, W. Bao-sheng, Y. Bo and Z. Qiu-xi, "Automatic malware classification and new malware detection using machine learning", Frontiers of Information Technology & Electronic Engineering, vol. 18, no. 9, pp. 1336-1347, 2017.

[20] K. Ratinder and M. Singh, "Hybrid real-time zero-day malware analysis and reporting system", International Journal of Information Technology and Computer Sciences, vol. 8, pp. 63-73, 2016.

[21] M. Jelena, M. Miroslaw and F. Alberto, "Time, accuracy and power consumption tradeoff in mobile malware detection systems", Computers and Security, vol. 82, pp. 314-328, 2019.

[22] S. Hudan, S. Ferdous and P. Christian, "A survey on forensic investigation of operating system logs", Digital Investigation, vol. 29, pp. 1-20, 2019.

[23] N. Bruce, K. K. Hwan, K. Y. Jin, K.H. Ho, K. T. Yong and L. H. Jae, "Cross-method-based analysis and classification of malicious behavior by api calls extraction", Applied Sciences, vol. 9, no.2, pp. 239, 2019.

[24] Z. Hanqi, X. Xi, M. Francesco, N. Shiguang, M. Fabio and S. A. Kumar, "Classification of ransomware families with machine learning based on N-gram of opcodes", Future Generation Com-

puter Systems, vol. 90, pp. 211-221, 2019.

[25] N. Jose, P. Araujo, P. Donald M and R. C. Ghedini, "MULTS: A multi-cloud fault-tolerant architecture to manage transient servers in cloud computing", Journal of Systems Architecture, vol. 101, pp. 101-108, 2019.

[26] R. Veeramani and R. Nitin, "Windows api based malware detection and framework analysis", In International Conference on Networks and Cyber Security, vol. 25, 2012.

[27] C. Mihai, J. Somesh, K. Johannes, K. Stefan and V. Helmut, "Software transformations to improve malware detection", Journal in Computer Virology, vol. 3, pp. 253-265, 2007.

[28] O. Yoshihiro, "Trends of anti-analysis operations of malwares observed in API call logs", Journal of Computer Virology and Hacking Techniques, vol. 14, no. 1, pp. 69-85, 2018.

[29] C. Sibi S and V. V. Sangeetha, "A survey on malware analysis and mitigation techniques", Computer Science Review, vol. 32, pp. 1-23, 2019.

[30] D. Jaime, S. Igor, C. Xabier, P. Yoseba K and B. Pablo G, "Automatic behaviour-based analysis and classification system for malware detection", International Conference on Enterprise Information Systems, vol. 2, pp. 395-399, 2010.

[31] D. Yuxin, X. Xiaoling, C. Sheng and L. Ye, "A malware detection method based on family behavior graph", Computers and Security, vol. 73, pp. 73-86, 2018.

[32] S. Jagsir and S. Jaswinder, "A survey on machine learning-based malware detection in executable files", Journal of Systems Architecture, vol. 112, pp. 10-18, 2021.

[33] R. Anusmita and N. Asoke, "Introduction to Malware and Malware Analysis: A brief overview", International Journal, vol. 4, no.10, 2016.

[34] W. Huanran, H. Hui and Z. Weizhe, "Demadroid: Object reference graph-based malware detection in Android", Security and Communication Networks, vol. 2018, 2018.

[35] B. Tao, T. Takeshi, G. Shanqing, I. Daisuke and N. Koji, "Integration of multi-modal features for android malware detection using linear SVM", In 2016 11th Asia Joint Conference on Information Security (Asia JCIS), IEEE, pp. 141-146, 2016.

[36] B. Ulrich, K. Engin and K. Christopher, "Improving the efficiency of dynamic malware analysis", In Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 1871-1878, 2010.

[37] W. Gerard, S. Radu and D. Alexandre, "Malware behaviour analysis", Journal in Computer Virology, vol. 4, pp. 279-287, 2008.

[38] M. Asit and T. Shashikala, "Virtual machine introspection: towards bridging the semantic gap", Journal of Cloud Computing, vol. 3, no.1, pp. 1-14, 2014.

[39] V. Jiri and P. Martin, "Virtualization of operating system using type-2 Hypervisor", In Software Engineering Perspectives and Application in Intelligent Systems: Proceedings of the 5th Computer Science On-line Conference 2016 (CSOC2016), Springer International Publishing, vol. 25, pp. 239-247, 2016.

[40] M. Andreas, K. Christopher and K. Engin, “Exploring multiple execution paths for malware analysis”, In 2007 IEEE Symposium on Security and Privacy (SP'07), IEEE, pp.231-245, 2007.

