Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Review Article

A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges

Pascal Maniriho [*], Abdun Naser Mahmood, Mohammad Jabed Morshed Chowdhury

Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC, Australia

ARTICLE INFO

Article history:
Received 23 March 2021
Received in revised form 10 October 2021
Accepted 26 November 2021
Available online 20 December 2021

MSC:
00-01
99-00

Keywords:
Malware analysis
Malware behaviour analysis
Malware detection
Dynamic analysis
Memory analysis
Machine learning
Deep learning

ABSTRACT

There has been an increasing trend of malware release, which raises the alarm for security professionals worldwide. It is often challenging to stay on top of different types of malware and their detection techniques, which are essential, particularly for researchers and the security community. Analysing malware to get insights into what it intends to perform on the victim's system is one of the crucial steps towards malware detection. Malware analysis can be performed through static analysis, code analysis, dynamic analysis, memory analysis and hybrid analysis techniques. The next step to malware analysis is the detection model's design using malware's extracted patterns from the analysis. Machine learning and deep learning methods have drawn attention to researchers, owing to their ability to implement sophisticated malware detection models that can deal with known and unknown malicious activities. Therefore, this survey presents a comprehensive study and analysis of current malware and detection techniques using the snowball approach. It presents a comprehensive study on malware analysis testbeds, dynamic malware analysis and memory analysis, the taxonomy of malware behaviour analysis tools, datasets repositories, feature selection, machine learning and deep learning techniques. Moreover, comparisons of behaviour-based malware detection techniques have been grouped by categories of machine learning and deep learning techniques. This study also looks at various performance evaluation metrics, current research challenges in this area and possible future direction of research.

Contents

* Corresponding author.
E-mail address: P.Maniriho@latrobe.edu.au (P. Maniriho).

https://doi.org/10.1016/j.future.2021.11.030
0167-739X/© 2021 Elsevier B.V. All rights reserved.

1. Introduction

The presence of malicious software (malware) has been around for many decades. Any software designed to damage or harm a computer system's resources or leak its confidential information through unauthorised access either remotely or locally under the control of a cybercriminal/cyber attacker is known as a malware. Various forms of malware, such as .exe, .dll, .ocx or .drv executable files, macros and scripts that operate in the system's background, have been already reported in the literature [1]. Considering the execution environment, the Windows operating system (OS) is by far the most targeted OS by malware due to its widespread usage [2,3]. According to the current statistics, existing malicious software is responsible for 80% of cyber-attacks worldwide [1]. Many malware spread via the Internet and target government, organisations and individual confidential resources. Cybercriminals use sophisticated malware to leak or compromise confidential information from a compromised system [1,2]. Cybercriminals are increasingly developing sophisticated malware to launch specialised cyber attacks on particular organisations, industries, critical networks, and systems infrastructures [3,4]. Fig. 1 depicts the total number of malware (in a million) over the last ten years (from January 2010 until March 2020) which shows how malware threats have continued to threaten various global industries [3]. Hence, defending against malware attacks seems to be a never-ending process as both security researchers and cybercriminals continue to innovate to find new attack vectors and improve their techniques.

Malware analysis can be mainly performed through static analysis, code analysis, dynamic analysis, memory analysis, and hybrid analysis techniques [5–7]. Static analysis involves analysing an executable binary file without executing/running it. Metadata associated with a suspected binary file can be extracted and analysed using static analysis. In many cases, the static analysis does not provide all the relevant information about an executable file; however, it can still reveal crucial information about the malware's objectives [8]. Antivirus programs often rely on a database of static signatures (patterns) to detect malicious software's activities. However, malware often evades static analysis using a variety of sophisticated detection evasion techniques including obfuscation, dead code insertion, and packing [9]. Code analysis can be applied through static code analysis or dynamic code analysis. In static code analysis, the suspected file/program is disassembled using some dissembler programs to obtain and analyse its source code before it is executed. Static code analysis helps to identify security issues in source code such as exploitable vulnerabilities. Dynamic code analysis deals with debugging the suspected program in a real or virtual environment. The program's source code is executed with different test inputs to identify security issues caused by its code while interacting with other programs or systems.

