ARTICLE IN PRESS

Journal of Systems Architecture xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

A survey on machine learning-based malware detection in executable files

Jagsir Singh [*], Jaswinder Singh

Department of Computer Science and Engineering, Punjabi University Patiala, India

ARTICLE INFO

Keywords:
Dynamic analysis
Machine learning
Malicious software
Obfuscation techniques
Static analysis
API calls
Ransomware

ABSTRACT

In last decade, a proliferation growth in the development of computer malware has been done. Nowadays, cybercriminals (attacker) use malware as a weapon to carry out the attacks on the computer systems. Internet is the main media to execute the malware attack on the computer systems through emails, malicious websites and by drive and download software. Malicious software can be a virus, trojan horse, worms, rootkits, adware or ransomware. Malware and benign samples are analyzed using static or dynamic analysis techniques. After analysis unique features are extracted to distinguish the malware and benign files. The efficiency of the malware detection system depends on how effectively discriminative malware features are extracted through the analysis techniques. There are various methods to set up the analysis environments using various static and dynamic tools. The second phase is to train the malware classifiers. Earlier traditional methods were used but nowadays machine learning algorithms are used for malware classification which can cope with complexity and pace of malware development. In this paper detailed study of malware detection techniques using machine learning algorithms are presented. In addition, this paper discusses various challenges for developing malware classifiers. At last future directive is discussed to develop an effective malware detection system by handling various issues in malware detection.

1. Introduction

Malware are the malicious programs which deliberately execute the destructive functions [1,2]. The malicious programs are classified into many categories based on their behaviour and executing processes such as worms, viruses, trojan horse, rootkits, backdoor, spyware, logic bombs, adware, and ransomware. Computer systems are attacked to destroy computer resources, for getting financial benefits, for stealing private and confidential data and using the computing resources, to make the services unavailable to the system, etc. [3–6].

The era of malware development was started around in years of the 1980s when computer virus the Brain was developed in 1986 and spread very fast which damaged thousands of computer systems. At that time pace of malware, development was slow because that was the growing phase of computerization. But now the time has changed, a thousand of new malware are developed every day [7]. In addition to the proliferation growth of the malware, the new malware are much targeted, zero-day, stealthy and persistent as compared to the traditional malware which were broad, open and one time executed [8–10]. Also, the newly developed malware are very much complex; specially designed to exploits the vulnerabilities of the computer system. To bypass the malware analysis and detection system, malware developers use various obfuscation techniques (anti-analysis techniques) [11,12]. Furthermore using the encryption and encoding techniques, the complex malicious program like polymorphic, metamorphic and packed malware are developed which are very hard to detect and analyze them [13–16]. In general, the several spreading vectors are used to transmit the malware from one computer system to another which are discussed in Table 1.

There is an endless competition between malware developers and analysts. Both research communities are working in parallel, one is developing the malware detection system and other is designing the malicious software to compromise the detection software for attacking the computer and networks resources. Malware analysts analyze the known malware and aim to detect the malware to avoid the attack on the computer systems [17,18]. Malware are detected using either signature-based or behaviour-based techniques. The signature-based malware detection systems are fast and efficient but can be easily evaded by the obfuscated malware [19,20]. On the other side, behaviour-based techniques are more resilient to the obfuscation technique. However, behaviour-based techniques are very time-consuming. Not only the signature and behaviour-based malware detection techniques have been developed but also many hybrid techniques have been developed to inherit the advantages of both techniques. Hybrid techniques are designed to handle the issues of signature and

* Corresponding author.
E-mail addresses: erjagsirsingh18@gmail.com (J. Singh), dr.jaswinder@pbi.ac.in (J. Singh).

https://doi.org/10.1016/j.sysarc.2020.101861
Received 21 March 2020; Received in revised form 13 July 2020; Accepted 15 August 2020
Available online 22 August 2020
1383-7621/© 2020 Elsevier B.V. All rights reserved.

Table 1
Spreading methods.

Spreading Methods	Description
Drive-by download	Drive-by download is the unintentional downloading of malware while surfing the Internet. Most of the time malware are embedded into the legitimate software. When the user downloads the legitimate software then the malware are also downloaded. Drive-by downloads can occur when opening an e-mail attachment, or clicking on an unreliable pop-up or clicking a link. For example, LoadPCBanker is a drive-by download malware which used Googles Sites to transmit from one PC to other.
Vulnerability	Vulnerability is a security flaw present in the system or application software which facilitates the attacker to exploit it to inject the malware into the system. It can be a programming error, design flaw or some other type of inbuilt flaw in an application or system software. A very famous ransomware WannaCry (2017) exploited the vulnerability of Windows 7 to encrypt the data of millions of user. This malware used the present flaw (weakness) in SMB of Windows 7 Operating System. The user who had patched this vulnerability they did not get affected and the rest of them lost their data. It is the most dangerous type of spreading vector which is very difficult to cope with. An attacker finds the vulnerability in the operating system or software and tries to write the malware to use the present flaw which can do the maximum damage. Therefore, the user needs to update the system regularly to handle these types of malware.
Backdoor	Backdoor are the entry points through which other malware payloads are downloaded into the systems. These holes are either left open in the software for the debugging purpose or created by the Trojan horses. These types of the malware itself do not disrupt the system but they open the doors to other malware to enter into the system by which other malware can enter the system. The FinSpy is a well-known backdoor. Once it is installed on a system, it provides the entry point to an attacker for downloading and executing the malicious script on the system remotely.
Removable Drives	Removable drives include flash drives or hard disks. These are the common ways to spread malware from one system to another. If we attach the removable drive to the infected system and then attach to an uninfected system then it creates the possibility to spread the malware even if that system has anti-malware software. User should always be conscious about using flash drives to share the data between the systems. It can spread every type of malware like a virus, worms, ransomware, etc.
Homogeneity	A setup of systems of same operating software and connected through the same network becomes the reason of worm malware propagation from one system to another.

behaviour-based approaches. The detection of zero-day malware is challenging because these malware exploits the new vulnerabilities which are not discovered yet [21,22]. Whenever new software is developed the crackers try to find the vulnerability in that and exploit that to compromise the security of the software. Therefore, there is a need to patch and develop the malware detection systems which can handle the severe attacks [23,24]. Since the inception of malware attacks on the computer system, the defending system has been developed.

As we know the pace of malware development is very high and complexity in malware writing is increasing day by day. Thus, the detection of malware using traditional approaches rule-based, graph-based, entropy-based, etc. is not possible. To tackle this problem, machine learning provides a solution to possibly develop malware classifiers to detect the new and variants of malware [25,26]. In literature, various machine learning-based techniques have been proposed using supervised and unsupervised algorithms [27,28]. After reviewing the proposed machine learning-based detection approaches, two main points are inferred. First, classification algorithms for training the classifiers and second is malware features which are extracted using dynamic and static malware analysis techniques. These two factors have an impact on the accuracy of malware classifiers. For classifier training, both simple classification algorithms SVM, DT, NB and ensemble classification algorithms RF, Ada boosting were used and produced good results in malware detection [29,30]. Generally, ensemble classifiers produce better results. Each classification algorithms has its advantages and limitations. Also, the feature representation makes a huge difference in the detection rate of the classifiers. To combat the malware we need much more reliable automated detection tools. Various researchers have developed automatic analysis cognitive systems to handle the highly catastrophic zero-day malware which can resistant malware attack on them as well. For making such automated tools the continuous analysis of malware is required to update the detection tools with the pattern and behaviour of new malware and variants of existing malware.

1.1. Contributions

- This research paper presents the evolution and present state of malware detection systems.
- Various machine learning classification algorithms are discussed and compared.
- Recent signature-based, behaviour-based and hybrid machine learning classifiers have been discussed. It shares the fraction in the body of knowledge of proposed malware detection systems with their advantages and limitations.
- In this paper, we have discussed several important factors which play a vital role in the performance of the malware classifiers. Also, a proposed hybrid model for malware detection using ML is presented. At last, the paper is concluded and the future directive is discussed.

1.2. Overview

1.2.1. Scope

In this paper, the emphasis is put on the ML-based malware detection techniques to develop an automatic intelligent malware detection system for the executable files. This paper discusses the proposed work for detecting the executable files and provides insight into ongoing research in malware detection using the various features and approaches.

1.2.2. Evaluation

An exhaustive survey of machine learning-based malware detection techniques is done. Due to intense unevenness in the size of used datasets, ML algorithms and assessment methodologies, it becomes very difficult to efficiently compare the proposed detection techniques. Though, in this paper, the performances of proposed ML-based malware classifiers are compared and summarized using some parameters.

1.2.3. Organization

This survey paper is organized in many sections. Section 2 introduces malware analysis techniques. Section 3 discusses the machine learning algorithms used for malware classification. Section 4 presents the survey of various proposed approaches: static, dynamic and hybrid. Section 5 evaluates the performances of the reviewed papers and Section 6 explains the various factors for developing the malware classifiers. Finally, the future scope is described and paper is concluded.

2. Malware analysis

Malware samples are analyzed for extracting the features which can be used to detect them. There are two basic malware analysis techniques: static and dynamic analyses which are described as follow.

2.1. Static malware analysis

In this analysis, features are extracted without executing the malware samples. The various static features are extracted after analysis like hash value, N-grams, opcodes, strings, and PE header information are extracted. Using these features, malware detection software are designed (antivirus, IDSs, etc.) [31,32]. We analyze malware with or without applying reverse engineering on malware samples. To examine the malware sample at the code level, executable malware files are disassembled into the assembly language code. For that, some most widely used debuggers and disassemblers like Ollydbg, IDA Pro, capstone and WinDbg disassemblers are used to convert the binary files into the assembly code [33,34]. Assembly code is examined to find the execution flow of file, structure or pattern of malicious activity which can be utilized for detecting new or variants of existing malware. It is a very time-consuming process to investigate assembly code to find the execution pattern and features. Additionally, code obfuscation techniques make the analyst's job much harder. Malware developers use various obfuscation techniques such as code encryption, reordering the program instructions and dead code insertion technique to evade the malware analysis [35,36]. Table 2 discusses the brief description of static analysis tools.

2.2. Dynamic malware analysis

In this analysis technique, malware files are executed and runtime activities of malware are captured. The runtime behaviour consists of file system operations, registry key changes, process execution and network activities [47–51]. Unlike static analysis, dynamic analysis is based interaction of malware with the computer system. Malware samples are executed in the controlled virtual environment because if the malware file is directly run on the host system then, it will harm the host operating system [52–55]. A virtual environment is created using virtualization tools like the Virtual Box or VMware. When a malware file is running in a monitored environment various activities are observed such as the creation of new files, deletion of system or user files, new log entries, registry key changes, URL accessed, API Calls, downloading malware or sending data to command and control system etc. Based on these activities, the file is considered as a benign file or malicious file. The files which are not disassembled or examined properly using static analysis can be analyzed through the dynamic analysis. Table 3 discusses the brief description of dynamic analysis tools.

2.2.1. Dynamic analysis environments

Dynamic analysis environment is set up using Type-1 Hypervisor and Type-2 hypervisor for gathering the behavioural features of the executable files [67,68]. Type-1 is called bare metal in which hypervisors directly run on the hardware machine to manage and control the guest machines. Examples of type-1 hypervisors are XEN, Oracle VM Server, Nutanix AHV, AntsleOs, VMware ESXi Microsoft Hyper-V, etc. Fig. 1 shows the architecture of type-1 hypervisor.

Types -2 hypervisors runs within the guest machines. To set up the type-2 dynamic analysis environment, tools like VMware, VirtualBox and Microsoft Hyper-V are used. Examples of type-2 hypervisors are VMware Player, VirtualBox, VMware Workstation, QEM, etc. Fig. 2 shows the architecture of type-2 architecture. These figures give the level of abstraction in malware execution. Based on these architectures both types have advantages and limitations which are mentioned in Table 4.

Table 5 shows the comparison between static and dynamic analysis techniques. In this table advantages and disadvantages are briefly described.

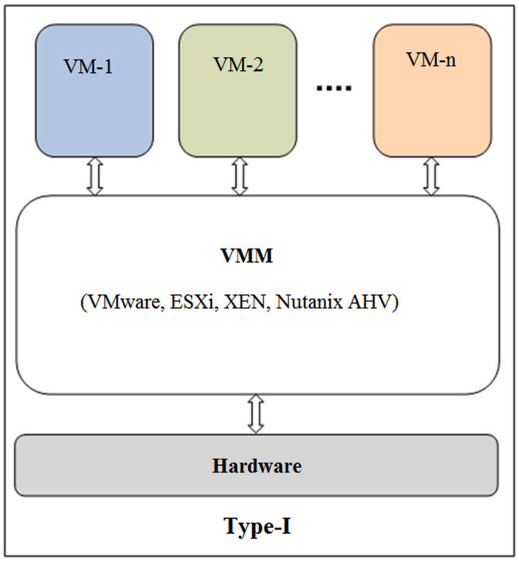

Fig. 1. Architecture of Hypervisor Type-I.

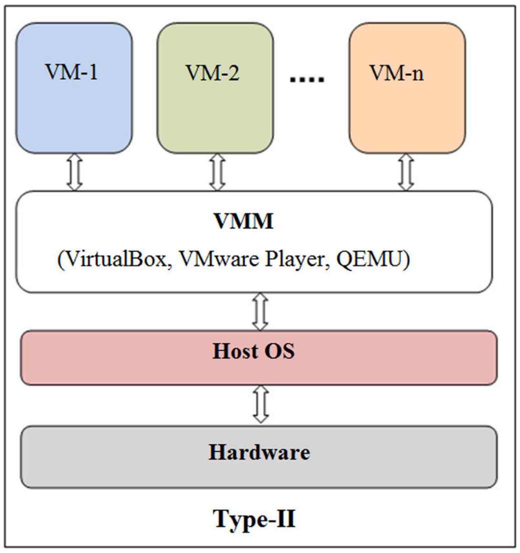

Fig. 2. Architecture of Hypervisor Type-II.

3. Machine learning for malware detection

Machine Learning (ML) is very helpful in the classification and clustering of malware in the present time. In literature, a lot of research work has been done for classifying benign and malware files. ML algorithms provide more option and space for developing a more accurate model by considering more features of malware and benign samples [26,69–71]. ML has a very wide scope in the field of computer security like for malicious URL detection, intrusion detection and malware detection. Malware samples are analyzed and feature set of extracted information for training the classifier. Fig. 3 shows the schematic diagram of the malware detection system using machine learning algorithms. This figure illustrates the basic architecture of machine learning classifiers like in other problem domain. First features are extracted. Then, feature selection and representation is done. Finally, classification algorithms are applied to train the malware classifiers.

Another major benefit of using machine learning in malware detection is that it can develop a model to detect unknown malware. The

Table 2
Static analysis tools.

Tool Name	Description
PeView [37]	This tool provides the complete header information of Windows executable files. Every executable file consists of various information including data, code, metadata of a file. In malware analysis, PE (Portable Executable) header information is used to classify the malware and benign.
PEid [38]	This software is used to identify whether the malware is obfuscated if yes then which packer tool (like UPX, NSPACK, etc.) is used. Nowadays malware developer uses the anti-analysis tools to conceal the actual code of malware. In order to identify the packed malware, PEid can be used.
CFF Explorer [39]	It gives complete header information and metadata of the executable file. This tool displays the more thoughtful information of executable files.
PsFile [40]	This utility tool displays the information of opened files on the system. It is very useful to analyze whether the computer system is operated remotely. It has the capability to give the information of opened files on the local machine by the remote computer.
Accesschk [41]	Accesschk is used to check the system security level. It provides the information of access rights like that which user or group has access to read, write or execute the registry key, files and global objects, etc.
IOC Finder [42]	IOC (Indicator of Compromise). This tool provides information about the compromised systems. It produces the report which provides information about the system and device usage of particular host and present in MS Word or HTML format. This report is analyzed to know the indication of system security.
Radare [43]	Radare is a complete toolkit which is used for doing reverse engineering. This tool can run on many platforms like Linux, Windows, Android and macOS. Radare can also perform the forensics on the file system.
EMET [44]	Enhanced Mitigation Experience Toolkit (EMET) tool kit is used for hardening the application hardening to mitigate the security risks. The main purpose of this tool kit is to prevent the exploitation of the vulnerabilities of systems and application software.
Yara [45]	Yara tool is used for matching the string in the executable files. In malware analysis, certain string signature can be used for identifying the malware file. Yara tool has this capability it can match the particular string pattern in a binary file. It also supports string matching in other file formats like PDF, DOC, etc.
SSDeep [46]	This tool calculates the fuzzy hash value of the executable which is used to verify the variants of the malware. Unlike simple hash value, the fuzzy hash has more capability to deal with variants of the malware.
Disassemblers [37]	When an advanced static analysis is to be done then disassemblers are used. A very commonly used disassemble is IDA Pro. Also, a new disassembler Ghidra has been developed to perform the reversing engineering. In this process, the executable file is converted into assembly code and then code is analyzed to know the behaviour of malware manually.