[41] K. Bojan, Z. Apostolis, L. Tamas, W. George and E. Claudia, “Adaptive semantics-aware malware classification”, In Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, Springer International Publishing, vol. 13, pp. 419-439, 2016.

[42] R. Edward, Z. Richard, C. Russell, S. Jared, Y. Paul, W. Rebecca, T. Anna, M. Mark and N. Charles, “An investigation of byte n-gram features for malware classification”, Journal of Computer Virology and Hacking Techniques, vol. 14, pp. 1-20, 2018.

[43] M. Zane and B. Michael, “Building a machine learning classifier for malware detection”, In 2014 second workshop on anti-malware testing research (Water), IEEE, pp. 1-4, 2014.

[44] S. Michael and H. Andrew, “A Practical malware analysis: the hands-on guide to dissecting malicious software”. Starch Press, 2012.

[45] W. Tzu-Yen and W. Chin-Hsiung, “Detection of packed executables using support vector machines”, International Conference on Machine Learning and Cybernetics, IEEE, vol. 2, pp. 717-722, 2011.

[46] A. Satheesh and R. Kumaravelu, “A mathematical model of HMST model on malware static analysis”, International Journal of Information Security and Privacy, vol. 13, no. 2, pp. 86-103, 2019.

[47] A. Imad and L. Saiida, “A new classification based model for malicious PE files detection”, International Journal of Computer Network and Information Security, vol. 9, no.6, pp.1-7, 2019.

[48] L. Michael, A. Steven, H. Blake, R. Matthew, “Malware Analyst's Cookbook and DVD”, Wiley Publishing, 2011.

[49] L. Xiaojing, Y. Kan, W. XiaoFeng, L. Zhou, X. Luyi and B. Raheem, “Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence”, ACM SIGSAC conference on computer and communications security, pp. 755-766, 2016.

[50] S. Sebastian and K. Stefan, “Code obfuscation against static and dynamic reverse engineering”, In Information Hiding: 13th International Conference, IH 2011, Prague, Czech Republic, Revised Selected Papers, Springer Berlin Heidelberg, vol. 13, pp. 270-284, 2011.

[51] C. Michael , “Scanning memory with Yara”, Digital Investigation, vol. 20, pp. 34-38, 2017.

[52] S. Nikolaos, B. Chafika, A. Omar and A. Ameer, “Forensic malware analysis: The value of fuzzy hashing algorithms in identifying similarities”, IEEE Trustcom/ Big Data SE/ ISPA, IEEE, pp. 1782-1787, 2016.

[53] K. Abhishek, G. Suchandra and G. Ratan, “Detecting obfuscated viruses using cosine similarity analysis”, Asia International Conference on Modelling and Simulation (AMS'07), IEEE, pp. 165-170, 2007.

[54] B. Danilo, M. Lorenzo, M. Mattia,

"Code normalization for self-mutating malware", IEEE Security and Privacy, vol. 5, no.2, pp. 46-54, 2007.

[55] Z. Boyun, Y. Jianping, H. Jingbo, Z. Dingxing and W. Shulin, "Malicious codes detection based on ensemble learning", In Autonomic and Trusted Computing: 4th International Conference, ATC 2007, Hong Kong, China, July 11-13, 2007. Proceedings 4, Springer Berlin Heidelberg, pp. 468-477, 2007.

[56] M. Robert, F. Clint, T. Nir, B. Eugene, G. Marina, D. Shlomi and E. Yuval, "Unknown malcode detection using opcode representation", In Intelligence and Security Informatics: First European Conference, Euro ISI 2008, Esbjerg, Denmark, Proceedings, Springer Berlin Heidelberg, pp. 204-215, 2008.

[57] B. S. Mojtaba, J. Saeed and T. Asghar, "PbMMD: A novel policy based multi-process malware detection", Engineering Applications of Artificial Intelligence, vol. 60, pp. 57-70, 2017.

[58] N. Vivens, X. Zhifeng, M. V. Rao, M. Ke and X. Yang, "Network forensics analysis using Wireshark", International Journal of Security and Networks, vol. 10, no.2, pp. 91-106, 2015.

[59] H. Nazrul, B. Monowar H, B. Ram Charan, B. Dhruba K and K. Jugal K, "Network attacks: Taxonomy, tools and systems", Journal of Network and Computer Applications, vol. 40, pp. 307-324, 2014.

[60] E. Eldad, "Reversing: secrets of reverse engineering", John Wiley & Sons, 2011.