Dynamic analysis is a technique for malware behaviour analysis. Detecting malware by analysing its behaviours offers many advantages over static analysis. Most malware executables share similar behaviours, giving dynamic analysis the ability to detect both known and zero-day malware attacks. Dynamic analysis is performed using sandbox programs like Cuckoo sandbox or CWSandbox [10], which allows monitoring malware behaviours at runtime. The dynamic analysis behavioural report defines a malware executable file's behaviours after its execution on the victim's host. Such behaviours include downloading infected files from the Internet, running malicious operating system tasks, delete and creating files, exfiltration of sensitive data through remote access, denying users from getting access to resources or slowing down the network.

Malware behaviour analysis can also be conducted through memory analysis. Memory analysis is performed by taking memory dumps of the infected system's physical memory at runtime. Malware behavioural information can then be extracted from the captured memory dump using advanced tools such as the Volatility framework [11]. More importantly, it is impossible for memory to disguise the aforementioned malicious operations [9]. Essentially, in order for a program (either benign or malicious) to be executed, it must be first loaded in memory and therefore, making the memory analysis technique the most suitable and preferred technique for identifying sophisticated malware behaviours [9,12].

The hybrid analysis is another solution for malware behaviour analysis. It integrates the functionalities of more than one of the aforementioned analysis techniques. It is worth noting that combining different malware analysis techniques can reveal more information on the malware, improving detection. After the features have been generated using any of the above malware analysis techniques, the next step is to design and train a sophisticated detection model using relevant features and machine learning or deep learning techniques. Machine learning and deep learning techniques have shown promising results in malware detection [2].

Contribution of this work

This paper presents a survey on the existing behaviour-based malware detection techniques and a broad range of related topics. More specifically, our main contributions include presenting:

- A survey that categorises malware detection systems and testbeds including their merits and demerits and provides detailed taxonomies of datasets and malware behaviour analysis tools.

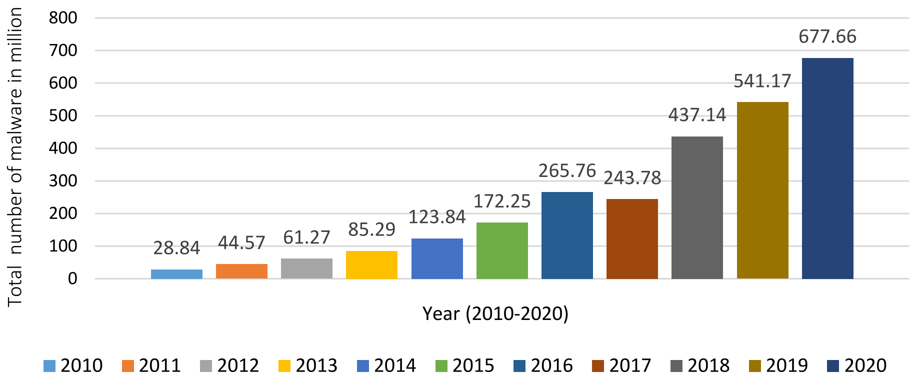

Fig. 1. Total number of malware over the last ten years, data from [3].

Table 1
Summary of topics presented in the previous surveyed work and our work.

Survey	Dynamic analysis	Memory analysis	Malware category	Taxonomy						
				Testbeds	Dataset	Tools	B.Feat	FS	ML	DL
[13]	✓	x	✓	✓	x	x	✓	✓	✓	x
[14]	✓	x	x	x	x	x	✓	x	✓	x
[15]	✓	x	x	x	✓	x	✓	x	✓	x
[16]	✓	x	x	x	x	✓	x	x	x	x
[17]	✓	x	x	x	✓	x	✓	x	✓	x
[18]	✓	x	x	x	x	x	x	x	✓	x
[19]	✓	x	x	x	x	x	✓	x	✓	✓
[20]	✓	x	x	x	x	x	x	x	✓	x
Our survey	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

- A taxonomy of feature selection, machine learning and deep learning techniques for malware detection.
- A comparison of behaviour-based malware detection techniques grouped by categories of machine learning and deep learning techniques.
- The current challenges in malware detection that have not been covered adequately in the previous works.

Comparison against the previous surveyed works

Malware analysis and detection are active research fields where the existing works have been reviewed from various perspectives. However, it is important to mention that conducting multiple surveys on a similar or different topic is common and multiple surveys on malware analysis and detection exist in the literature. However, they differ based on their scope, analysis, selected or reviewed articles, and each survey paper's outcome. Our survey is compared against the previous surveyed work to highlight the differences. Throughout our review process, we identified eight survey papers on malware behaviour analysis and detection and key aspects of each survey have been briefly summarised in Table 1.