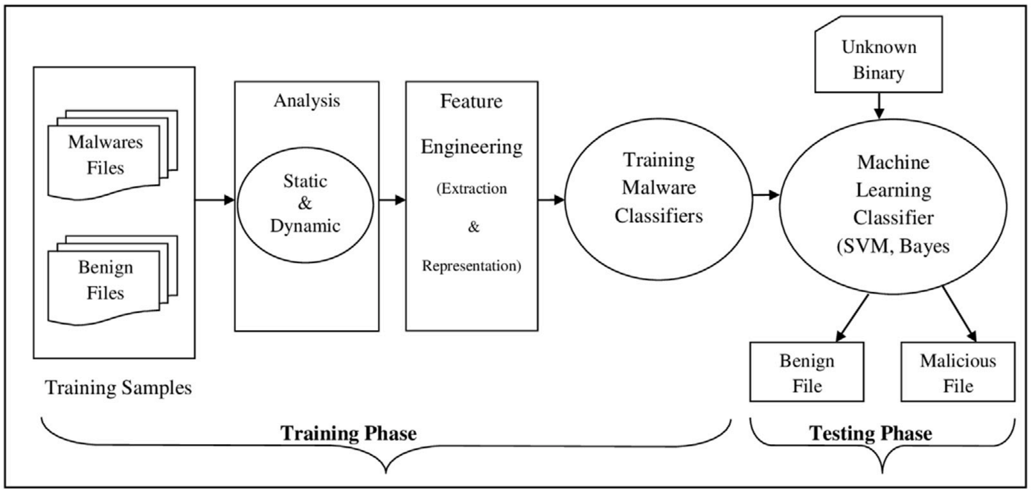

Fig. 3. Schematic Framework of Malware Detection System using Machine Learning.

reason is that it consists of several algorithms which can be applied to the wide variety of malware features set and produces better malware detection results. In addition to the usability of the machine learning algorithm, it has many advantages to detect the malware which are described as follows.

- Existing anti-virus and sandbox techniques can be subverted.
- Automates extracting insight from malware samples.
- Can better generalize at identifying unknown variations.
- It can reduce human effort and time for analyzing the malware.
- Research work has been done by many researchers which shown the significant confidence in the malware detection.

3.1. Challenges to implement ML in malware detection

There are three some challenges to apply the machine learning in malware detection which are discusses as follows.

Table 3
Dynamic analysis tools.

Tool name	Description
Process Explorer [9]	Process Explorer is a system monitor and task manager tool. It works like Windows task manager which gives detailed information about the running processes in the system.
ProcMon [56]	ProcMon (Process Monitor) captures the activities performed by running processes. It captures every system calls of file systems operations, registry modification, memory operation, and network activities
Wireshark, Tshark [57]	Wireshark and Tshark are used to analyze network traffics. These tools have the functionality to capture every incoming and outgoing packet from the system to the outer world. Various types of network feature like URLs, port address and data stream are captured which are further investigated to classify the malware files.
TCPdump [58]	It is a command-line utility tool to analyze live network traffic. Tcpdump is a common packet analyzer which displays the TCP/ IP packets that are transmitted to and from the computer system.
TCPview [59]	It is Windows networking software which shows information of all UDP and TCP endpoints on the system.
Regshot [60]	Regshot tool is used to logs the registry changes made by the executed sample. Before and after the execution of the malware sample, the state of the Window registry is saved. Both the states are compared to know what changes are done by executed sample file.
Memoryze [61]	Memoryze is a command based memory forensics tool. It can take the full memory dump. For analyzing the complex malware, memory dump of the system is taken after running the malware. Memory dumps are investigated to extract the various features like running processes, strings and also enumerate the hidden process in case of rootkit malware.
Volatility [62]	Volatility is an advanced framework for analyzing memory dumps. It is written in python can be easily used on multiple OS (Linux, Windows, Mac OS). It can extract advanced features from the memory dump like process, DLL injected libraries, registry keys, network connections, strings, etc.
Redline [63]	Redline is a portable agent which is configured to capture the data automatically for performing the IOC (Indicator of Compromise) analysis. It is a security analysis tool for the various Windows components such as memory, file system, registry keys and network.
inetsim [64]	It is a network simulator which imitates the internet-based services to give virtual internet to the malware to interact with. During dynamic analysis, inetsim provides the virtual network environment to the malware so that malware can work properly. Suppose if malware wants to interact with the remote computer then inetsim provides the response to that query and malware can continue its execution.
FakeDNS [37]	FakeDNS tool is used to give the response to DNS queries. In real scenarios as the DNS queries are resolved, similarly to handle the DNS queries of malware are resolved by FakeDNS. Both tools inetsim and FakeDNS provides the virtual networking services to know the actual behaviour of the malware.
ApateDNS [65]	ApateDNS is an advanced version of FakeDNS with GUI (Graphical User Interface) features. ApateDNS is easy to configure and analysis the DNS responses as compared to FakeDNS tool.
Sandboxes [66]	Various sandboxes like Cuckoo, Anubis, Panda, Limon, Parsa, etc are used for performing automatic malware analysis. These tools consist of inbuilt many analysis tools which are mentioned above. For extracting the features of belongs to different categories, sandboxes are integrated with many tools like in the Cuckoo sandbox, tcpdump is used for network traffic and volatility for memory dump, etc.

Table 4
Comparison of hypervisors for the dynamic analysis.

	Type-1	Type-2
Architecture	Virtualization directly on hardware	Majority of non-privileged code run on hardware Low
Detection techniques	CPU behaviour edge cases low but non-zero timing overhead	Hosted VMs suffer from the same issues as emulators though may be marginally more transparent
Advantages	Minimal overhead close to hardware	Ease of use introspection, control over state
Disadvantages	Less introspection ability still detectable	Designed for compatibility over transparency
Secure	It is more secure.	Less, if there is problem in host which effect the entire OS including hypervisor
Scalability	It provides better scalability.	Less dependent on host OS

Table 5
Concise Comparison of Static and Dynamic Analysis Techniques.

	Static	Dynamic
Approach	Analysis is done without executing the files	Analysis is done by executing the files
Advantages	Fast and Less time-consuming	More robust to deal with obfuscation techniques
Disadvantages	Cannot detect new malware and easily bypass by obfuscation techniques	Slow complex and time-consuming

3.1.1. High computation cost

The first challenge to implement machine learning is training and updating the malware classifiers. Malware detectors need to be updated consistently. Unlike other domain NLP, computer vision where machine learning has been applied successfully but in this domain, we need to retrain the classifiers frequently to detect the new and mutated malware. As the study explains thousands of new malware are developed every day and malware changes their behaviour less than 24. That is why the ML is expensive and complicated as compared to others. So while applying ML in malware detection we have to think differently.

3.1.2. Adversarial machine learning

Adversarial machine learning is a big problem in machine-based malware classifiers. Adversarial machine learning means when malware developer uses the tactics of ML to bypass the malware detector. Also, Kolosnjaji et al. (2016) [72] discussed that with the help of an intelligent evasion attack they can bypass the machine-learning detection system proposed in Raff et al. (2016) [33].

This is an undeniable fact that there is no other way except machine learning to detect the present malware which is highly complex. Also, the pace of malware development is very high. Now the thing is how we tackle these challenges to implement ML in the cybersecurity domain. To reduce the training cost, we can reduce the dimensionality of the dataset because the machine learning algorithms take more time if the dataset consists of more numbers of data features. For that, feature selection and dimensionality reduction algorithms can be applied to select only the most useful and discriminative malware feature. This could lower the training cost. And the second challenge of adversarial ML can be resolved by building the hybrid malware classifiers. In hybrid classifier, both the static and dynamic features can be utilized. After extracting the features, multiple classification algorithms can be trained. Malware detector based on a single ML algorithm can be bypassed however the ensemble malware detector can be more resilient to handle the adversarial machine learning.

In the literature, various machine learning algorithms have been used for training the classifiers are used such as Support Vector Machine (SVM), Naive Bayes (NB), Random Forest (RF), Decision Tree (DT), Artificial Neural Network (ANN), k-Nearest Neighbours (k-NN), Logistic Regression (LR) and ensemble algorithms like Random Forest (RF), Adaptive Boosting (ADA). In Table 6, a brief comparison among the ML classification algorithms is done.

4. Malware detection techniques

The objectives of malware detection methods are to detect and defend the malicious programs that can harm the computer system or network assets. The Input is represented in many ways to detect and classify the malware samples into an adequate family. The relevant information or knowledge about the malicious file is required which represents the actual behaviour of malware file. For that various malware samples are analyzed with static, dynamic or hybrid techniques. After that, the extracted information is represented into the proper format which is further utilized to train the detection system. Ucci et al. (2019) [73] explained various different features which have been used by various researcher to develop the malware detection system such as Byte sequences, Opcodes, Strings, APIs calls, PE Header information, Windows Registry, File system accesses, AV/Sandbox submissions, CPU Registers, Length of functions, Network Activities and Generated Exceptions. Fundamentally, the malware files are analyzed; features are extracted and represented into an intermediate form before inputting to malware detection. This intermediate form plays a vital role in detection. If the extracted information is appropriate to detect the malware only then, the false-positive rate can be decreased. The various proposed malware detection techniques are classified into three main categories which are explained as follows.

4.1. Signature-based malware detection

In this approach, malware are detected based on a particular signature or pattern of the files. It is a conventional method which is very fast to detect known malicious files as compared to other techniques. Mostly, antivirus software are implemented using signature-based techniques. One simple way to create the signature of malware files using a hash algorithm like MD5, SHA1 etc. Signatures of malware files are generated and stored in the database of the detection system for verifying the signature of an unknown file. If the signature of the file is matched then, it is declared as malicious file otherwise benign file. The problem of this technique is that the signature of the file is changed even if a single byte of the file is changed. Thereby for each modified, malware and new malware, a new signature has to be generated. Only then, a malware detector can detect that malware. Supplementary, new signature-based malware detection systems were proposed which uses the other patterns such as program instructions, control flow graph, and mnemonic sequence. Fig. 4 shows the schematic architecture of signature-based techniques.

As shown in Fig. 4 malware samples are analyzed using various tools IDA Pro, Lida, Peview, PeStudio and features are extracted. Extracted features are used to train the malware detector either using traditional or machine learning algorithms. This figure represents the all the possible ways to develop the signature-based malware classifiers using static features. it covers the various tools, static features and approaches to design malware detectors. In this section, all the presented signature-based techniques have been proposed using this architecture.

4.1.1. Signature-based malware detection techniques

In this section, the review of various proposed signature-based approaches is discussed. Karnik et al. (2007) [78] proposed a technique in which sequences of the functions were used for malware detection. The element of a sequence represented the group of the opcodes. And, function sequence represented the signature of the malicious file to detect the variants of the malware. The cosine similarity measure was computed to cope with the obfuscation techniques. However, this technique was not capable to handle the complex obfuscation techniques (equivalent instruction substitution) and packed malware. Bruschi et al. (2007) [79] proposed a graph-based malware classification model. The author claimed that this technique handles some basic obfuscation techniques. The control flow graphs were created produced from the binary files and then, matched with graphs of already known malicious files. This malware detection approach consisted of two algorithms. The first algorithm searches the similarity within both graphs of the binary file B(which is under test) and already known file M(malware). And the second algorithm matched the reduced graphs of B file with known file M. Author tested 78 malware samples that were tested against these two algorithms and gave a false-positive rate of 4.5% and 4.4% of first algorithm and second algorithm respectively. However, this technique cannot handle zero-day malware.

Zhang et al. (2007) [80] proposed a technique in which n-gram byte sequences features were used to detect and classifies the malware. A selection approach extracted the preeminent n-gram bytes which can signify the malware file. Then, a classifier was constructed using a probabilistic neural network technique. Each classifier consisted of a set of decision-rules for detecting the malware. For training three classes of malware were taken from the online VX Heavens database. Moskovitch et al. (2008) [81] proposed an opcode based machine learning approach to detect the unknown malicious files. The operation codes were grouped into the set of 1-byte, 2-byte, 3-byte, 4-byte, 5-byte, and 6-byte features. Four classification algorithms such as Decision Tree (DT), Naive Bayes (NB), Artificial Neural Networks (ANN) and Adaboost were applied over the n-byte feature sets. The author claimed that this technique can predict the maliciousness of the file significantly.

Griffin et al. (2009) [82] proposed the heuristic malware detection technique. This technique generated a sequence of 48-bytes which

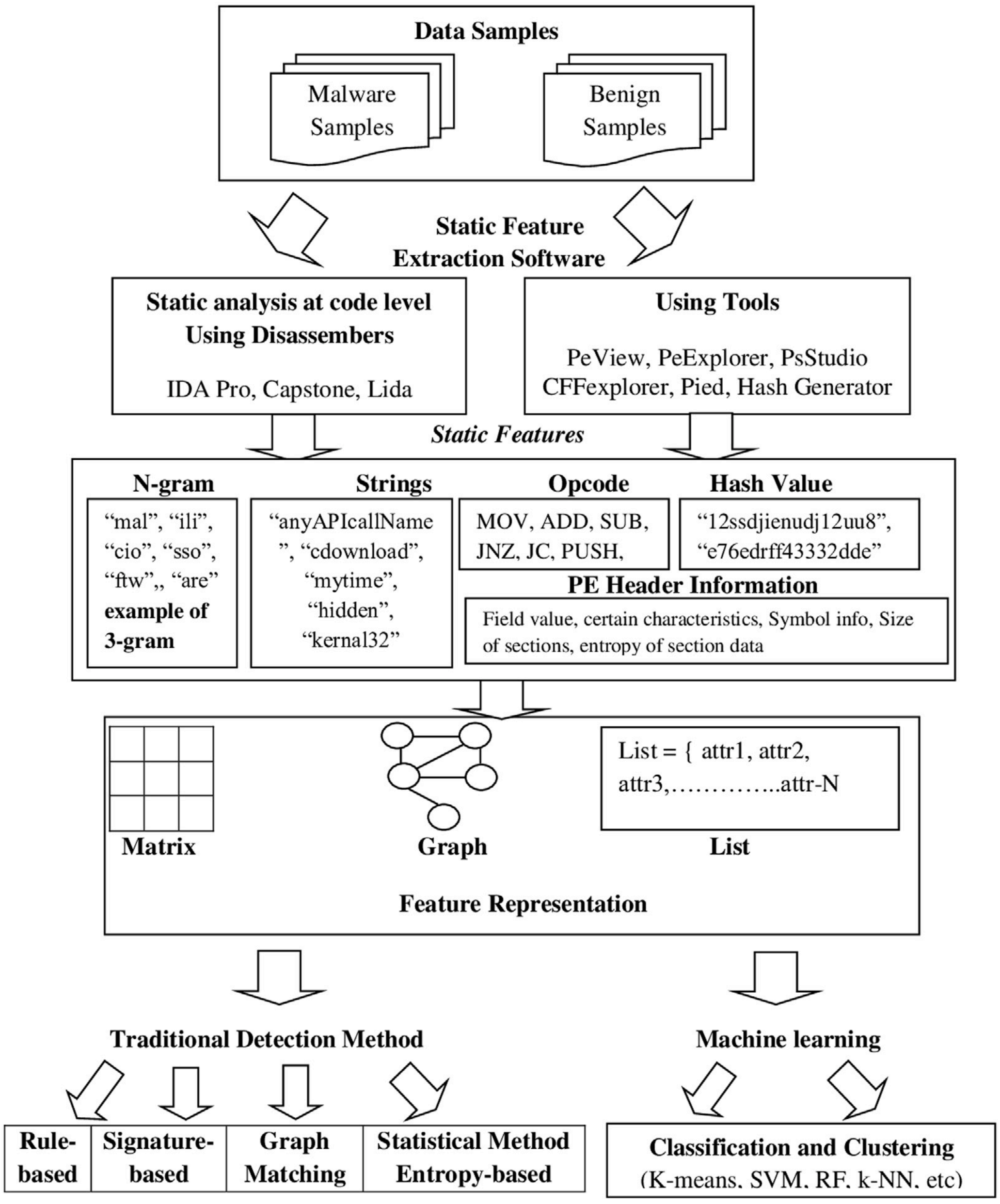
Data Samples
Malware Samples
Benign Samples
Static Feature Extraction Software
Static analysis at code level Using Disassembers
IDA Pro, Capstone, Lida
Using Tools
PeView, PeExplorer, PsStudio CFFexplorer, Pied, Hash Generator
Static Features
N-gram
"mal", "ili", "cio", "sso", "ftw",, "are" example of 3-gram
Strings
"anyAPIcallName", "cdownload", "mytime", "hidden", "kernal32"
Opcode
MOV, ADD, SUB, JNZ, JC, PUSH,
Hash Value
"12ssdjienudj12uu8", "e76edrff43332dde"
PE Header Information
Field value, certain characteristics, Symbol info, Size of sections, entropy of section data
List = { attr1, attr2, attr3,…………..attr-N
Matrix
Graph
List
Feature Representation
Traditional Detection Method
Machine learning
Rule-based
Signature-based
Graph Matching
Statistical Method Entropy-based
Classification and Clustering (K-means, SVM, RF, k-NN, etc)

Fig. 4. Schematic Architecture of signature-based malware detection systems.

served as a string signature to identify the variants of malware. Rather than using a single component signature, the author used the multiple component signatures to train the classifier. It increases the probability of good accuracy result in comparison to a single component signature. However, the impact of multiple components signature during the runtime was not comprehensible. Gavrilut et al. (2009) [83] used different machine learning algorithm cascade one-sided perceptron to differentiate between benign and malware. The objective was to implement this algorithm to reduce the false-positive rate. First, they implemented simple cascade one-sided perception, then, they applied the cascade kernelized one-sided perceptrons algorithm which showed better accuracy (88.78% - COS-P-Poly3) than simple cascade one-sided perceptrons. Beaucamps et al. (2010) [84] proposed a malware detection technique in which execution flow of malicious files was extracted. On the basis of the execution flow, the control flow graphs of malware were created using function calls. After that, graph matching was done to confirm the malware file with the stored templates of CFG of malware. Templates of malware were already stored. If the computed CFG were the part of template CFG then a malware detector classified it as the malicious file. The problem with this technique is that obfuscation techniques like code reordering can evade the malware detector because code reordering techniques change the actual execution paths of the malware.