[61] G. Daniel, M. Carles and P. Jordi, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges", Journal of Network and Computer Applications, vol. 153, pp. 102526, 2020.

[62] R. Chathuranga and J. Aruna, "An efficient approach for advanced malware analysis using memory forensic technique", In 2017 IEEE Trustcom/ Big Data SE/ ICESS, IEEE, pp. 1145-1150, 2017.

[63] K. Ilker, "A basic malware analysis method", Computer Fraud and Security, vol. 2019, no. 6, pp. 11-19, 2019.

[64] K. Joakim, "Fundamentals of Digital Forensics", Springer International Publishing, 2020.

[65] P. Radu, H. Steven, L. Thor, S. Matija, P. Jens and C. Alexandre, "Analysis of malware behavior: Type classification using machine learning", International conference on cyber situational awareness, data analytics and assessment (Cyber SA), IEEE, pp. 1-7, 2015.

[66] A. Omer and S. Refik, "Investigation of possibilities to detect malware using existing tools", 14th International Conference on Computer Systems and Applications. IEEE, pp. 1277-1284, 2017.

[67] M. Q. Ali, A. Irfan and Y. Muhammad, "Cloud Intell: An intelligent malware detection system", Future Generation Computer Systems, vol. 86, pp. 1042-1053, 2018.

[68] G. Kent, S. Scott, H. Xin and C. Tzi-cker, "Automatic generation of string signatures for malware detection", Recent Advances in Intrusion Detection: 12th International Symposium, RAID 2009, Saint-Malo, France, Proceedings, Springer Berlin Heidelberg, vol. 12, pp.

101-120, 2009.

[69] G. Dragoş, C. Mihai, A. Dan and C. Liviu, "Malware detection using machine learning", International multi-conference on computer science and information technology, IEEE, pp. 735-741, 2009.

[70] B. Philippe, G. Isabelle and M. Jean-Yves, "Behavior abstraction in malware analysis", In Runtime Verification: First International Conference, RV 2010, Proceedings, Springer Berlin Heidelberg, vol. 1, pp. 168-182, 2010.

[71] C. Sang Kil, M. Iulian, J. Jiyong, T. John, B. David and A. David G, "Split Screen: Enabling efficient, distributed malware detection", Journal of Communications and Networks, vol. 13, no.2, pp. 187-200, 2011.

[72] S. Asaf, M, Robert, F. Clint, D. Shlomi and E. Yuval, "Detecting unknown malicious code by applying classification techniques on opcode patterns", Security Informatics, vol. 1, no.1, pp. 1-22, 2012.

[73] S. Asaf, M. Robert, E. Yuval and G. Chanan, "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey", Information Security Technical Report, vol. 14, no.1, pp. 16-29, 2009.

[74] E. A. Ahmed E, M. M. Aizaini and B. Bazara, "Improving the detection of malware behaviour using simplified data dependent API call graph", International Journal of Security and its Applications, vol. 7, no. 5, pp. 29-42, 2013.

[75] P. Bassir, J. M. Vafaie and J. Mehrdad, "Malware detection using hidden Markov model based on Markov blanket feature selection method". International congress on technology, communication and knowledge, IEEE, pp. 558-563, 2015.

[76] S. Nedim and L. Pavel, "Hidost: a static machine-learning-based detector of malicious files". EURASIP Journal on Information Security, vol. 2016, pp. 1-20, 2016.

[77] K. Dong Hee, W. Sang Uk, L. Dong Kyu and C. Tai Myoung, "Static detection of malware and benign executable using machine learning algorithm". Eighth International Conference on Evolving Internet, pp. 14-19, 2016.

[78] H. Shamsul, A. Jemal, A. Mamoun, A. Mali, I. Rafiqul and Y. John, "Hybrids of support vector machine wrapper and filter based framework for malware detection". Future Generation Computer Systems, vol. 55, pp. 376-390, 2016.

[79] N. Yuta and U. Ryuya, "Static analysis with paragraph vector for malware detection". In Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, pp. 1-7, 2017.

[80] W. Cheng, Q. Zheng, Z. Jixin and Y. Hui, "A malware variants detection methodology with an opcode based feature method and a fast density based clustering algorithm". 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), IEEE, pp. 481-487, 2016.

[81] M. Aziz, A. Omar and M. Manar, "AMAL: high-fidelity, behavior-based automated malware analysis and classification". Computers and Security, vol. 52, pp. 251-266, 2015.

[82] Y. Yanfang, W. Dingding, L. Tao, Y.