Table 1 highlights some of the limitations that we intend to address in this work. The ✓ tick indicates the topic covered by each paper and the cross (x) denotes either the topic is not discussed or lacks in-depth discussion. This survey considers ten main topics, namely, dynamic malware analysis, memory analysis, general categories of malware, testbeds, datasets, behaviour analysis tools, behavioural features (B.Feat), feature selection (FS), machine learning (ML) and deep learning (DL) techniques. More importantly, this information is provided to show that this work does not overlap with the previous surveyed works to emphasise and ensure the contributions and novelty of our work.

The remaining part of this work is structured as follows: Section 2 presents the methodology and scope of this work, Section 3 presents a detailed background and Section 4 discusses various taxonomies of malware behaviour analysis techniques. Section 5 elaborates machine learning and deep learning techniques. Mobile malware detection techniques are discussed in Section 6, while performance evaluation metrics are presented in Section 7. Section 8 discusses current challenges and future research directions, while conclusion on this work is presented in Section 9.

2. Survey and taxonomy methodology

This section presents the approach used for selecting relevant papers, various data sources, paper search criteria, taxonomy development, and the scope of this work.

2.1. Paper search approach and inclusion criteria

The taxonomies and detailed analysis presented in this work are based on a review of a large number of papers on malware detection using dynamic analysis and memory analysis techniques. We have used the snowball approach [21] to find and select relevant papers. Given a paper *P*, the snowball approach involves using a list of references of *P* or papers citing *P* to find a large set of papers to review. It employs the backward approach (finding new papers using a list of references) and the forward approach where new papers are identified based on papers that cited the papers under consideration (papers being examined).

Six databases in computer science and information technology publications, namely, IEEE Xplore (IEE/IEEE), Association for Computing Machinery (ACM) Digital Library, Web of Science, ScienceDirect, SpringerLink and Google Scholar were selected and each was queried for seven key terms: "malware analysis", "malware behaviour analysis", "malware detection", "dynamic malware analysis", "memory analysis", "machine learning" and "deep learning" which led to finding relevant papers. As some relevant research papers might not be indexed in the selected databases, it is worth noting that all papers are not taken from the aforementioned databases; however, they are our primary sources. In addition, we have limited our research scope by mainly focusing on behaviour-based malware detection using machine learning and deep learning techniques in the Windows OS. However, this work has also covered malware detection in the Android OS.

Identifying a start set of relevant papers is the first step before applying the snowball approach and is considered critical as failure may lead to irrelevant papers being used in the review. Therefore, using the defined key terms, 70 papers were initially selected from the above databases. Each paper's abstract and contents were screened for potential inclusion into our study. The paper screening was conducted by all authors and only 43 papers were selected to be used as the starting set. These selected papers have been published in well-reputed journals such as ACM Computing Surveys, Future Generation Computer Systems, Computers & Security, Expert Systems with Applications, Journal of Network and Computer Applications, IEEE Access, Computer Science Review, etc. and some IEEE conferences. After applying the snowball approach to the initial starting set, a large set (final set) of relevant papers was generated that have been used throughout the review process. It is important to mention that Google Scholar was mainly used to perform a forward search as it keeps track of papers' citations. The inclusion of papers is determined based on aspects such as citation score, publication venue (peer-reviewed conference or Journal) in which the paper has been published, the paper's focus is on behaviour-based malware detection or is reviewing current techniques in behaviour-based malware detection. All research studies that do not focus on malware behaviour analysis and detection were excluded including papers from non-English journals or conferences.

2.2. Method for taxonomy development

Knowledge on a particular research field or subfield can be classified through taxonomy [22]. As stated in the Lexico [23], a taxonomy is defined as "a scheme of classification". This work presents a taxonomy for behaviour-based malware detection designed by adapting the method for taxonomy development proposed in Nickerson et al.'s work [24]. Their method is based on an iterative process comprising of four main steps, (1) define meta-characteristic, (2) determine stopping conditions, (3) select an empirical-to-conceptual or conceptual-to-empirical approach, and (4) follow the approach iteratively, until the stopping conditions are met. Following these steps provided in [24] and based on our classification from the literature, we have identified 3 dimensions (behavioural feature generation, learning and detection approach, and performance evaluation and validation) and 10 sub-dimensions (testbeds, tools, platforms, feature extraction, behavioural features, datasets, feature selection, machine learning, deep learning, and evaluation metrics) including their categories which are relevant for behaviour-based malware detection taxonomy.