Cha et al. (2010) [85] introduced the text-based pattern matching method for developing the malware detection system. All the selected sample malware files were scanned with feed-forward bloom filter. It produced the two types of outputs (i) matched malicious files (ii) subset of signature. Subsets of the signature were extracted from the database of signature which was needed to identify the malicious files. After this, verification is done for reducing the false-positive responses of the bloom filters. It addresses the two problems. First, it becomes easy to manage the large database of signatures by reducing them with the subsets of signatures. Secondly, the challenge of memory scaling is managed using a single-bit vector for suspect matching. Text-based detection matching system did not provide significant results that it will not give the high false-positive results. Wang and Wu (2011) [86]

Table 6
Description of the machine learning classification algorithms.

ML Algos	Description	Advantages	Limitations
NB [74]	In the NB approach, the probability of each class and the conditional probability of each data instance with class are calculated. Then, the prediction of a class is done with cumulative probability which is calculated by multiplying the class probability and conditional probability of each contributing data instance. NB can be used for both classifications: binary and multi-class.	Implementation of NB classification algorithm is simple and easy to understand. It can perform well with irrelevant dataset. Also, the classifier can be trained using the small size of the dataset.	The main limitation of NB classifier is that it does not perform well if the data features in trained data are correlated to each other. NB assumes that data elements must be independent.
K-NN [34]	K-Nearest Neighbour (K-NN) classification algorithm classifies the input instance by considering the class label of k nearest training instances. The class of input instance is predicted as of the class of majority instances. Distance measures Euclidean, Manhattan, Hamming and Minkowski are used to find the class label of an input instance from nearest K nearest instances.	Implementation of KNN is very simple and can be updated at a very low cost as new instances with known class labels. KNN algorithm does not make any assumption about the dataset. It is more robust to search space means do not need to be linear separable dataset.	The main drawback of the k-NN algorithm is that it does not perform well if a dataset is unevenly distributed. Also the selecting the appropriate value of k is unpredictable.
SVM [75]	SVM algorithm creates a hyperplane to partitions the data instances of dataset input in different classes. For binary classification, a vector of points on two-dimensional input space can be visualized which separate the input data instance into two different classes benign class and malware class. Application of kernel function in SVM classifier training plays a vital role to classify the classes accurately. Linear, Radial and Poly kernel functions are commonly used in SVM classifiers.	SVM is the most promising Classification algorithm which produces good accuracy in classification. SVM can perform well with high dimensional dataset It can classify the non-linear separable data as well. Selection of the regularization parameter and kernel function varies from problem to problem.	Training time is getting large in SVM with a large value of penalty parameter (C). Also, choosing the value of C is a trade-off between testing and training error.
LR [76]	Logistic Regression is a parametric binary classification algorithm.LR learns the coefficients from the training data to build the logistic regression classifier. In general, LR estimates the empirical value of the parameter in a qualitative response model	Easier to examine and less complex. No need that there must normal distribution of independent variables equal variance. Also, it can handle the nonlinear effects because it does not assume the linear relationship between the independent and dependent variables.	LR has average predictive accuracy is low It cannot handle a lot of inappropriate features well.
DT [77]	In decision tree classification, a decision tree is created by computing the info gain of each attribute in datasets. The attribute has maximum info gain becomes the root. Then other becomes a leaf of the root. Then, the created decision tree is used for making class predictions. In the decision tree each internal (non-leaf) node check the feature, a feature value corresponds to each branch and leaf node represent a class label. Two splitting functions Info gain and Gini index are used to train the model.	DT classifiers can perform well with high dimensional dataset also with noisy data. Unlike KNN and SVM classifiers, it works as a white box. The interpretation of trained can be done. Because of this analysis of the trained model can be elaborated. Training speed of DT classifiers is also fast.	A small change in the dataset cancan cause a large change in the structure of the decision tree which causes instability of the model. It does not perform well with the small number of data features.
ANN [70]	Artificial Neural Networks are the process models which mimics the human brains working to facilitate for finding the decision boundaries by minimizing error rate	ANN can model the non-linear dataset of large number of input features. The ANN can be used almost every kind of problem specially for the optimal problem	ANN might be lead to over-fitting. The weights established for the training data may not be generalize the other datasets, even from the same populations. Computationally Expensive
RF [3]	Random Forest is an ensemble bagging machine learning algorithm. In DT only a single decision tree is created but in RF multiple decision trees are created based on independent subsets of the dataset with replacement. The outcome of random forest is computed through the votes given by every individual tree.	RF machine learning algorithm is immune to variation in the dataset because selects the multiple subsets randomly which reduce the risk of overfitting that why It gives the best classification results. Unlike the decision tree, it can provide good results if there is a change in the dataset.	Training speed is slow. It is directly dependent on the no. of classifiers which are trained to build the strong classifier. It is not transparent as decision tree because of the number of decision trees are computed to builds a strong modal. That is why it becomes difficult to interpret the results.
Boosting Algorithms [66]	Another ensemble machine learning approach is the boosting algorithms which train the multiple weak classifiers in a sequential manner. The weak classifiers are trained on the subsets of data in linear order without replacement. Later then a strong classifier is built by taking the average of all weak classifiers. Boosting is another type of ensemble machine learning algorithms. Boosting is a little bit different than bagging. In bagging, every time training subset is selected randomly from the whole dataset but in boosting training subsets are selected from not selected data features. The Gradient Boosting (GB) and Adaptive Boosting (AdaBoost) are two commonly used boosting algorithms.	Boosting algorithms perform well every type of dataset. Like the random forest, boosting algorithms also handle the variance of data features. Boosting algorithms provide the extra option that different simple machine learning classification algorithms (SVM, KNN, DT) can be used to train the week classifiers before building the final classifier.	Boosting algorithms are time and computationally expensive. To implement the boosting algorithms in real-time platform is hard. In the case of malware analysis, to train the classifier with millions of malware sample will take huge time and computation.

proposed a Packing Detection Framework (PDF) to deal with the obfuscated malware. The main idea of this technique is to design a system which can detect packed malware. Malware samples were analyzed using static analysis for gathering the executable attributes which are further refined. Then, a malware detector was trained using two-class support vector machine algorithm.

Veeramani and Rai (2012) [31] developed a malware detector framework using relevant API calls. In this technique, IDA pro disassembler was used to extract the API calls of malware. To analyze packed malware, unpacking tools were used before disassembling the malware. Both the files of window system32 and malware files are analyzed statically to extract the relevant API of both classes. Then, the support vector machine classifier was trained with the extracted API calls. Even though various unpacking tools are available to unpack packed malware, deobfuscate every malware is very hard. Also, to extract the API calls of obfuscated malware is a more challenging task. Therefore there were more possibilities of the high false-positive response. Shabtai et al. (2012) [87] extended the proposed work of Moskovitch et al. (2009) [81] which was developed using feature sets of opcode patterns. Malware samples were inspected and opcode n-gram sequences were made of four different sizes 50, 100, 150 and 200. Then, the eight types of classifier were trained. Logistic Regression, Artificial Neural Networks (ANN), Naive Bayes, Decision Trees, Random Forest, Support Vector Machine, Boosted Decision Tree and Boosted Naive Bayes was trained with opcode pattern. The evaluated results gave good accuracy more than 96% and 0.1% false-positive rate. Elhadi et al. (2013) [88] proposed the API graph-based malware detection system. In this technique, a data-dependent API call graph was created of each malware file. In this manner, the database of malware was prepared which consists of data-dependent API graphs. In the testing phase, the Longest Common Subsequence (LCS) algorithm was used to match the similarity of the unknown file with the stored graph of malware. This technique produced the 98% detection rate however the experiment was conducted only with 85 malware samples. Also, no methods were discussed to handle the computational complexity of graph generation and graph matching.

Markel and Bilzor (2014) [74] proposed malware detection technique especially for window portable executable file based on PE header information. This technique trained the model on the basis of metadata of the file. The experimental results showed that the metadata of the executable can be used to discriminative between benign and malware software. Three machine learning algorithms were employed on the generated feature of the portable executable header. Decision Tree classifier outperformed the logistic regression and NB classifier. Pechaz et al. (2014) [89] combined the n-gram model and statistical analysis with machine learning algorithm for malware detection. The purpose was for making the collaborated model to detect the new malware like metamorphic which are hard to detect using only the statistical approach to the n-gram model. For selecting the features, the Markov blanket method was used. It was used because it reduces the size of features. Then, the Hidden Markov Model (HMM) was trained on the produced feature sequences which gave the 90% accuracy.

Srndic et al. (2016) [90] presented a paper on static analysis with machine learning techniques to classify the malware samples. In this paper, two types of files format were processed. Nowadays, PDF and SWF files are exploited by the malware developer to embed the executable scripts to harm the computer resources. In this paper, 40,000 SWF and 440,000 PDF were tested. This technique provided the framework to detect the embedded malicious program into PDF and SWF file. Kim et al. (2016) [91] proposed a PE header-based malware detection technique using the machine learning algorithm. The main aim of this technique is to improve the detection rate as compared to previous malware detection techniques which had been designed by using PE header information. Distinguishing features of the portable executable header (PE header information) were extracted from both malicious and benign files. After that, three machine learning algorithm Stochastic Gradient Descent (SGD), SVM and Classification and Regression Tree (CART) were applied over the extracted feature. Near about 27 thousand malware and 11 thousand benign files were used to test the technique. It showed 99% accuracy and 0.2% false-positive rate. However, if the original PE header of the file is obfuscated then this technique will not perform well.

Narra et al. (2016) [92] proposed a clustering malware detection technique which concluded that it can also perform well as compared to SVM. In their previous study, they just implemented the clustering algorithm such as k-means, expectation–maximization with HMM. In this paper, both clustering algorithms were trained without HMM using 7800 malware samples and results were compared with SVM classifier which was also trained on the same dataset. Author concluded that the clustering algorithm performs well like SVM. But the problem in the clustering technique to take the appropriate threshold value of the k in k-means algorithm. It was taken up to 15 then, it produced good results. It is hard to decide that many clusters were there in malware dataset which leads the classifier to hit and trial solution. Raff et al. (2016) [33] presented a byte n-gram based technique and explored the flaws of existing n-gram proposed detection techniques which have been implemented using n-gram feature. For model training, the features were selected using Elastic Net Regularized logistic Regression and analyzed the multi-byte identifiers. After this process, three major flaws were discovered in previous n-gram based techniques. The first issue is how the previous corpora were formed that leads the overestimation of detection accuracy, the second issue was that most of the n-gram was extracted from strings features only and third, n-gram feature leads to the over-fitting of the classifier.

Searles et al. (2017) [93] extended the conventional control flow graph based malware detection technique. In this technique, the Shortest Path Graph Kernel (SPGK) was used to discover similarities between the CFG extracted from binary files. Then, the SVM algorithm was employed with a similarity matrix to train the classifier more accurately. Different parallelization technique was also evaluated to reduce the computational cost or to speed up the classifier. It showed greater accuracy as compared to the 2-gram and 3-gram model on 22,000 binary files. However, to deal with large size CFG is very difficult. In this technique, no such method has been introduced to cope with that. Burnap et al. (2017) [2] used various features such as CPU usage, Swap usage, and network traffic for malware detection. This technique provides a solution to detect the APTs (Advanced Persistent Threats) malware. The author also claimed that it can detect the obfuscated malware as well. The classifier was implemented using Self Organizing Feature Map for reducing the overfitting problem and showed the good results 7% to 25% better than former techniques. Nagano and Uda (2017) [34] proposed malware detection technique in which execution files were analyzed using static analysis tools and extracted the features like DLL import, hex dump and assembly code. With these features, the paragraph vectors were created over which SVM and k-NN algorithms were trained. The experiment was done with 3600 malware samples and produced the 99% detection accuracy. However, simple obfuscated techniques can evade the proposed technique. Nikolopoulos and Polenakis (2017) [94] extended the concept of the graph-based malware detection system. In the existing techniques the graph used to build by considering each system call independently; means each node represent the single system call but in this technique, each node of the graph represents the group of system calls of similar types to build the System Call Dependency graph (ScD-graph). This technique can also detect the mutated malware (like oligomorphic, metamorphic) by applying the classification on the weighted directed graph which is called group relation graph. The main issues of graph-based detection techniques are the similarity matching between the graphs. For handling this problem, the SaMe-similarity and NP-similarity metrics were proposed. 2631 malware samples were used for evaluating the proposed model which consists of 48 malware families. With this dataset, this proposed model produced 83.42% detection rate.

Le et al. (2018) [95] presented a malware detection technique by converting the whole binary file into a greyscale image. Then, the Convolutional Neural Network (CNN) algorithm was used for training the malware classifier. The proposed technique was evaluated using 10568 binary files and produced an accuracy of 98.8%. Wadkar et al. (2020) [96] developed the SVM classifiers by using the various feature of the PE Header of window executables. They extracted 54 static features of the PE header. The SVM classifiers were trained using a large dataset of 500,000 malware samples which were collected from the Vxheaven and Virusshare repositories. The main limitation of this technique was that they did not discuss the accuracy of the proposed model. Also, there is no clear discussion for getting static PE features of obfuscated malware. Table 7 shows the Comprehensive Analysis of reviewed signature-based techniques.

After reviewing signature-based malware classifiers, the comparison of proposed techniques is discussed in Table 7. Table 7 shows various malware features, classification approaches and performances measures used to develop the proposed techniques. The advantages of signature-based techniques are less overhead and execution time for the implementation of these detection systems in real-time scenarios. Some proposed techniques claimed accuracy of more than 99/% in malware detection. These proposed techniques utilized a wide range of malware features and proved their significance for classifying malware and benign files. Besides malware features, the interpretation and representation approaches play a very vital role in malware detection as we have seen Burnap et al. (2017) [2] and Huda et al. (2016) [98] used API calls but applied different machine learning and representation approaches and produced better detection rate. But, there are some limitations of signature-based techniques; it is not possible to detect the new and obfuscated malware. Even we consider that these techniques especially heuristic signatures based can detect unknown malware but to detect the obfuscated malware is not possible. That is because they can overrun the malware detector which is just implemented using static features of known malware. Definitely with the inclusion of machine learning in signature-based techniques, the accuracy in malware detection and prediction has been increased than traditional rule-based malware detection techniques. Like Raff et al. (2016) [33] Nagano et al. (2017) [34]and Shabtai et al. (2012) [87] produced better accuracy than Wang et al. (2016) [97] and Pechaz et al. (2014) [89] with the application of Machine learning algorithms.

4.2. Behaviour-based malware detection

In the behaviour-based approach, the detection of malware is done on the basis of malicious activities performed by malware during execution. In sense of the behaviour is defined by various features such as APIs, browser events, systems events, network events, etc. [56,76,99]. In behaviour-based approaches, we classify these parameters into three main categories such as file activities, registry activities, and network activities. Fig. 5 shows the detailed description of the process to build the behaviour-based malware detection systems.

This malware detection approach can identify the new malware file as well because malware files share the malicious activities in somewhat manner. Therefore, similar behaviour is used to train the malware detection system for detecting the new malware or variant of known malware. Additionally, the behaviour-based approach also provides a solution to handle obfuscated malware. The obfuscations techniques used by malware developers can evade signature-based techniques. The behaviour-based techniques are trained in two ways anomaly means a malfunction which is performed by malicious files. When a file exhibit the abnormal behaviour other than the stored behaviour of normal files then that file is declared as the malicious file. The logic behind to detect the malware is the anomaly (unusual activities performed by malware). In the anomaly-based detection system, the malware detector is trained only by examining the benign files. The benign files are analyzed using static analysis or dynamic analysis. And classifier is trained with the normal activates of the benign files. In anomaly and benign-based, malware files are also analyzed along with the benign files. It is the better approach than the anomaly-based approach to differentiate between benign and malware because both the normal and malicious activities are captured. However, it consumes more time to train the detector as compared to anomaly-based approach. The heuristics techniques are the extended version of behaviour-based malware detection techniques. In contrast to applying traditional malware detection methods, machine learning is much more significant for detecting complex malware as well.

4.2.1. Behaviour-based malware detection techniques

In this section, the proposed behaviour-based malware detections techniques are discussed. Bailey et al. (2007) [100] proposed malware detection technique in which the interaction of malware with system services was captured which consists of system API calls, specific addresses, and functions. The behaviour of malware files also belongs to many categories. Like, a malware file can be a virus, Trojan horse, spyware or worms etc. Further, using that extracted information, malware is clustered into different classes and subclasses. Trinius et al. (2011) [101] described a new representation Malware Instruction Set (MIST) to represent the behaviour of malware. Malware samples were analyzed using CWSandbox. Sandbox generates XML report which is further converted into MIST representation. It increased the efficiency of malware analysis using machine learning and data mining techniques and the size of reports were reduced appreciably. Rieck et al. (2011) [102] proposed the automatic malware detection framework that can detect the variants of the malware classes. This framework consists of two phases. The first phase employs the clustering algorithm to make the classes of malware having similar behaviour. In the second phase, the unknown malware file is classified or assigned to the discovered classes. MIST representation was used to speed up the clustering and classification process.