Dongyi and J. Qingshan, “An intelligent PE-malware detection system based on association mining”, Journal in computer virology, vol. 4, pp. 323-334, 2008.

[83] B. Michael, O. Jon, A. Jon, M. Z. Morley, J. Farnam and N. Jose, “Automated classification and analysis of internet malware”, Recent Advances in Intrusion Detection: 10th International Symposium, RAID, vol. 10, pp. 178-197, 2007.

[84] R. Konrad, T. Philipp, W. Carsten and H. Thorsten, “Automatic analysis of malware behavior using machine learning”, Journal of Computer Security, vol. 19, no.4, pp. 639-668, 2011.

[85] V. Mihai, G. Laura and T. Nicolae, “Practical malware analysis based on sandboxing”, Ro Edu Net Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference, IEEE, pp. 1-6, 2014.

[86] H. Jozsef, M. Yoan, I. Alexander and L. Amaury, “Methodology for behavioral-based malware analysis and detection using random projections and k-nearest neighbors classifiers”, International Conference on Computational Intelligence and Security, IEEE, pp. 1016-1023, 2011.

[87] G. Mahboobe, S. Ashkan and S. Zahra, “Dynamic VSA: a framework for malware detection based on register contents”, Engineering Applications of Artificial Intelligence, vol. 44, pp. 111-122, 2015.

[88] P. Zhi-Peng, F. Chao and T. Chao-Jing, “Malware classification based on the behavior analysis and back propagation neural network”. In ITM Web of Conferences, EDP Sciences, vol.7, pp. 20-28, Nov. 2016.

[89] H. Shamsul, M. Suruz, H. M. Mehedi, I. Rafiqul, Y. John, A. Majed and A. Ahmad, “Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data”, Information Sciences, vol. 379, pp. 211-228, 2017.

[90] M. Fahad, H. Wei and B. Antony, “Novel malware detection methods by using LCS and LCSS”, 22nd International Conference on Automation and Computing (ICAC), IEEE, pp. 554-559, 2016.

[91] H. Eduardo, M. Rubén S and L. Ignacio M, “Evaluating the reliability of computational grids from the end user’s point of view”, Journal of Systems Architecture, vol. 52, no. 12, pp. 727-736, 2006.

[92] N. B. Narayanan, D. Ouboti and K. Temesguen, “Performance analysis of machine learning and pattern recognition algorithms for malware classification”, IEEE national aerospace and electronics conference (NAECON) and ohio innovation summit (OIS), IEEE, pp. 338-342, 2016.

[93] C. In Kyeom, K. T. Guen, S. Y. Jin, R. Minsoo and I. E. Gyu, “Malware analysis and classification using sequence alignments”, Intelligent Automation and Soft Computing, vol. 22, no.3, pp. 371-377, 2016.

[94] M. Jiang, X. Zhi, L. Pengwei, W. Dinghao, L. Peng and M. Bing, “Impeding behavior-based malware analysis via replacement attacks to malware specifications”, Journal of Computer Virology and Hacking Techniques, vol. 13, pp.193-207, 2017.

[95] W. Markus, R. Alexander, T. Niklas and

A. Wolfgang, "A knowledge-assisted visual malware analysis system: Design, validation, and reflection of KAMAS", Computers and Security, vol. 67, pp. 1-15, 2017.

[96] N. Stavros D and P, Iosif, "A graph-based model for malware detection and classification using system-call groups", Journal of Computer Virology and Hacking Techniques, vol. 13, no.1, pp. 29-46, 2017.

[97] L. Quan, B. Oisin, M. N. Brian and S. Mark, "Deep learning at the shallow end: Malware classification for non-domain experts", Digital Investigation, vol. 26, pp. S118-S126, 2018.

[98] P. Abdurrahman and A. Tankut, "Classification of malware families based on runtime behaviors", Journal of information security and applications, vol. 37, pp. 91-100, 2017.

[99] S. Jan, P. Tomás and R. Martin, "Multiple instance learning for malware classification", Expert Systems with Applications, vol. 93, pp. 346-357, 2018.

[100] G. Ibrahim, H. Mohammad, P. Vaclav, H. Liangxiu, H. Robert, R. Khaled, and A. Francisco J, "Detection of advanced persistent threat using machine-learning correlation analysis", Future Generation Computer Systems, vol. 89, pp. 349-359, 2018.

[101] G. Ibrahim, H. Mohammad, P. Vaclav, H. Liangxiu, H. Robert, R. Khaled, and A. Francisco J, "Detection of advanced persistent threat using machine-learning correlation analysis", Future Generation Computer Systems, vol. 89, pp. 349-359, 2018.