Hegedus et al. (2011) [103] applied two classifiers k-nearest neighbour and random forest on the behavioural features for malware detection. The proposed technique works in two stages. In the first stage, the random projection is applied for reducing the dimensionality of variable space which also decreases the computational time. Jaccard index was used for measuring the similarity between the features of malware. Then, in the second stage, K-nearest neighbour classifier is used with VirusTotal to detect and label the malware samples. Vasilescu et al. (2014) [104] used cuckoo sandbox for getting dynamic analysis features. The generated report includes system API calls, log entries, portable execution information of binaries etc. Using these extracted features, the malware detector was trained to detect Zero-day malware. Elhadi et al. (2014) [105] designed malware detection technique based on the data-dependent API graph. The graph node represents the API calls along with system resources to increase the accuracy of malware detection. Also to reduce the graph matching complexity an improved graph edit distance algorithm was used which is based on a greedy approach. Mohaisen et al. (2015) [76] extracted the behavioural artefacts by running the malicious file in the virtual environment. After that, the malware samples are characterized using the file operations, network activities, registry key changes and memory operations. SVM was used for classification. The logical regression and hierarchical clustering technique were used for grouping the malware into families of same features. The experiment was done on the medium-scale (400 samples) and large scale datasets (115,000 malware samples). The author claimed 98% accuracy in malware classification.

Ghiasi et al. (2015) [106] introduced a new approach in malware recognition using the contents of CPU registers. This technique extracted API calls with dynamic analysis by running binary files in the monitored environment. Then, a similarity distance between two binary files was computed on the basis of their register contents. Four

Table 7
Comprehensive Analysis of Signature-based Malware Detection Techniques.

Author(s)	Input	Data sources / No. of samples	Results
Wang and Wu (2011) [86]	Portable Executable (PE) Header	Vxheaven and PCHome Malware Repositories / 1056-Packed Malware and 3789-Unpacked Malware and Benign Files	TPR-94.54%
Veeramani et al. (2012) [31]	API calls	Vxheaven Malware Repository / 214- Malware and 300-Benign files	Accuracy-97%
Gavrilut et al. (2009) [83]	API calls	12817-Malware, 16437-Benign Files	Accuracy-88.78%
Shabtai et al. (2012) [87]	N-gram Opcode sequence	Vxheaven Malware Repository / 7688-Malware and 22735-Benign Files	Accuracy-96% TPR-95%, FPR-0.1%
Elhadi et al. (2013) [88]	Data Dependent API Graph	Vxheaven Malware Repository / 85-Malware Files	Accuracy-98%
Markel et al. (2014) [74]	Portable Executable Header	122799-Malware, 42003-Benign Files	Accuracy-97% (Tree CART) 94.5% (LR)
Pechaz et al. (2014) [89]	N-gram	Vxheaven Malware Repository / 1207-Malware and 194-Benign Files	Accuracy-90%
Srndic et al. (2016) [90]	Strings, API Calls	Virustotal Malware Repository / 440000-PDF and 40000-SWF files	Accuracy-99%(PDF)95%(SWF)
Wang et al.(2016) [97]	Opcode Sequence	Vxheaven Malware Repository / 11665-Malware, 1000-Benign Files	Accuracy-88.75%
Huda et al.(2016) [98]	API calls	Vxheaven Malware Repository and using Honeypot	Accuracy-96.84%
Kim et al. (2016) [91]	Portable Executable Header	Vxheaven Malware Repository / 271095-Malware, 9773-Benign Files	Accuracy-99%
Narra et al.(2016) [92]	Opcode Sequence	VirusShare dataset / 7800-Malware Files	Accuracy-98%
Raff et al. (2016) [33]	Byte N-gram	VirusShare Malware Repository and Open Malware, MS Window / 400000-Malware and Benign Files	Accuracy-97.4%
Liu et al. (2016) [21]	N-gram opcode, Image representation	20000-Malware Files	Accuracy-95.10%
Searles et al. (2017) [93]	Control Flow Graph	22000-Malware Files	19% more accuracy than n-gram model
Nagano and Uda (2017) [34]	DLL import, hexdump and assembly code	MWS 2016 Malware Dataset / 3600-Malware Files	Accuracy-99%
Nikolopoulos and Polenakis (2017) [94]	System Call Dependency graph	2631-Malware Files	Accuracy-83.42%
Le et al. (2018) [95]	Greyscale images	10568-Malware Files	Accuracy-98.8%

machine learning algorithm namely Random Forest, Bayesian Logistic Regression, SMO and J48 were used to train the detection model. Finally, all the classifiers were evaluated using 10-fold cross-validation and achieved promising results. The author claimed efficient time in matching with the existing patterns. Pirscoveanu et al. (2015) [64] used Cuckoo sandbox approach for gathering the behavioural features of the binary files. The classification system was trained with random forest algorithms. Ki et al. (2015) [107] proposed malware detection technique based on API call sequence. It was tested that dynamic analysis is more effective for getting behavioural features of malware. It is considered that in all the malware there are common API calls which perform the malicious task even if the malware belong to different categories. Sequence alignment algorithm was applied which can handle the insertion of redundant or irrelevant code in malware. In the testing phase if the extracted API calls of the unknown file are matched with stored API patterns; then, the file is declared malicious otherwise the file is benign.

Pan et al. (2016) [108] proposed a malware classifier using BPNN model. They designed a system HABO to capture the runtime features. After that, main features were extracted from the report such as foreign memory read, mutexes created, the process created or modified

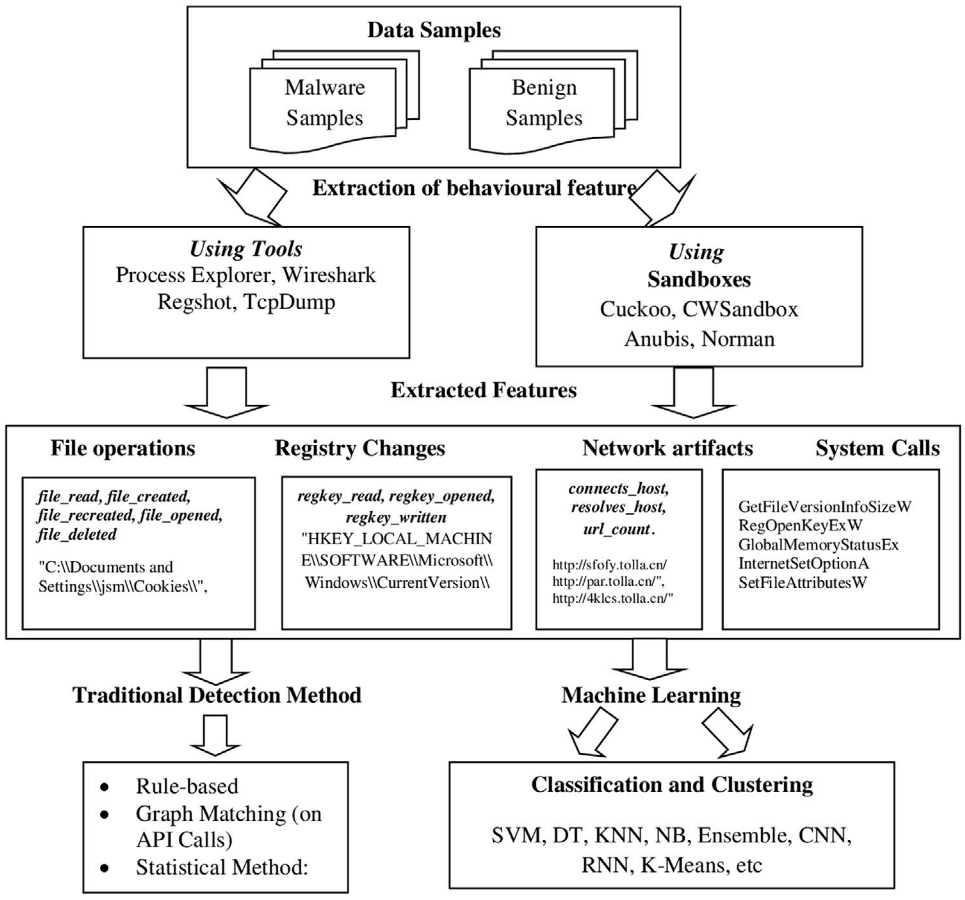
Data Samples
Malware Samples
Benign Samples
Extraction of behavioural feature
Using Tools
Process Explorer, Wireshark
Regshot, TcpDump
Using Sandboxes
Cuckoo, CWSandbox
Anubis, Norman
Extracted Features
File operations
Registry Changes
Network artifacts
System Calls
file_read, file_created, file_recreated, file_opened, file_deleted
"C:\\Documents and Settings\\jsm\\Cookies\\",
regkey_read, regkey_opened, regkey_written
"HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\
connects_host, resolves_host, url_count.
http://sfofy.tolla.cn/
http://par.tolla.cn/",
http://4klcs.tolla.cn/"
GetFileVersionInfoSizeW
RegOpenKeyExW
GlobalMemoryStatusEx
InternetSetOptionA
SetFileAttributesW
Traditional Detection Method
Machine Learning
Rule-based
Graph Matching (on API Calls)
Statistical Method:
Classification and Clustering
SVM, DT, KNN, NB, Ensemble, CNN, RNN, K-Means, etc

Fig. 5. Schematic Architecture of behaviour-based malware detection systems.

registry keys. Back Propagation Neural Network technique was applied to train the model. Narayanan et al. (2016) [109] developed a malware classifier using supervised machine learning algorithms the proposed technique handled the polymorphic malware by representing them in form of images which can capture the minor changes as well while retaining the whole structure. With prepared dataset, three classifiers were trained using KNN, ANN and SVM algorithms. Cho et al. (2016) [110] proposed a technique using the sequence of API calls. For making the API sequences, API calls were extracted by running the malware samples through cuckoo sandbox. The proposed model was trained using 150 samples of 10 families of malware and produced 87% accuracy. Mira et al. (2016) [77] developed an API call based malware detection model which was trained using two algorithms Longest Common Subsequence (LCSS) and Longest Common Substring (LCS).

Bidoki et al. (2017) [56] developed a model by considering the multi-process executing behaviour of malicious files. A malware can divide its functioning into many processes which seem like legitimate operations but their collective response is malicious. Training phase employs the reinforcement learning algorithm and detection phase collects all API calls of each process. Execution policy of all the process was collaborated to find the maliciousness of binary file. Ming et al. (2017) [111] proposed a system call graph-based technique which provides the solution to invisible malware by building the system call based dependence graph. The author suggested that variants of same malware share the same semantic; only the syntax of malware changes. The developed technique was trained with 5200 malware files and validated using 960 malware. The proposed model produced 97.30% precision. Wagner et al. (2017) [112] suggested that malware can be detected by representing the behaviour of malware in visual form. A knowledge assisted visual system was designed to see the visual pattern of the execution of malware. Mao et al. (2017) [113] designed a model which assesses the importance of system objects on the basis of their usability. In this way, a security dependency network is designed which provide insight about the importance of the security of system objects.

Pektas and Acarman (2017) [114] presented a malware classifier which was trained using Online Algorithms (OA) [94,95]. In this approach, various features like registry, network and file system activities were used to build the feature vector. Also, n-gram modelling was done on the API call sequence to represent the behaviour in the feature vector. The model was evaluated with 17,900 malware samples of almost every class of malware like viruses, back-door, Trojan, worms etc. Trained malware classification system showed 92.5% accuracy. Ali et al. (2017) [66] proposed a combined machine learning approach and applied on the large dataset of both malicious and benign files. In this classification technique, three machine learning algorithms SVM, DT and Boosted DT were applied on the extracted features. Boosted DT produced a better result than SVM and DT classifiers. Amazon web services were used to host the introduced cloud-based architecture which provides the scalability. To train the classifier with a dataset as large as 150,000 malware and 87000 benign samples were used. The detection system showed 99% accuracy with boosted DT algorithm.

Ding et al. (2018) [48] presented a solution to design the graph-based malware detector. The author proposed rather than building a behaviour graph for each malware, a common graph can be built for each family of malware. For building the behaviour graph; dynamic taint analysis technique was applied. Maximum weight subgraph parameter was used to match the graph of unknown file with stored graphs of each malware family. Stiborek et al. (2018) [115] designed a technique which captures the behaviour of malware by executing the malware samples in the sandbox. In the sandbox, each malware sample is executed as a set of pairs of names and resources types. This view of problem framing is called a multiple instance learning. Machine learning methods are applied over the variable size of malware samples; a review article on multiple instance learning explained the several techniques to handle the problem of sample size. The vocabulary-based method applied to handle the problem of variability of samples sizes. The proposed technique was trained with a large corpus of 112115 binaries and produced the accuracy of 95.4%.

Ghafir et al. (2018) [116] proposed a technique for detecting Advanced Persistent Threats. The author trained the classifiers using SVM, KNN and ensemble algorithms. This technique produced the 84.8% accuracy in prediction of APT correctly. Alaeiyan et al. (2019) [117] developed a malware detection technique using run-time features. This technique also gave the elucidation for evasive malware by running into Parsa sandbox. The proposed technique was tested using 1100 malware and produced the accuracy of 97.9% for classifying the malware. Xiaofeng et al. (2019) [118] developed an API calls sequence-based malware classifiers for window platform. Using the cuckoo sandbox, dynamic API calls were extracted. After that, two classifiers were trained. First, a random forest classifier and second models were trained using a bidirectional residual neural network. The main objective of this technique to discover the next API call to be called based on the current API call. The author claimed that both models detect malware effectively. But the combined model produced better accuracy of 96.7% better than individual ones. Arivudainambi et al. (2019) [119] presented a network traffic analysis model for classifying the malicious traffic. They incorporated PCA with to address the advanced anti network traffic analysis approaches. The proposed model was implemented by running 1000 malware samples using the various sandboxes Noriben, Cuckoo and Limon. This technique produced an accuracy of 99% in malware detection.

Namavar et al. (2020) [120] modified a Two-hidden layered Extreme Learning Machine (TELM) to increase the malware accuracy as compared to Long Short-Term Memory (LSTM) and CNN models. The author considered the local dependencies of adjacent malware features and global dependencies in the second layer for detection unknown and variants of malware. Proposed models produced an accuracy of 99.65% in the detection of IoT malware samples. Yucel et al. (2020) [121] developed techniques based on memory images of executable files. They took 123 malware samples from different malware families. Malware samples were executed using virtual machines and 3D images of memory were created to find the similarity with known malware. This technique showed that different similarity rates in different malware families like 0.99 for Marina Botnet Family, 0.99 for the Rex Virus Family and the average ratio of 0.886. Rabbani et al. (2020) [122] proposed a Probabilistic Neural Network (PNN) model for detecting malicious behaviour over the network traffic. Various network features related to IP addresses, TCP, UDP, CON protocols, jitters, etc were used as the feature vector. Then, the proposed technique was implemented by combining the Particle Swarm Optimization (PSO) algorithm with PNN algorithms and produced a 96.5% detection rate in malicious behaviour of network traffic.

The comprehensive comparison of reviewed behaviour-based techniques is discussed in Table 8. Nowadays, more research work is done on behaviour-based malware detection techniques. The reason behind it is that the limitation of signature-based technique to cope with new and complex malware. Unlike signature-based techniques, these techniques produced better accuracy results by using runtime features. Elhadi et al. (2014) [105], Pan et al. (2016) [108], Ali et al. (2017) [66] and many others used dynamic API calls, Stiborek et al. (2018) [115], Pektas et al. (2017) [114] and Alaeiyan et al. (2019) [117] used file, registry, network activities for training malware classifiers using supervised classification algorithms. Some proposed techniques like Ghafir et al. (2018) [116] utilized many runtime features for developing the models. The detection rate of behaviour-based techniques is much better than signature-based techniques. Also, these proposed techniques claimed the prediction of new and obfuscated malware. In traditional behavioural techniques, it was a more time-consuming process to extract the run-time feature but by the use of machine learning algorithms, this process has become faster which led the proposed model to use more features and large malware samples for training and testing the malware classifier. But, these proposed techniques have some issues and challenges to implement them. The issues like proposed techniques are validated used different malware samples and evaluated using dissimilar performance measures. Also, the used approaches for training the classifiers are different. The challenges to implement the behaviour-based technique on the real system are high computational overhead and execution time of malware samples before predicting them. To exhibit the advantages of signature-based and behaviour-based, hybrid techniques have been also proposed which are discussed in the next section.

4.3. *Hybrid malware detection techniques*

Both signature and behaviour-based techniques have pros and limitations. To consider that many researcher have proposed hybrid malware detection approaches for taking the advantages of both methods of malware detection using static and dynamic features. In this section, the study of various hybrid malware detection techniques is discussed and compared with many parameters. Rabek et al. (2003) [123] proposed a malware detection technique to detect the obfuscated malicious files. Static analysis was done for getting the information about all the system calls. The extracted information includes the name of all the functions, addresses and also the return address of system calls which were used along with the dynamic features. To capture the runtime activities, malware files were executed in the controlled dynamic environment. If the executable file invokes same system calls which are already stored (which represents the malware file), the file is classified as a malicious file. However, this technique gets fail if the malware developer embeds some irrelevant system calls in the code. Collins et al. (2008) [124] proposed a protocol graph detector to detect the worms in the network. A network was created in which hosts were represented as node and connection as an edge. This technique simulated the network to know the behaviour of worms. It only deals with worms, not with other types of malware such as Trojan horse, virus etc. The study of various hybrid malware detection technique is discussed as follows.

Mangialardo et al. (2015) [125] proposed a FAMA framework for handling the weaknesses of both static and dynamic analysis technique to reduce the false-positive response. Static features were extracted using IDA pro and Cuckoo sandbox was used for capturing the behavioural features. Then, the extracted features were fed to Random Forest and C5.0 algorithms for training the classifier. The experimental results showed 95.75% accuracy for classifying the unknown file as benign or malicious. Shijo et al. (2015) [126] proposed an integrated malware detection approach. The binary files were disassembled and printable string information was extracted. Additionally, it handles the unwanted printable strings which are inserted to obfuscate the code. And, through the dynamic analysis, the binary files were executed with the Cuckoo Sandbox; API system calls related to the file system, registry modification and special process and memory address were extracted. Finally, the malware classifier was trained using the Random Forest and Support Vector Machine.

Edem et al. (2015) [127] proposed an automatic malware detection framework. Author integrated the data mining clustering techniques with malware analysis. Malware samples that exhibit similar behaviour were grouped using the k-means clustering algorithm for making the analysis easier and insightful. Initially, the malware samples were analyzed statically and dynamically. In static analysis, IDA Pro, OllyDbg tools were used for getting the static features and CWSandbox was used. The CWSandbox generates an XML report of the behaviour of the malware samples. Then, both static and behaviours features were processed using the data mining algorithm. Okane et al. (2016) [128] proposed an advanced malware detection technique with the SVM

Table 8
Comprehensive analysis of behaviour-based malware detection techniques.

Author(s)	Input	Data sources / No. of samples	Results
Elhadi et al. (2014) [105]	Data Dependent API Graph	Vxheaven Malware Repository / 416-Malware and 98-Benign Files	Accuracy-98%
Mohaisen et al. (2015) [76]	API calls and Network Activities	Antivirus companies / 115000-Malware Files	Precision –99.5% Recall-99.6%
Ghiasi et al. (2015) [106]	Contents of Registers, API Calls	850-Malware and 300-Benign Files	Accuracy-95.9%
Pirscoveanu et al. (2015) [64]	Window API calls	VirusShare and Virustotal Malware Repositories / 42000-Malware Files	Accuracy-98%
Ki et al. (2015) [107]	API Call Sequence	VirusTotal Malware Repository and Malica Project / 23080-Malware Files	Recall-98.8%F1-Score-99.9%
Pan et al. (2016) [108]	API calls	Kafan Forum / 13600-Malware Files	Accuracy-98%
Nayaranan et al (2016) [109]	Representation of Malware in form of images	Kaggle Microsoft Malware Dataset / 10868-Malware Files	Accuracy-96.6% (Linear KNN)
Cho et al. (2016) [110]	API call sequence	Vxheaven Malware Repository / 150-Malware Files	Accuracy-87%
Mira et al. (2016) [77]	API call sequence	VirusSign, SAMI, CSDMC Malware datasets / 13600-Malware	Accuracy-99%
Mao et al. (2017) [113]	System objects	Vxheaven, MALICA and Virustotal Malware Repository / 7257-Malware Files	Accuracy-93.92%, FPR-0.1%
Bidoki et al. (2017) [56]	API Calls	Vxheaven Malware Repository / 378-Malware and 500-Benign Files	Accuracy-91.66%
Ming et al. (2017) [111]	Dependency Graph	5200-Malware Files	Accuracy-97.30%
Wagner et al. (2017) [112]	Sequence of API calls	Vxheaven Malware Repository / 8847-Malware and 1460-Benign Files	Accuracy-95.25%
Pektas and Acarman (2017) [114]	Registry, Network, File System, API call sequence	VirusShare Malware Repository / 17900-Malware	Accuracy-92.5%
Ali et al. (2017) [66]	Run-time features	Malware Repository of Nettitude / 150000-Malware and 87000-Benign Files	Accuracy-99%
Stiborek et al. (2018) [115]	Behaviour artefacts	112115-Malware Files	Accuracy-95.4%
Alaeiyan et al. (2019) [117]	Behavioural features using Parsa sandbox	VirusShare Malware Repository / 1700-Malware and 1700-Benign Files	Accuracy-97.9%
Xiaofeng et al. (2019) [118]	API call Sequence	VirushShare and VirusTotal Repositories / 1430 Malware and 1352 benign Files	Accuracy-96.7%
Arivudainambi et al (2019) [119]	Network Artefacts	1000-Malware Files	Accuracy-99%
Rabbani et al. (2020) [122]	Network features	22211-Malware and 677789-Benign Files	Accuracy-96.5%
Namavar et al. (2020) [120]	Behavioural features	Vxheaven and Microsoft Kaggle Datasets / 18831-Malware Files	Accuracy-99.65%
Yucel et al. (2020) [121]	Memory Images	Virusign Malware Dataset / 123-Malware Files	Accuracy-99.5%

algorithm for malware classification. This technique takes the program runtime trace as a feature to train the detection system not only API like other dynamic behaviour-based detection techniques. After reducing the extracted features to smaller feature sets using opcode filtering methods, the Support Vector classifier was trained. Nauman et al. (2016) [129] introduced the three-way decision making the concept in the malware detection system. All the previous proposed techniques gave the result in two-way either the binary file is malware or benign. In practically, detector cannot classify all tested files correctly. Some complex malware like Stuxnet or very unique binary files are not classified accurately. In that case, there are more chances of false-positive or negative results from the detectors. Therefore, this proposed techniques deal with such malware by three types of decisions accepted, rejected and deferred. The two probabilistic rough models (i)

Game-theoretic rough sets (ii) information-Theoretic rough test were used to classify the binary files. The drawback of this technique is that it does not provide the solution to handle the deferred malware. Kaur et al. (2016) [22] proposed a hybrid malware detection system which integrates the various static and dynamic analysis tools in a component-based framework. The main goal was to design the hybrid framework is to detect the zero-day malware automatically by utilizing the malicious activities of known malware. The malware detector is trained by extracting the static and dynamic features from the malware samples. In this technique, many static features such as hash value, PE header information, strings and dynamic activities related to the process activities, file operations, memory and network activities were captured for the detection and classification of new malware.

Kolosnjaji et al. (2016) [72] presented an enhanced semi-supervised malware detection technique which combines the results from both dynamic and static malware analysis for increasing the classification performance. It is a different technique from previous malware detection techniques because the different algorithms were applied to process extracted features of static analysis and dynamic analysis. A semi-supervised propagation procedure was used to classify the static results and the statistical topic modelling was applied over the dynamic reports which come across the hidden semantically significant features of malware files. Above all this non-parametric technique provides a dynamic online scheme for the malware categorization. Damodaran et al. (2017) [3] proposed a hybrid technique in which opcode sequence and API calls were used to train the classifier. Like in other hybrid technique both static and dynamic features are extracted from the binary files. Then, the Hidden Markov algorithm was used to train the classifier with static and dynamic data. This technique showed good results only for some families of malware. Also, current obfuscations techniques can evade the malware detection technique static malware analysis process.

Pfeffer et al. (2017) [130] proposed a malware detection framework MAAGI (Malware Analysis and Attributed using Genetic Information). In this framework, the genetic algorithm was applied over the behavioural features of malware. Malware samples are processed in static and dynamic sandboxes for gathering the features of malware. For static analysis, PEid and IDA pro tools were used and for dynamic analysis, the Symmon and Introvirt tools were used. The idea behind the MAAGI framework is that there are solid likenesses between biological behaviour and malware behaviour. Therefore, the artificial intelligence algorithm was applied to make the malware detection framework and yielded promising outcomes. Hopefully, it will encourage more incorporation between the cyberdefence communities and artificial intelligence.

Huda et al. (2017) [75] proposed a semi-supervised machine learning technique which automatically integrates the information about unidentified malware into the detection system from previously labelled and unlabelled data. The uniqueness of this technique is that it automatically updates the database of the detection system without any external efforts. This technique uses the k-means clustering with inverse document frequency, term frequency as a distance measure to extract the information of cluster and support vector machine algorithm to classify the binaries. Huda et al. (2018) [131] proposed a hybrid technique using wrapper complementary filter and feature selection. In this paper, the author chose the maximum and minimum relevant features. After that, different machine learning algorithms MR+SVM, MRED+SVM and Fisher+SVM were used to train the model which produced the accuracy of 99.49% using extracted features static and dynamic features. Table 9 shows the comprehensive analysis of reviewed hybrid malware detection proposed techniques.

Table 9 shows the study of hybrid malware classifiers. Huda et al. (2018) [131] produced the highest accuracy of 99.49% in malware detection. These techniques were proposed to bridge the gap between static and dynamic techniques by inheriting the advantages of both malware detection approaches.

5. Discussion and analysis of proposed malware detection techniques

In this section, a statistical analysis of detection techniques using ML is discussed. Since three types of malware detection techniques are discussed. There are many ways to classify the malware detection techniques based on the analysis methods, traditional or machine learning-based techniques, deep learning-based, PCs and mobile malware-based techniques. In this paper, we studied malware detection techniques which have been designed for PC malware. Table 10 shows the studies of malware detection techniques based used machine learning algorithms with input features. It shows which algorithms is used most and have been used in present time with static and dynamic features as well.

- SA — Static Analysis
- DA — Dynamic Analysis
- MN — Mnemonic Frequency, Opcodes
- AC — API Calls, Imports, DLL Import
- IM — Image Representation of binary file
- RC — Register Content
- PEH — Portable Executable Header
- PSI — Printable String Information
- RTF — Runtime features (FILE, Network, Registry)

Besides the analysis type and machine learning algorithms, the other major thing in malware detection is the type of malware features. Malware features can be hash value, particular string information, opcodes, n-bytes, file system, registry key changes, process operations, network activities and other system information. As shown in Table 10, some authors like Searles et al. (2017) [93], Wagner et al. (2017) [112] and Wang et al. (2016) [97] prepared feature sets using a single feature while many researchers like Shijo et al. (2015) [126], Srndic et al. (2016) [90], Ali et al. (2017) [66] and Huda et al. (2018) [131] used multiple features for training malware classifiers. Furthermore, the processing and representation of features make a difference in the performance of malware detection. As we have seen many techniques used the API calls, runtime features, opcodes, n-grams, etc; however their processing and application of different methods produced better results. Fig. 6 shows a comparison between the accuracy of static malware detection techniques. Searles et al. (2017) [93], Kim et al. (2017) [91] and Nagano and Uda (2017) [34] produced the maximum accuracy of 99% accuracy using SVM classifiers.

Fig. 7 shows the accuracies of dynamic malware classifiers. Namavar et al. (2020) [120], Yucel et al. (2020) [121] and Ali et al. (2019) [66] produced classification results with more than 99% accuracy. Also, Fig. 8 illustrates accuracy analysis of hybrid malware techniques in which Huda et al. (2018) [131] produced the highest accuracy of 99. 49% using MR + SVM and MRED + SVM algorithms.

From reviewed literature, it can be inferred that both static and dynamic techniques using machine learning can develop a more accurate malware detection system. Also, a particular classification algorithm is not appropriate for developing a model using different malware features. However, in static analysis SVM performed better than others. In the case of dynamic analysis ensemble algorithms also performed well. After studying different malware detection techniques, some main research issues are observed. Each system has advantages and limitations. Like, Signature-based malware detection approaches can only detect the known malware whose signature is already stored in the database of the detector. Obfuscation techniques can easily evade signature-based malware detection systems. The behaviour-based malware detection system can solve the problems of signature-based techniques. However, behaviour-based techniques are more time-consuming as compared to signature-based and also have a high false-positive rate. Malware detection is a continuous process. Day by day it is getting harder. As computer users are increasing and more attackers are developing very complex malware which is very hard to detect. Along with the

Table 9
Comprehensive Analysis of Hybrid Malware Detection Techniques.

Author(s)	Input	Data sources / No. of samples	Results
Mangialardo et al. (2015) [125]	API call sequences	VirusShare Malware Dataset / 131073-Malware Files	Accuracy-95.75%
Shijo et al. (2015) [126]	Printable strings information (PSI) and API calls	997-Malware and 490-Benign Files	Accuracy-98.7% (SVM) , 97.68% (RF)
Edem et al. (2015) [127]	Signature and Window API calls	Malware Sample taken from MWanalysis.org / 1143-Malware Files / 1143-Malware	
Okane et al. (2015) [128]	System API calls	Vxheaven Malware repository / 350-Malware, 300-Benign	Accuracy-86.3%
Nauman et al. (2016) [129]	System Calls	UNM Application Dataset	Accuracy-92.51%
Kolosnjaji et al. (2016) [72]	PE Header Information and API Calls	Virustotal Dataset / 2000-Labelled Malware and 15000-Unlabelled Malware Files	Precision-90%
Damodaran et al. (2017) [3]	PE information, API Calls	Vxheaven Dataset / 745-Malware and 40-Benign Files	Accuracy-98%
Pfeffer et al. (2017) [130]	API calls	MIT Lincoln Lab Malware Dataset / 8336-Malware and 128-Benign Files	Accuracy-86%
Huda et al. (2017) [75]	Printable Strings, Imports, Procedure Call Graph (PCG)	CA Technologies VET Zoo Malware Dataset / 967-Malware Files	Accuracy-93.83% , FPR-0.144%
Huda et al. (2018) [131]	Run-time activities	Malware sample from CA Technologies VET, ZOO, Offensivecomputing.net and Vxheaven Repositories / 2000-Malware and 1500-Benign Files	Accuracy 99.49%

Table 10
Analysis of studied malware detection proposed techniques on basis of three analysis type, ML algorithms and malware features.

Authors	Types		ML Algorithms								Features						
	SA	DA	DT	SVM	KNN	NB	LR	ANN	RF	ADA	MN	PEH	AC	PSI	RC	IM	RTF
Ali et al. (2017) [66]		o	o	o						o			o				o
Gavrilut et al. (2009) [83]	o							o					o				
Ghafir et al.(2018) [116]		o		o	o				o	o							o
Ghiasi et al. (2015) [106]		o											o		o		
Huda et al. (2017) [75]	o	o	o	o		o			o				o	o			
Huda et al. (2018) [131]	o	o		o													o
Huda et al.(2016) [98]	o			o									o				
Ki et al. (2015) [107]		o											o				
Kim et al. (2016) [91]	o		o	o								o					
Kolosnjaji et al. (2016) [72]	o	o							o	o		o	o				
Le et al. (2018) [95]	o							o								o	
Liu et al. (2016) [21]	o								o		o		o			o	
Mangialardo et al. (2015) [125]	o	o	o						o				o				
Mao et al. (2017) [113]		o															o
Markel et al. (2014) [74]	o		o			o	o					o					
Mohaisen et al. (2015) [76]		o	o	o	o		o						o				o
Nagano and Uda (2017) [34]	o			o	o						o		o				
Narra et al.(2016) [92]	o			o							o						
Nauman et al. (2016) [129]	o	o											o				
Nayaranan et al (2016) [109]		o		o	o			o								o	
Okane et al. (2015) [128]	o	o		o									o				
Pan et al. (2016) [108]		o						o					o				
Pfeffer et al. (2017) [130]	o	o											o				
Pirscoveanu et al. (2015) [64]		o							o				o				
Raff et al. (2016) [33]	o						o				o						
Searles et al. (2017) [93]	o			o													
Shabtai et al. (2012) [87]	o		o	o	o	o	o	o	o		o						
Shijo et al. (2015) [126]	o	o		o					o				o	o			
Srndic et al. (2016) [90]	o			o									o	o			
Stiborek et al. (2018) [115]		o		o					o								o
Veeramani et al. (2012) [31]	o			o									o				
Wagner et al. (2017) [112]		o			o								o				
Wang et al. (2016) [97]	o										o						

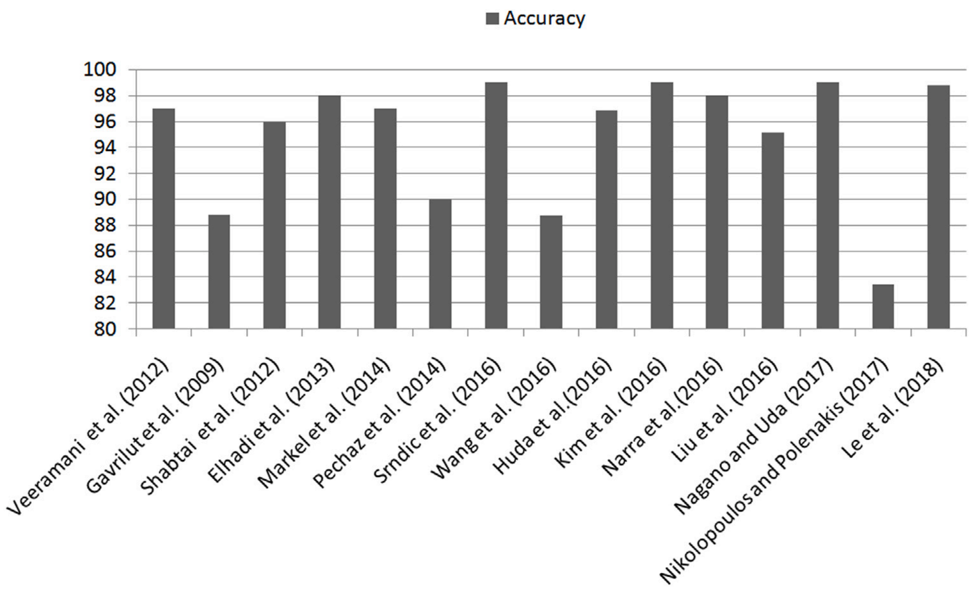

Fig. 6. Accuracy Comparison Graph of Static Malware Detection Techniques.