[102] X. Lu, F. Jiang, X. Zhou, S. Yi and J. Sha and L. Pietro, "ASSCA: API sequence and statistics features combined architecture for malware detection", Computer Networks, vol. 157, pp. 99-111, 2019.

[103] D. Arivudainambi, K.A. Varun Kumar and P. Visu, "Malware traffic classification using principal component analysis and artificial neural network for extreme surveillance", Computer Communications, vol. 147, pp. 50-57, 2019.

[104] Y. Cagatay and K. Ahmet, "Imaging and evaluating the memory access for malware", Forensic Science International: Digital Investigation, vol. 32, pp. 20-27, 2020.

[105] R. Mahdi, W. Yong Li, K. Reza, J. Hamed, Z. Ruxin and H. Peng, "A hybrid machine learning approach for malicious behaviour detection and recognition in cloud computing", Journal of Network and Computer Applications, vol. 151, pp. 12-19, 2020.

[106] E. A. Ahmed E, M. M. Aizaini, B. Bazara- IA and H. Hentabli, "Enhancing the detection of metamorphic malware using call graphs", Computers and Security, vol. 46, pp. 62-78, 2014.

[107] R. Jesse C, K. Roger I, L. Scott M and C. Robert K, "Detection of injected, dynamically generated, and obfuscated malicious code", ACM workshop on Rapid malcode, pp. 76-82, 2003.

[108] C. Michael, "A protocol graph based anomaly detection system", Doctoral dissertation, Carnegie Mellon University, 2008.

[109] M. R. Jose and D. J. Cesar, "Integrating static and dynamic malware analysis using machine learning", IEEE Latin America Transactions, vol. 13, no. 9, pp. 3080-3087, 2015.

[110] S. PV and S. AJPCS, "Integrated static and dynamic analysis for malware detection", Procedia Computer Science, vol. 46, pp. 804-811, 2015.

[111] E. E. Inang, B. Chafika, A. Ameer and W. Paul, "Analysis of malware behaviour: Using data mining clustering techniques to support forensics investigation", Fifth Cybercrime and Trustworthy Computing Conference, IEEE, pp. 54-63, 2014.

[112] W. Ahsan, I. Azhar, L. Jahanzaib, N. Ahsan and B. Anas, "A novel approach of unprivileged keylogger detection", 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), IEEE, pp. 1-6, 2019.

[113] N. Mohammad, A. Nouman and Y. Jing Tao, "A three-way decision making approach to malware analysis using probabilistic rough sets", Information Sciences, vol. 37, no. 4, pp. 193-209, 2016.

[114] N. Mohammad, A. Nouman and Y. Jing Tao, "Detecting malware evolution using support vector machines", Expert Systems with Applications, vol. 143, pp. 113022, 2020.

[115] P. Avi, R. Brian, K. Lee, H. Michael, C. Catherine, O. Alison, T. Glenn, R. S. Neal, P. Terry, and T. Jason, "Artificial intelligence based malware analysis", arXiv preprint, pp.16-23, 2017.

[116] H. Shamsul, I. Rafiqul, A. Jemal, Y. John, H. M. Mehedi and F. Giancarlo, "A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection", Future Generation Computer Systems, vol. 1, no. 83, pp. 193-207, 2018.

[117] S. Robert, X. Lifan, K. William, V. Tristan, F. Teague, H. John, P. Zachary, S. Corey, S. Joshua and C. John, "Parallelization of machine learning applied to call graphs of binaries for malware detection", 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), IEEE, pp. 69-77, 2017.

[118] K. Youngjoon, K. Eunjin and K. Huy Kang, "A novel approach to detect malware based on API call sequence analysis", International Journal of Distributed Sensor Networks, vol. 11, no.6, pp. 659101, 2015.

[119] M. Weixuan, C. Zhongmin, T. Don, F. Qian and G. Xiaohong, "Security importance assessment for system objects and malware detection", Computers and Security, vol. 68, pp. 47-68, 2017.

[120] O. Philip, S. Sezer, and K. McLaughlin, "Detecting obfuscated malware using reduced opcode set and optimised runtime trace", Security Informatics, vol. 5, pp. 1-12, 2016.

[121] D. Anusha, T. Fabio Di, V. C. Aaron, A. Thomas H and S. Mark, "A comparison of static, dynamic, and hybrid analysis for malware detection", Journal of Computer Virology and Hacking Techniques, vol. 13, pp. 1-12, 2017.