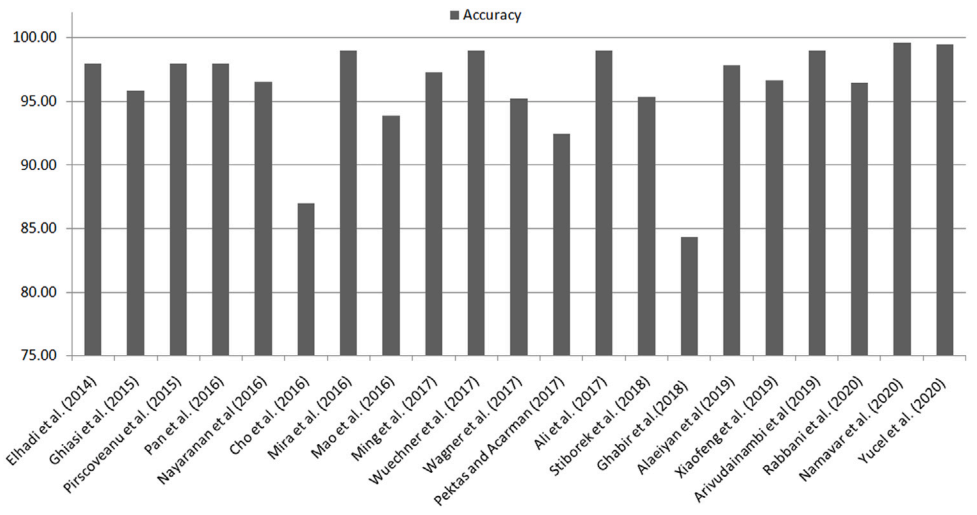

Fig. 7. Accuracy Comparison Graph of Dynamic Malware Detection Techniques.

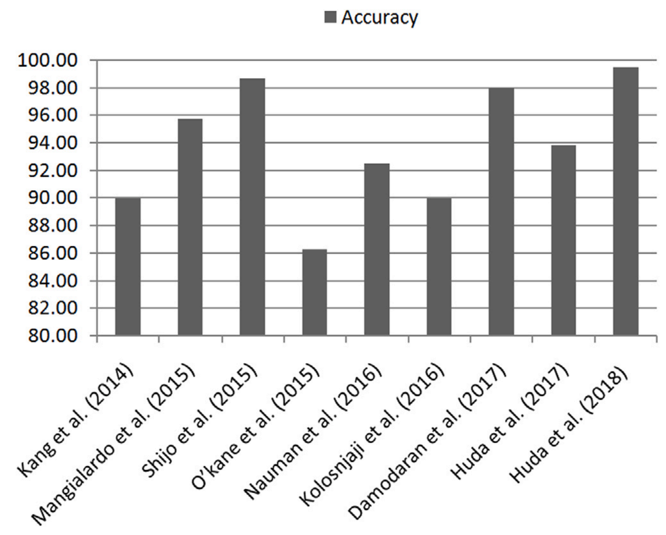

Fig. 8. Accuracy Comparison Graph of Hybrid Malware Detection Techniques.

complexity of malware, the pace at which malware are being developed is also a colossal problem to stop malware attacks. Therefore the battle between malware developer and security analyzer is never-ending as improvement grows. In the next section, we have discussed a framework to address these problems of malware detection. It would not be a steel bullet but this approach includes the maximum dimensions for malware detection.

6. Important factors for developing the malware detection systems

In this section, some important factors of malware development are discussed which provides intuition to the researcher to do research this problem domain.

6.1. Handling anti-analysis techniques

The problem comes when malware is developed using the anti-static or anti-dynamic analysis technique. Malware detects the analysis environment in dynamic analysis and hide actual behaviour or stop executing in that environment. Therefore while developing the malware analysis system; analysts must have to handle this issue. Malware analyst needs to develop the unpacking software in the case of static analysis and setup dynamic environment by dealing with all those patterns of monitoring virtual machines which are used by malware to identify the analysis environment. Some tips for setting the dynamic malware analysis to run malware samples.

- The default MAC address of virtual machines can be changed to avoid the detection of VM machines by malware using known and standard MAC addresses of VMs.
- Keep the name of VM machines which seem to be the host systems like Jagsir-pc, John, etc.
- Always install all basic application software like MS Office, Adobe Reader, VLC, etc to feel like a personal computer.
- Add documents in various directories like Documents, Desktop, Downloads, temp, etc. Also, use the VM machine for many days that malware cannot find it that this is a new machine prepared for malware analysis. After using VM as personal work then, the documents are created and cache or other temporary files are created while using the internet.

6.2. Analysis tools and environment setup

Besides the anti-analysis techniques, used tools and type of environment setup specially in dynamic analysis have a large impact on the accuracy of malware classifiers. In the case of static analysis, we have seen many static feature extractor tools like IDA Pro disassembler, Capstone, Peid, PsStudio, etc were used. Static analysis is performed without disassembling the file and by disassembling the malware files. Using disassemblers, more detailed features are extracted which are not possible using simple tools like CFF Explorer, Psfile, IOCFinder, etc. In the case of dynamic analysis, the type of architecture of the analysis environment makes a difference in detection systems. The type-1 hypervisor is more robust as compared to type-2 to handle the anti-analysis techniques. However, both have advantages and limitations which are discussed in Section 2. Therefore, during the development of anti-malware system, the developer has to select the capable tools to extract the essential malware feature.

6.3. Selection of proper data samples for training the malware classifiers

In studied literature, malware samples are collected from various repositories. Some repositories like VirusShare provide access to the researcher for downloading the million malware samples including the latest ones. The most commonly used malware samples were collected from Vxheaven which provides labelled malware samples. However, after 2010 this malware repository is not updated. So from the perspective of malware detector malware samples must-have new malware samples. As we have found some research used Kaggle, Microsoft, malwar, contagio, theZoo, etc. repositories are used for training malware samples. But the main problem in that there is no specific database of labelled malware samples. While there are some very widely used online platforms like VirusTotal, Hybrid Malware Analysis which can label the malware samples. However, it is a time-consuming process labelling the malware sample one by one. Additionally to include malware samples of different categories in equal fraction for the training is a big challenge. There is another alternative for getting live malware is the honeypot. Some proposed techniques utilized honeypot for getting real-time malware payload but most of the research paper misses this. Except all, we can also get malware samples from antivirus organization for validating our proposed model. This would be a great option to test malware classifiers with realistic malware samples.

6.4. Selection of malware features

Scalability is a big challenge to implement the proposed model. It directly depends on how many features are used to classify benign and malware files. As the features are increased accuracy increases but take longer to complete the scanning which limits the real-time application of malware detection system. In addition to the cardinality of malware features, the selection of malware features plays primary functions which are more discriminative to differentiate between malware and benign files. In literature, many research used API calls, some used PE header and some used file, network file activities.

6.5. Selection of machine learning algorithms

After an exhaustive survey of many techniques, it has seen that all types of machine learning algorithms have been used. Ensemble produces better accuracy results than simple (SVM, DT, KNN) classification algorithms however these algorithms take more training time as compared to simple algorithms. As per the size and variation in malware features, the machine learning classification algorithms can be selected. Another essential factor in the accuracy of malware classifier is the parameterization of used algorithms. In the reviewed proposed technique, this concept is not covered in detail. It is a major factor behind the accuracy of the malware classifier.

7. Future directive

Now the question is how to develop a malware classifier which can cope with these issues. As we know continuously, malware analysts analyze the malware samples and keep updating the malware detection system to stop the malware attacks. Presently, only signature-based techniques are almost helpless to detect new malware. Therefore another approach is behaviour-based malware detection which gives hope for handling new malware. So we will develop a hybrid framework, not like previously proposed hybrid techniques. This proposed technique will be implemented using two-layered architecture. At the first level, signature-based malware detection will be done, if it does not succeed than performs the second level using the behaviour-based analysis techniques. The first layer can easily detect known and simple unobfuscated malware while dynamic analysis can predict the unknown malware using the run-time feature of that. And at the occurrence of each new malware, the database will be updated which can be further used to predict future malware. Fig. 9 shows the framework of the malware detection system.

In this technique, the dynamic analysis will be performed using both automatic sandboxing (like Cuckoo sandbox) and various dynamic analysis tools like Ollydbg, Regshot, Wireshark and ProcMon for gathering runtime features. And, the static analysis will be done using tools like PExplorer, Peview, PeId and IDA pro to extract static features like strings, imports, exports, etc. Then, malware classifiers will be trained using machine learning algorithms. This technique will inherit advantages of signature-based and behaviour-based technique by detecting known malware efficiently and detecting unknown malware using runtime behaviour, In addition to this, during the static and dynamic analysis, we will apply anti-obfuscation techniques for analyzing malware samples properly. It is merely achievable after analyzing malware and benign samples using static and dynamic malware analysis techniques.

The aim of this proposed technique to bridge the gap between signature-based and behaviour-based techniques. The development of the two-layered model of hybrid technique will provide the real-time implementation of the malware detection system. Also, with the application of multiple ML algorithms will give more resilience to the hybrid model against adversarial machine learning.

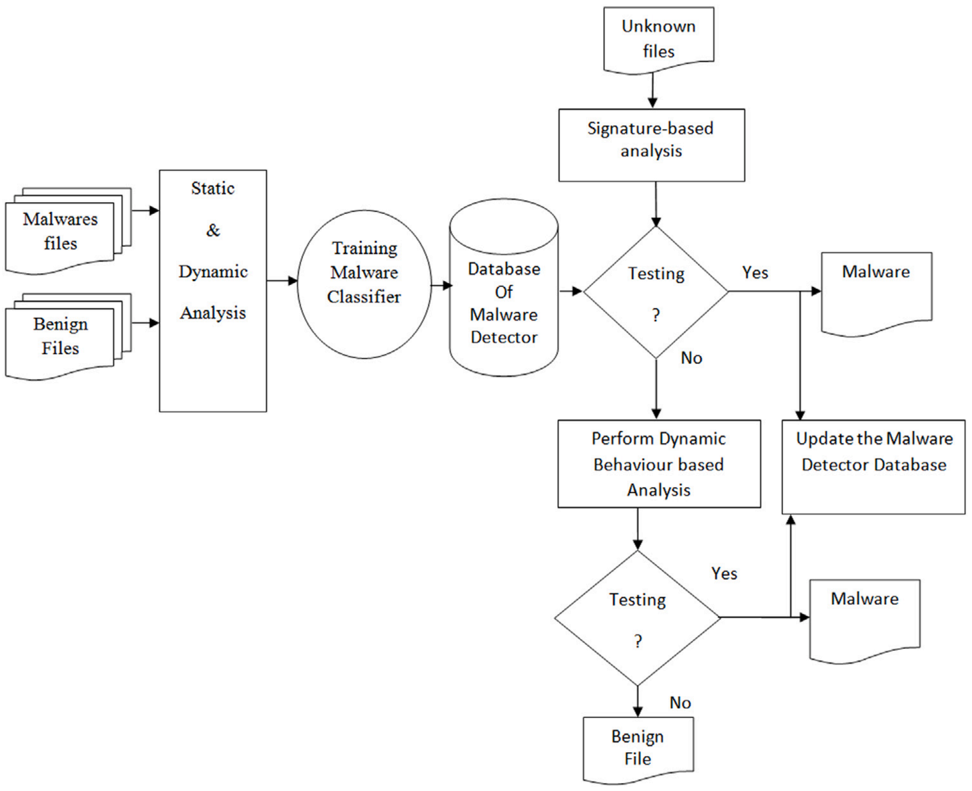
Unknown files
Signature-based analysis
Malwares files
Benign Files
Static & Dynamic Analysis
Training Malware Classifier
Database Of Malware Detector
Testing ?
Yes
No
Malware
Perform Dynamic Behaviour based Analysis
Update the Malware Detector Database
Testing ?
Yes
No
Malware
Benign File

Fig. 9. Schematic Architecture of the Proposed Hybrid Malware Detection Technique.

Table 11
Types of malware.

Type of malware	Description	Examples
Virus	It performs the destructive functions like deleting or modifying the system or user data files. Viruses are spread from one computer to other by attaching with legitimate programs. It starts working when user click over the infected program files.	Brain, Code Red, ILOVEYOU, the Chernobyl virus, Anna Kournikova and RavMonE
Worm	It replicates itself without infecting and targeting specific program files. Worms damage computer systems by consuming computer resources.	Morris, Melissa, Sasser, Stuxnet, Mydoom, Blaster.
Trojan Horse	A malicious program which seems to be a legitimate program, once it enters into the systems it starts performing malicious activities	ZeroAccess, DarkComet, Emotet Banking, Rakhni
Spyware	Spyware secretly monitor the user activities without the user knowledge. Spyware captures the keystrokes, screen monitoring, collecting logins details, financial and account information etc.	finfisher, internet optimizer, Look2Me, Movieland,
Rootkit	Rootkit is specially designed malware which is remotely control the victim computer without being perceived by user. It opens the door for the attacker to steal the users data or perform any malicious activities.	NTRootkit, SONY BMG Copy Protection rootkit
Adware	Malware delivers the advertisements automatically. Pop-up advertisements on websites and installed software are the common examples of adware	1ClickDownloader, 7search, Adoresearch, TornTv, Web Searches Toolbar,
Bot	Bot malware are the software programs which actually compromise the systems to use the resources of computer systems	Zeus, Mirai, Danabo, TrickBot,Backswap,
Ransomware	Ransomware malware encrypts the data of user and ask to pay ransom (money) to return the data back.	Cerber, Wannacry, Petya, Not Petya, Samsam

Conclusion

This research study contributes to malware detection using runtime features. It shares the fraction in the body of knowledge for malicious software analysis and detection techniques using machine learning algorithms. Nowadays, it is well understood that malware are very complex and developing at so high rate. Today these are not only used to annoy the user, steal data, destroy user's data but also they are

Table 12
Chronological study of various famous malware.

Year	Malware attacks
1986	Virus IBM-PC Brain virus was released
1987	Jerusalem virus was developed to destroy the files on every Friday. It was the first time based triggered malware.
1988	The very famous Morris Worm was created. It spread all over world though internet
1991	Michelangelo virus was designed to infect the DOS operating system.
1999	In this year more advanced malware such as Melissa Worm, Happy99 virus and Kak worm was developed. These malware were spread by internet users,
2000	A VBScript worm, ILOVEYOU infected the millions of Windows operation systems.
2001	CodeRed worm affected computer system by exploiting the buffer overflow vulnerability and Anna Kournikova email worm affected the email server system all over the world. Also, Nimda malware was released in 2001 which affect the window based computer systems.
2002–2003	Through this period, internet users got affected by unnecessary pop-up and malicious JavaScripts code. In these years, social engineering spams and worms started appearing for stealing credit card details. Few notable worms like Slammer and Blaster were created which slowed down the internet services to user and executed DoS attacks.
2004	MyDoom, Netsky and Bagle email worms were created in the steam of competition between the authors. It actually helped to improve the email filtering and scanning systems at that time.
2005	Sony rootkit malware was created which led to develop the modern day rootkit malware.
2006	A variety of monetary scams like lottery, phishing and Nigerian 419 scams were widespread during this time.
2007	Websites were compromised in large part of globe like crimeware kit was used to execute the exploits on the internet. Some famous compromised websites were The Sun, Miami Dolphins Stadium, MySpace, Photobucket and the India Times. SQL injection attack also had begun by end of this year.
2008	At this time, websites hacking and stealing FTP credentials took off place. Not only websites, PCs had been compromised and controls of them were taken to perform malicious activities without user knowledge. In June 2008, Asprox botnet (a network of compromised PCs) executed a SQL Injection attack and claimed that Walmart is victim of SQL injection attack.
2009	Gumblar malware infected windows systems which were the running the older version of OS. This weakness of older system led to create a botnet of such users.
2010	In this year a very famous cyber attack Stuxnet was done which gave news to world that what a malware attack can do to damage the industrial or any organization systems with just one click. It was so destructive which damages Iranian nuclear centrifuges. It is viewed as one of most advanced type of malware ever created in history.
2011	Trojan horse called Zero access was created for Window systems. It had been downloaded in the system through botnet. It was kept hidden in the operating system using rootkits and spread via bitcoin mining software.
2012	Shamoon attacked the computers of energy sector and CrySyS malware was very complex malware cited by Cybersecurity lab. Also, flame malware was very popular malware for cyber espionage especially in the Asia continent.
2013	It was the rising time of ransomware malware. CryptoLocker Trojan horse encrypted the users files and demanded to pay a ransom for getting the decryption key. Other malware like Gameover Zeus employed keystroke logging for stealing the login credentials of the users
2014	Regin Trojan horse was developed for mass surveillance and espionage purpose in UK and US.
2016	Locky ransomware encrypted the data of million computers in the Europe. Mirai executed very destructive DDoS attack on the various well
2017	WannaCry, one of very destructive ransomware attack happened on 12th May, 2017. It encrypted the data of million users from more than 150 countries. It did the damages of millions of dollars. In this year another ransomware Petya was released which was the advanced version of previous ransomware Petya(2016).
2018–20	In this period various crypto miners and ransomware were developed like SamSam ransomware, COVID19 RAT, Clop ransomware, Cyborg ransomware, etc.

being used by organization or countries for fulfilling their objective. In the field of computer science, security is much-needed ingredients to secure the data and resources. It is so important because the computer is used in every place or embedded digital equipments to perform the task fast and more accurately without human attraction. Because of this large scope, it makes the computer system more vulnerable to get compromised by attackers. This research study presents the evolution, present state and detection technique for malware. Malware are analyzed using static and dynamic techniques

Nowadays, malware are very specific and complex to harm the computer system by destroying computer resources or to steal the personal and confidential information of users. For developing a malware detection system, malware samples are analyzed to get the features which can represent the maliciousness. The principle of malware analysis is to extract useful information from the malware and use that information to detect and classify the malware. Two malware analysis techniques static and dynamic malware analysis techniques are used. Based on the analysis type, signature-based (antivirus software) and behaviour-based anti-malware systems are developed. Signature-based techniques suffer from two big problems. Firstly, new and unknown malware cannot be detected by signature-based techniques. Secondly, variants of malware can easily overrun the detection systems. While behaviour-based techniques can detect the new and variants of malware, additionally dynamic techniques are more robust to handle the malware obfuscation techniques than signature-based techniques. However, the practical implementation of the dynamic technique is very inflexible and time-consuming whereas signature-based techniques are fast and proficient to detect the known malware than dynamic techniques.

Developing a malware detection system using traditional approaches is not up to mark in the present time. In this paper, we have discussed malware proposed techniques in which machine learning has been used for training malware classifiers. The reason is that it consists of many algorithms which can be applied to the wide variety of malware features. In addition to the usability of the machine learning algorithms, it has many advantages as pampered to conventional malware classifiers like extracting insight from file samples, identifying unknown variations easily and reduces the human effort and time for analyzing the malware. In this paper, the categorization of proposed techniques has been done based on the analysis approaches such as static, dynamic and hybrid malware detection techniques. Every proposed approach has advantages and limitations. However, for a particular context, these proposed techniques gave promising results in malware classification. Like static techniques are faster and has a low false-positive rate but to handle the obfuscation technique while getting static features is a big challenge. On the other hand, dynamic techniques have an edge on that but implementation in real-time is cumbersome. At last, we have discussed the two-layered malware detection framework using static and dynamic features which can detect the new and known malware efficiently and accurately.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix

A.1. Evaluating measures used in malware detection system

TP (True Positive): Number of files which are classified as malware correctly.
FP (False-Positive): Number of files which are classified as malware incorrectly.
TN (True Negative): Number of files which are classified as benign correctly.
FN (False Negative): Number of files which are classified as benign incorrectly
True Positive Rate (Recall): It defines the fraction of total malware files which is true classified as malware.

$$TruePositiveRate = TP/(TP + FN) \tag{1}$$

False-Positive Rate: It defines the fraction of total benign files which is false classified as malware.

$$False - PositiveRate = FP/(FP + TN) \tag{2}$$

Precision: Precision indicates the fraction of true classified malware with the total classified malware.

$$Precision = TP/(TP + FP) \tag{3}$$

Accuracy: Accuracy measure gives the overall result performance of the malware detector. It is calculated by dividing the true positive and true negative with total number of submitted files to the malware detector.

$$Accuracy = (TP + TN)/(TP + FP + TN + FN) \tag{4}$$

ROC (Receiver Operating Characteristics) : Roc represents the false-positive rate and true positive rate in form curve.

A.2. Malware types and attacks

Table 11 explains the brief introduction of malware types with examples and Table 12 shows the background of some very famous malware [37,132–134].

References

[1] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, MalDAE : Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics, Comput. Secur. 83 (2019) 208–233, http://dx.doi.org/10.1016/j.cose.2019.02.007.
[2] P. Burnap, R. French, F. Turner, K. Jones, Malware classification using self organising feature maps and machine activity data, Comput. Secur. 73 (2017) 399–410, http://dx.doi.org/10.1016/j.cose.2017.11.016, http://linkinghub.elsevier.com/retrieve/pii/S0167404817302535.
[3] A. Damodaran, F.D. Troia, C.A. Visaggio, T.H. Austin, M. Stamp, A comparison of static, dynamic, and hybrid analysis for malware detection, J. Comput. Virol. Hacking Tech. 13 (1) (2017) 1–24, http://dx.doi.org/10.1007/s11416-015-0261-z.
[4] E.M. Dovom, A. Azmoodeh, A. Dehghantanha, D.E. Newton, R.M. Parizi, H. Karimipour, Fuzzy pattern tree for edge malware detection and categorization in iot, J. Syst. Archit. 97 (March) (2019) 1–7, http://dx.doi.org/10.1016/j.sysarc.2019.01.017.
[5] M. Ficco, F. Palmieri, Leaf : An open-source cybersecurity training platform for realistic edge-iot scenarios, J. Syst. Archit. 97 (September 2018) (2019) 107–129, http://dx.doi.org/10.1016/j.sysarc.2019.04.004.
[6] K. Khan, A. Mehmood, S. Khan, M.A. Khan, Z. Iqbal, W.K. Mashwani, A survey on intrusion detection and prevention in wireless ad - hoc networks, J. Syst. Archit. (2019) 101701, http://dx.doi.org/10.1016/j.sysarc.2019.101701.
[7] AV-TEST, Malware statistics and trends report, AV-TEST, 2020, https://www.avtest.org/en/statistics/malware.
[8] A. Bushby, F. Cybersecurity, How deception can change cyber security defences, Comput. Fraud Secur. Bull. 2019 (1) (2019) 12–14, http://dx.doi.org/10.1016/S1361-3723(19)30008-9.
[9] E. Gandotra, D. Bansal, S. Sofat, Malware analysis and classification: A survey, J. Inf. Secur. 05 (02) (2014) 56–64, http://dx.doi.org/10.4236/jis.2014.52006, http://www.scirp.org/journal/PaperDownload.aspx?DOI=10.4236/jis.2014.52006.
[10] S.M. Muzammal, M.A. Shah, S.J. Zhang, H.J. Yang, Conceivable security risks and authentication techniques for smart devices: A comparative evaluation of security practices, Int. J. Autom. Comput. 13 (4) (2016) 350–363, http://dx.doi.org/10.1007/s11633-016-1011-5.
[11] J. Singh, J. Singh, Challenges of malware analysis : Obfuscation techniques, Int. J. Inf. Secur. Sci. 7 (3) (2018).
[12] Y. Gao, Z. Lu, Y. Luo, Survey on malware anti-analysis, in: 5th International Conference on Intelligent Control and Information Processing, ICICIP 2014 - Proceedings, 2015, pp. 270–275, http://dx.doi.org/10.1109/ICICIP.2014.7010353.
[13] S. Alam, R. Horspool, I. Traore, I. Sogukpinar, A framework for metamorphic malware analysis and real-time detection, Comput. Secur. 48 (2015) 212–233, http://dx.doi.org/10.1016/j.cose.2014.10.011, arXiv:arXiv:1011.1669v3, http://linkinghub.elsevier.com/retrieve/pii/S0167404814001576.
[14] J. Singh, J. Singh, Ransomware: an illustration of malicious cryptography (2) (2019), 1608–1611, http://dx.doi.org/10.35940/ijrte.B2327.078219.
[15] X. Hu, Large-scale malware analysis, detection, and signature generation (ProQuest Dissertations and Theses), 2011, p. 190, http://search.proquest.com/docview/918832186?accountid=44888.
[16] P. Coogan, Spyeye bot versus zeus bot, 2010, http://www.symantec.com/connect/blogs/spyeye-bot-versus-zeus-bot.
[17] N. Eltayieb, R. Elhabob, A. Hassan, F. Li, A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud, J. Syst. Archit. 102 (August 2019) (2020) http://dx.doi.org/10.1016/j.sysarc.2019.101653.

[18] D. Jang, Y. Jeong, S. Lee, M. Park, K. Kwak, D. Kim, B.B. Kang, Rethinking anti-emulation techniques for large-scale software deployment, Comput. Secur. (2019) http://dx.doi.org/10.1016/j.cose.2019.02.005.

[19] S. Mahdavifar, A.A. Ghorbani, Application of deep learning to cybersecurity: A survey, Neurocomputing 347 (2019) 149–176, http://dx.doi.org/10.1016/j.neucom.2019.02.056, http://www.sciencedirect.com/science/article/pii/S0925231219302954.

[20] W. Zhang, H. Wang, H. He, P. Liu, DAMBA: Detecting android malware by ORGB analysis, IEEE Trans. Reliab. 69 (1) (2020) 55–69, http://dx.doi.org/10.1109/TR.2019.2924677.

[21] L. Liu, B.-s. Wang, B. Yu, Q.-x. Zhong, Automatic malware classification and new malware detection using machine learning, Front. Inf. Technol. Electron. Eng. 18 (9) (2016) 1–12, http://dx.doi.org/10.1631/FITEE.1601325.

[22] R. Kaur, M. Singh, Hybrid real-time zero-day malware analysis and reporting system, Int. J. Inf. Technol. Comput. Sci. 8 (4) (2016) 63–73, http://dx.doi.org/10.5815/ijitcs.2016.04.08, http://www.mecs-press.org/ijitcs/ijitcs-v8-n4/v8n4-8.html.

[23] J. Milosevic, M. Malek, A. Ferrante, Time, accuracy and power consumption tradeoff in mobile malware detection systems, Computers & Security 82 (2019) 314–328, http://dx.doi.org/https://doi.org/10.1016/j.cose.2019.01.001, http://www.sciencedirect.com/science/article/pii/S0167404818307880.

[24] H. Studiawan, F. Sohel, C. Payne, A survey on forensic investigation of operating system logs, Digital Investigation 29 (2019) 1–20, http://dx.doi.org/https://doi.org/10.1016/j.diin.2019.02.005, http://www.sciencedirect.com/science/article/pii/S1742287618303980.

[25] B. Ndibanje, K.H. Kim, Y.J. Kang, H.H. Kim, T.Y. Kim, H.J. Lee, Applied sciences cross-method-based analysis and classification of malicious behavior by API calls extraction, 2019, http://dx.doi.org/10.3390/app9020239.

[26] J. Zhang, machine learning with feature selection using principal component analysis for malware detection: A case study, January 2019.

[27] X. Sun, X. Li, K. Ren, J. Song, Z. Xu, J. Chen, Rethinking compact abating probability modeling for open set recognition problem in cyber-physical systems, J. Syst. Archit. 101 (2019) 101660, http://dx.doi.org/10.1016/j.sysarc.2019.101660.

[28] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, Classification of ransomware families with machine learning based on N -gram of opcodes, Future Gener. Comput. Syst. 90 (2019) 211–221, http://dx.doi.org/10.1016/j.future.2018.07.052.

[29] J. Singh, J. Singh, Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms, Inf. Softw. Technol. 121 (May) (2020) 106273, http://dx.doi.org/10.1016/j.infsof.2020.106273.

[30] J.P.A. Neto, D.M. Pianto, C.G. Ralha, MULTS: A multi-cloud fault-tolerant architecture to manage transient servers in cloud computing, J. Syst. Archit. 101 (2019) 101651, http://dx.doi.org/10.1016/j.sysarc.2019.101651, http://www.sciencedirect.com/science/article/pii/S1383762119304588.

[31] R. Veeramani, N. Rai, Windows API based malware detection and framework analysis, . . . conference on networks and cyber security (3) pp. 1–6.

[32] M. Christodorescu, S. Jha, J. Kinder, S. Katzenbeisser, H. Veith, Software transformations to improve malware detection, J. Comput. Virol. 3 (4) (2007) 253–265, http://dx.doi.org/10.1007/s11416-007-0059-8.

[33] E. Raff, R. Zak, R. Cox, J. Sylvester, P. Yacci, R. Ward, A. Tracy, M. Mclean, C. Nicholas, An investigation of byte n-gram features for malware classification, J. Comput. Virol. Hacking Tech. (2016) http://dx.doi.org/10.1007/s11416-016-0283-1.

[34] Y. Nagano, Static analysis with paragraph vector for malware detection.

[35] Y. Oyama, Trends of anti-analysis operations of malwares observed in API call logs, J. Comput. Virol. Hacking Tech. (2017) http://dx.doi.org/10.1007/s11416-017-0290-x.

[36] S. Sibi Chakkaravarthy, D. Sangeetha, V. Vaidehi, A survey on malware analysis and mitigation techniques, Comp. Sci. Rev. 32 (2019) 1–23, http://dx.doi.org/10.1016/j.cosrev.2019.01.002.

[37] M. Sikorski, A. Honig, Pratical malware analysis, 2012, http://dx.doi.org/10.1017/CBO9781107415324.004, arXiv:arXiv:1011.1669v3.

[38] T.Y. Wang, C.H. Wu, Detection of packed executables using support vector machines, in: Proceedings - International Conference on Machine Learning and Cybernetics, (2) (2011) pp. 717–722, http://dx.doi.org/10.1109/ICMLC.2011.6016774.

[39] S. Abimannan, R. Kumaravelu, A mathematical model of HMST model on malware static analysis, Int. J. Inf. Secur. Priv. 13 (2019) 86–103, http://dx.doi.org/10.4018/IJISP.2019040106.

[40] I. Abdessadki, S. Lazaar, A new classification based model for malicious PE files detection, Int. J. Comput. Netw. Inf. Secur. (2019) 1–9, http://dx.doi.org/10.5815/ijcnis.2019.06.01.

[41] M.H. Ligh, S. Adair, B. Hartstein, M. Richard, Malware Analyst's Cookbook, 2011, p. 746.

[42] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, R. Beyah, Acing the IOC game: toward automatic discovery and analysis of open-source cyber threat intelligence, 2016, pp. 755–766, http://dx.doi.org/10.1145/2976749.2978315.

[43] S. Schrittwieser, S. Katzenbeisser, Code obfuscation against static and dynamic reverse engineering, 2011, pp. 270–284, http://dx.doi.org/10.1007/978-3-642-24178-9_19.

[44] Z. Nemeth, Modern binary attacks and defences in the windows environment – fighting against microsoft EMET in seven rounds, 2015, http://dx.doi.org/10.1109/SISY.2015.7325394.

[45] M. Cohen, Scanning memory with yara, Digit. Investig. 20 (2017) 34–43, http://dx.doi.org/10.1016/j.diin.2017.02.005.

[46] N. Sarantinos, C. Benzaid, O. Arabiat, A. Al-Nemrat, Forensic malware analysis: The value of fuzzy hashing algorithms in identifying similarities, in: 2016 IEEE Trustcom/BigDataSE/ISPA, 2016, pp. 1782–1787, http://dx.doi.org/10.1109/TrustCom.2016.0274, http://ieeexplore.ieee.org/document/7847157/.

[47] T. Aubrey-jones, C. Science, Behaviour based malware detection, Computer (2007) 3–5.

[48] Y. Ding, X. Xia, S. Chen, Y. Li, A malware detection method based on family behavior graph, Comput. Secur. 73 (2018) 73–86, http://dx.doi.org/10.1016/j.cose.2017.10.007.

[49] W. Halfond, A. Orso, Malware detection, 2007, pp. 85–109, http://dx.doi.org/10.1016/B978-0-12-417157-2.00006-0.

[50] B. Rajesh, Y.J. Reddy, C. Chakradhar, Efficient detection of malicious worms with different analysis methods and techniques, 5 (4) pp. 1144–1148.

[51] A. Ray, Introduction to Malware and Malware Analysis: A brief overview 4 (10) (2016) pp. 22–30.

[52] H. Wang, Demadroid : Object reference graph-based malware detection in android, Secur. Commun. Netw. 2018 (2018) http://dx.doi.org/10.1155/2018/7064131.

[53] N. Kawaguchi, K. Omote, Malware function classification using apis in initial behavior, in: Proceedings - 2015 10th Asia Joint Conference on Information Security, AsiaJCIS 2015, 2015, pp. 138–144, http://dx.doi.org/10.1109/AsiaJCIS.2015.15.

[54] U. Bayer, E. Kirda, C. Kruegel, Improving the efficiency of dynamic malware analysis, in: Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10, 2010, p. 1871, http://dx.doi.org/10.1145/1774088.1774484, http://portal.acm.org/citation.cfm?doid=1774088.1774484.

[55] Malware behavior analysis, 2008, 279–287, http://dx.doi.org/10.1007/s11416-007-0074-9.

[56] S.M. Bidoki, S. Jalili, A. Tajoddin, Pbmmd: A novel policy based multi-process malware detection, Eng. Appl. Artif. Intell. 60 (August 2016) (2017) 57–70, http://dx.doi.org/10.1016/j.engappai.2016.12.008.

[57] V. Ndatinya, Z. Xiao, V. Manepalli, K. Meng, Y. Xiao, Network forensics analysis using wireshark, Int. J. Secur. Netw. 10 (2015) 91, http://dx.doi.org/10.1504/IJSN.2015.070421.

[58] N. Hoque, M.H. Bhuyan, R.C. Baishya, D.K. Bhattacharyya, J.K. Kalita, Journal of network and computer applications network attacks : taxonomy, tools and systems 40 (2014), pp. 307–324, http://dx.doi.org/10.1016/j.jnca.2013.08.001.

[59] E. Eilam, Revesing secrets of reverse Enginering.

[60] D. Gibert, C. Mateu, J. Planes, The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, J. Netw. Comput. Appl. 153 (2020) 102526, http://dx.doi.org/10.1016/j.jnca.2019.102526.

[61] C. Rathnayaka, A. Jamdagni, An efficient approach for advanced malware analysis using memory forensic technique, 2017, pp. 1145–1150, http://dx.doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.365.

[62] I. Kara, A basic malware analysis method, Comput. Fraud Secur. 2019 (6) (2019) 11–19, http://dx.doi.org/10.1016/S1361-3723(19)30064-8.

[63] J. Kavrestad, Memory analysis tools, 2020, pp. 217–224, http://dx.doi.org/10.1007/978-3-030-38954-3_19.

[64] R. Pirscoveanu, T. Hansen, J. Stevanovic, A. Czech, Analysis of malware behavior: Type classification using machine learning, in: International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), 2015, pp. 1–7, http://dx.doi.org/10.1109/CyberSA.2015.7166128.

[65] O. Aslan, R. Samet, Investigation of possibilities to detect malware using existing tools, 2017, http://dx.doi.org/10.1109/AICCSA.2017.24.

[66] Q.K. Ali Mirza, I. Awan, M. Younas, Cloudintell: An intelligent malware detection system, Future Gener. Comput. Syst. (2017) http://dx.doi.org/10.1016/j.future.2017.07.016.

[67] A. More, S. Tapaswi, Virtual machine introspection: towards bridging the semantic gap, J. Cloud Comput. 3 (1) (2014) 16, http://dx.doi.org/10.1186/s13677-014-0016-2, http://www.journalofcloudcomputing.com/content/3/1/16.

[68] Hypervisors, 2020, https://en.wikipedia.org/wiki/Hypervisor.

[69] A. Moser, C. Krügel, E. Kirda, Exploring multiple execution paths for malware analysis, 2007, pp. 231–245.

[70] A. Shabtai, R. Moskovitch, Y. Elovici, C. Glezer, Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey, Inf. Secur. Tech. Rep. 14 (1) (2009) 16–29, http://dx.doi.org/10.1016/j.istr.2009.03.003, http://www.sciencedirect.com/science/article/pii/S1363412709000041.

[71] J. Singh, J. Singh, J. Singh, Assessment of supervised machine learning algorithms using dynamic API calls for malware detection assessment of supervised machine learning algorithms using dynamic API calls for malware detection, Int. J. Comput. Appl. (2020) 1–8, http://dx.doi.org/10.1080/1206212X.2020.1732641.

[72] B. Kolosnjaji, A. Zarras, T. Lengyel, G. Webster, C. Eckert, Adaptive semantics-aware malware classification, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9721, 2016, pp. 419–439, http://dx.doi.org/10.1007/978-3-319-40667-1_21.

[73] D. Ucci, L. Aniello, R. Baldoni, Survey of machine learning techniques for malware analysis, Comput. Secur. 81 (2019) 123–147, http://dx.doi.org/10.1016/j.cose.2018.11.001, http://www.sciencedirect.com/science/article/pii/S0167404818303808.

[74] Z. Markel, M. Bilzor, Building a machine learning classifier for malware detection, in: 2014 Second Workshop on Anti-malware Testing Research (WATeR), 2014, pp. 1–4, http://dx.doi.org/10.1109/WATeR.2014.7015757, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7015757.

[75] S. Huda, S. Miah, M. Mehedi Hassan, R. Islam, J. Yearwood, M. Alrubaian, A. Almogren, Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data, Inform. Sci. 379 (2017) 211–228, http://dx.doi.org/10.1016/j.ins.2016.09.041.

[76] A. Mohaisen, O. Alrawi, M. Mohaisen, AMAL: high-fidelity, behavior-based automated malware analysis and classification, Comput. Secur. 52 (2015) 251–266, http://dx.doi.org/10.1016/j.cose.2015.04.001.

[77] F. Mira, A. Brown, W. Huang, Novel malware detection methods by using LCS and LCSS, in: 2016 22nd International Conference on Automation and Computing, ICAC 2016: Tackling the New Challenges in Automation and Computing, 2016, pp. 554–559, http://dx.doi.org/10.1109/IConAC.2016.7604978.

[78] A. Karnik, S. Goswami, R. Guha, Detecting obfuscated viruses using cosine similarity analysis, in: International Conference on Modeling and Simulation, 2007, pp. 165–170.

[79] B. D., L. Martignoni, M. Monga, Code normalization for self-mutating malware, IEEE Secur. Priv. 5 (2) (2007) 46–54.

[80] Z. B., Y. J., H. J., Z. D., W. S., Malicious codes detection based on ensemble learning, Lecture Notes in Comput. Sci. 4610 (2007).

[81] R. Moskovitch, C. Feher, N. Tzachar, E. Berger, M. Gitelman, Unknown malcode detection using OPCODE representation.

[82] K. Griffin, S. Schneider, X. Hu, T.-c. Chiueh, Automatic generation of string signatures for malware detection, 2009, pp. 101–120.

[83] D. Gavrilut, M. Cimpoesu, D. Anton, L. Ciortuz, Malware detection using machine learning, 2009, pp. 735–741.

[84] P. Beaucamps, I. Gnaedig, J.-y. Marion, I. Inria, N. Grand, E.N.-u. Loria, Behavior abstraction in malware analysis, 2010, pp. 168–169.

[85] S.K. Cha, I. Moraru, J. Jang, J. Truelove, D. Brumley, D.G. Andersen, SplitScreen : Enabling Efficient, Distributed Malware Detection, vol. 2, pp. 1–14.

[86] T.Y. Wang, C.H. Wu, Detection of packed executables using support vector machines, in: Proceedings - International Conference on Machine Learning and Cybernetics, vol. 2, pp. 717–722, http://dx.doi.org/10.1109/ICMLC.2011.6016774.

[87] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, Y. Elovici, Detecting unknown malicious code by applying classification techniques on OpCode patterns, Secur. Inform. 1 (1) (2012) 1, http://dx.doi.org/10.1186/2190-8532-1-1.

[88] A.A.E. Elhadi, M.A. Maarof, B.I. Barry, 2013-improving the detection of malware behaviour using simplified data dependent API Call Graph.pdf, 7 (5) pp. 29–42.

[89] B. Pechaz, Malware detection using hidden Markov model based on Markov blanket feature selection method (Ictck) pp. 26–27.

[90] N. Šrndić, P. Laskov, Hidost: a static machine-learning-based detector of malicious files, EURASIP J. Inf. Secur. 2016 (1) (2016) 22, http://dx.doi.org/10.1186/s13635-016-0045-0, https://jis-eurasipjournals.springeropen.com/articles/10.1186/s13635-016-0045-0.

[91] D.-h. Kim, S.-u. Woo, D.-k. Lee, T.-m. Chung, Static detection of malware and benign executable using machine learning algorithm (c), 2016, pp. 14–19.

[92] U. Narra, F. Di, T. Visaggio, A. Corrado, T.H. Austin, M. Stamp, Clustering versus SVM for malware detection, J. Comput. Virol. Hacking Tech. 12 (4) (2016) 213–224, http://dx.doi.org/10.1007/s11416-015-0253-z.

[93] R. Searles, L. Xu, W. Killian, T. Vanderbruggen, T. Forren, J. Howe, Z. Pearson, C. Shannon, J. Simmons, J. Cavazos, Parallelization of machine learning applied to call graphs of binaries for malware detection, http://dx.doi.org/10.1109/PDP.2017.41.

[94] S.D. Nikolopoulos, I. Polenakis, A graph-based model for malware detection and classification using system-call groups, J. Comput. Virol. Hacking Tech. 13 (1) (2017) 29–46, http://dx.doi.org/10.1007/s11416-016-0267-1.

[95] Q. Le, O. Boydell, B. Mac, M. Scanlon, Deep learning at the shallow end : Malware classi fi cation for non-domain experts, Digit. Investig. 26 (2018) S118–S126, http://dx.doi.org/10.1016/j.diin.2018.04.024.

[96] M. Wadkar, F. Di Troia, M. Stamp, Detecting malware evolution using support vector machines, Expert Syst. Appl. 143 (2020) 113022, http://dx.doi.org/10.1016/j.eswa.2019.113022.

[97] C. Wang, Z. Qin, J. Zhang, H. Yin, A malware variants detection methodology with an opcode based feature method and a fast density based clustering algorithm, in: 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, ICNC-FSKD 2016, 2016, pp. 481–487, http://dx.doi.org/10.1109/FSKD.2016.7603221.

[98] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, J. Yearwood, Hybrids of support vector machine wrapper and filter based framework for malware detection, Future Gener. Comput. Syst. 55 (2016) 376–390, http://dx.doi.org/10.1016/j.future.2014.06.001.

[99] Y. Ye, D. Wang, T. Li, D. Ye, An intelligent PE-malware detection system based on association mining, 2008, pp. 323–334, http://dx.doi.org/10.1007/s11416-008-0082-4.

[100] M. Bailey, J. Oberheide, J. Andersen, Z. Mao, F. Jahanian, J. Nazario, Automated classification and analysis of internet malware, Lecture Notes in Comput. Sci. 4637 (2007).

[101] P. Trinius, C. Willems, T. Holz, K. Rieck, A malware instruction set for behavior-based analysis, in: F.C. Freiling (Ed.), Sicherheit 2010: Sicherheit, Schutz und Zuverlässigkeit, Beiträge der 5. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), 5.-7. Oktober 2010 in Berlin, in: LNI, P-170, GI, 2010, pp. 205–216, https://dl.gi.de/20.500.12116/19782.

[102] K. Rieck, P. Trinius, C. Willems, T. Holz, Automatic analysis of malware behavior using machine learning, J. Comput. Secur. 19 (4) (2011) 639–668, http://dx.doi.org/10.3233/JCS-2010-0410.

[103] J. Hegedus, Y. Miche, A. Ilin, A. Lendasse, Methodology for behavioral-based malware analysis and detection using random projections and k-nearest neighbors classifiers, in: 7th International Conference on Computational Intelligence and Security ({CIS}2011), 2011.

[104] M. Vasilescu, L. Gheorghe, N. Tapus, Practical malware analysis based on sandboxing, in: Proceedings - RoEduNet IEEE International Conference, 2014, http://dx.doi.org/10.1109/RoEduNet-RENAM.2014.6955304.

[105] A.A.E. Elhadi, M.A. Maarof, B.I.A. Barry, H. Hamza, Enhancing the detection of metamorphic malware using call graphs, Comput. Secur. 46 (2014) 62–78, http://dx.doi.org/10.1016/j.cose.2014.07.004.

[106] M. Ghiasi, A. Sami, Z. Salehi, Dynamic VSA: a framework for malware detection based on register contents, Eng. Appl. Artif. Intell. 44 (2015) 111–122, http://dx.doi.org/10.1016/j.engappai.2015.05.008.

[107] Y. Ki, E. Kim, H.K. Kim, A novel approach to detect malware based on API call sequence analysis, Int. J. Distrib. Sens. Netw. 2015 (2015) http://dx.doi.org/10.1155/2015/659101.

[108] Z.-p. Pan, C. Feng, C.-j. Tang, Malware classification based on the behavior analysis and back propagation neural network, 02001, 2016.

[109] B.N. Narayanan, O. Djaneye-Boundjou, T.M. Kebede, Performance analysis of machine learning and pattern recognition algorithms for Malware classification, in: 2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS), 2016, pp. 338–342.

[110] I.K. Cho, T.G. Kim, Y.J. Shim, M. Ryu, E.G. Im, Malware analysis and classification using sequence alignments malware analysis and classification using sequence alignments, 8587 (June) http://dx.doi.org/10.1080/10798587.2015.1118916.

[111] J. Ming, Z. Xin, P. Lan, D. Wu, P. Liu, B. Mao, Impeding behavior-based malware analysis via replacement attacks to malware specifications, J. Comput. Virol. Hacking Tech. 13 (3) (2017) 193–207, http://dx.doi.org/10.1007/s11416-016-0281-3.

[112] M. Wagner, A. Rind, N. Thür, W. Aigner, A knowledge-assisted visual malware analysis system: Design, validation, and reflection of KAMAS, Comput. Secur. 67 (2017) 1–15, http://dx.doi.org/10.1016/j.cose.2017.02.003, arXiv:1612.06232.

[113] W. Mao, Z. Cai, D. Towsley, Q. Feng, X. Guan, Security importance assessment for system objects and malware detection, Comput. Secur. 68 (2017) 47–68, http://dx.doi.org/10.1016/j.cose.2017.02.009.

[114] A. Pektaå, T. Acarman, Classification of malware families based on runtime behaviors, J. Inf. Secur. Appl. 37 (2017) 91–100, http://dx.doi.org/10.1016/j.jisa.2017.10.005, http://linkinghub.elsevier.com/retrieve/pii/S2214212617301643.

[115] J. Stiborek, T. Pevný, M. Rhák, Multiple instance learning for malware classification, Expert Syst. Appl. 93 (2018) 346–357, http://dx.doi.org/10.1016/j.eswa.2017.10.036, arXiv:1705.02268.

[116] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, Detection of advanced persistent threat using machine-learning correlation analysis, Future Gener. Comput. Syst. 89 (2018) 340–349, http://dx.doi.org/10.1016/j.future.2018.06.055.

[117] M. Alaeiyan, S. Parsa, M. Conti, Analysis and classification of context-based malware behavior, Comput. Commun. (2019) http://dx.doi.org/10.1016/j.comcom.2019.01.003.

[118] L. Xiaofeng, J. Fangshuo, Z. Xiao, Y. Shengwei, S. Jing, P. Lio, ASSCA: API sequence and statistics features combined architecture for malware detection, Comput. Netw. 157 (2019) 99–111, http://dx.doi.org/10.1016/j.comnet.2019.04.007, http://www.sciencedirect.com/science/article/pii/S138912861930461X.

[119] A. D., V.K. K.A., S.C. S., V. P., Malware traffic classification using principal component analysis and artificial neural network for extreme surveillance, Comput. Commun. 147 (2019) 50–57, http://dx.doi.org/10.1016/j.comcom.2019.08.003, http://www.sciencedirect.com/science/article/pii/S0140366419306693.

[120] A. Namavar Jahromi, S. Hashemi, A. Dehghantanha, K.K.R. Choo, H. Karimipour, D.E. Newton, R.M. Parizi, An improved two-hidden-layer extreme learning machine for malware hunting, Comput. Secur. 89 (2020) 101655, http://dx.doi.org/10.1016/j.cose.2019.101655.

[121] Ç. Yücel, A. Koltuksuz, Imaging and evaluating the memory access for malware, Forensic Sci. Int. Digit. Investig. 32 (2020) 200903, http://dx.doi.org/10.1016/j.fsidi.2019.200903.

[122] M. Rabbani, Y.L. Wang, R. Khoshkangini, H. Jelodar, R. Zhao, P. Hu, A hybrid machine learning approach for malicious behaviour detection and recognition in cloud computing, J. Netw. Comput. Appl. 151 (2020) 102507, http://dx.doi.org/10.1016/j.jnca.2019.102507.

[123] J.C. Rabek, R.I. Khazan, S.M. Lewandowski, R.K. Cunningham, Detection of injected, dynamically generated, and obfuscated malicious code. New York, NY, USA, in: Proceedings of the 2003 ACM workshop on Rapid malcode, 2003, pp. 76–82.

[124] M. Collins, J. Mchugh, A protocol graph based anomaly detection system, 2008.

[125] R. Mangialardo, J. Duarte, Integrating static and dynamic malware analysis using machine learning, 13 (9) (2015) pp. 3080–3087, http://dx.doi.org/10.1109/TLA.2015.7350062.

[126] P.V. Shijo, A. Salim, Integrated static and dynamic analysis for malware detection, Procedia Comput. Sci. 46 (Icict 2014) (2015) 804–811, http://dx.doi.org/10.1016/j.procs.2015.02.149.

[127] E.I. Edem, C. Benzaid, A. Al-Nemrat, P. Watters, Analysis of malware behaviour: Using data mining clustering techniques to support forensics investigation, in: Proceedings - 5th Cybercrime and Trustworthy Computing Conference, CTC 2014, 2015, pp. 54–63, http://dx.doi.org/10.1109/CTC.2014.10.

[128] P. O'kane, S. Sezer, K. McLaughlin, Detecting obfuscated malware using reduced opcode set and optimised runtime trace, Secur. Inform. 5 (1) (2016) 2, http://dx.doi.org/10.1186/s13388-016-0027-2, http://security-informatics.springeropen.com/articles/10.1186/s13388-016-0027-2.

[129] M. Nauman, N. Azam, J. Yao, A three-way decision making approach to malware analysis using probabilistic rough sets, Inform. Sci. 374 (2016) 193–209, http://dx.doi.org/10.1016/j.ins.2016.09.037, http://linkinghub.elsevier.com/retrieve/pii/S0020025516308969.

[130] A. Pfeffer, B. Ruttenberg, L. Kellogg, M. Howard, C. Call, A. O'Connor, G. Takata, S.N. Reilly, T. Patten, J. Taylor, R. Hall, A. Lakhotia, C. Miles, D. Scofield, J. Frank, Artificial intelligence based malware analysis, 2017, pp. 1–38, arXiv:1704.08716, http://arxiv.org/abs/1704.08716.

[131] S. Huda, R. Islam, J. Abawajy, J. Yearwood, A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection, 2018, http://dx.doi.org/10.1016/j.future.2017.12.037.

[132] S. Bosworth, E. Whyne, M.E. Kabay, Computer Security Handbook, Wiley, 2014.

[133] J. Love, Malware types and classifications, Lastline (2018) https://www.lastline.com/blog/malware-types-and-classifications.

[134] Wikipedia, Malware, 2020, https://en.wikipedia.org/wiki/Malware.

Jagsir Singh is a Ph.D. Research Scholar in the department of Computer Science and Engineering at the Punjabi University Patiala, India. He did B.Tech. in Computer Science and Engineering from Punjabi University, Patiala in 2014. He did M.Tech. in Information Technology form UIET, Panjab University, Chandigarh, India in 2016. His areas of interest include Information and Network security, Malware Analysis, Cognitive Radio and Machine Learning.

Dr. Jaswinder Singh is an Associate Professor in the department of Computer Science and Engineering, Punjabi University, Patiala, India. He did Ph.D. in Computer Engineering from Punjabi University, Patiala. He possesses 16 years of teaching experience. His work is published and cited in highly reputed journals of Elsevier, Springer, Taylor and Francis and IEEE. His areas of interest include Network Security, Malware Analysis, Machine Learning and Cloud Computing.