3. Background

Malicious software is known by various names such as malware, malicious code, malcode, harmful software or malicious program, and these names have been used interchangeably in the existing work. Several definitions have been used to describe malicious software [6,25]. In this paper, we define malware as any software which is designed to damage or harm computer system's resources or leak its confidential information through unauthorised access either remotely or locally. The following are general categories of malware.

Worm–A standalone malicious software that spreads from one computer to other computers over the network by replicating itself.

Virus–Malicious software having the ability to copying and replicating itself to another legitimate program. Once executed, it replicates itself by altering other computer programs and injecting its malicious code [13].

Adware–Short for advertising-supported software, it is unwanted software designed to automatically deliver or throw adverts on the user's screen (mostly in the web browser), which are often displayed as pop-ups.

Trojans (a.k.a. Trojan horse)–A category of malware that often appears as a legitimate software which is typically employed by hackers to gain access to the target system.

Backdoor (a.k.a. Remote Access Trojan)–A type of malware that denies authentication to the authorised users while giving the cybercriminals the ability to remotely access systems applications such as file servers and databases in addition to issuing commands that updates the malware.

Spyware–This type of malware operates by spying on the activities of a user without their knowledge. Some of the spying capabilities include collecting keystrokes, monitoring user's activities, and harvesting various data.

Ransomware–Malware that infects a system and prevents users from accessing their confidential files until a ransom is paid [5].

Information stealer–A type of malware designed to steal typed keystrokes or confidential data like bank credentials from the victim's system. [26].

Bots–Automated malicious software programs designed to perform some repetitive activities. Cybercriminals command these bots to carry out denial-of-service (DoS), distributed denial-of-service (DDoS) attacks or send several spam emails.

Rootkit–A Malicious software that gives cybercriminals privileged access to the infected system and disguises its presence to the system's users [27].

Fileless malware–A computer memory-based malicious software that infects the system through the volatile memory and functions without writing any part of its activities to the physical hard drive [28].

Hybrid malware–A type of malware that incorporates various forms of malicious codes into a new malware variant to achieve more powerful attack functionalities which complicate detection.

4. Techniques for analysing malware behaviours

This section discusses malware analysis and why malware analysis should be performed. Besides, testbeds, tools, malware behaviour analysis techniques, dataset sources and feature selections methods are also presented.

4.1. Malware behaviour analysis

Malware behaviour analysis involves understanding how the malware works, detecting it and eliminating it before or after executing its malevolent activities. Thus, observing whether the behaviours of a given executable program conform to the user requirements is one of the keys to finding whether the behaviour of that program is malicious [29]. According to [29], a typical malware's behaviours can be usually described in three aspects or characteristics. The first aspect is deformation and hiding its structure to evade analysis and detection (described as its evasion behaviour). Second, modifying the operating system resources (described as its systematic behaviour). Lastly, activities related to network communication (described as its network behaviours). This work presents malware features that can be extracted from PE files, a Windows operating system's file format. According to [3], Windows is the most worldly distributed OS and PE malware samples are the most produced compared to other malware's file formats and therefore, making Windows the most targeted OS followed by Android OS. Some of the behavioural features that can be extracted from PE file during malware behaviour analysis are presented in Tables 4–6. Additionally, malware behavioural features from Android files are also presented in Section 6.

4.2. Malware analysis testbeds taxonomy

Malware analysis testbed refers to various environments for monitoring and analysing malware's behaviours. Malware analysis must be conducted in a secured and isolated environment to avoid any damage it may cause during the analysis. Therefore, this section discusses various categories of malware analysis testbeds.

4.2.1. Bare-metal

Executing malware on a clean system running on real hardware is one way to analyse malware's behaviours, and such malware analysis environment is known as a "bare-metal" [30]. Malware analysis on a bare-metal can be mainly run in a user-mode, or kernel-mode [31]. Nevertheless, this approach is only convenient for the analysis of user-mode malware as in case of kernel model malware such as a rootkit; the malware can interrupt the analysis and greatly infect the system or harm the hardware of the device. As executing malware on a clean system can damage the system or leave it in a critical state (insecure state), the system must be returned to its clean state (by undoing all malware's operations) after executing each user-mode malware. For kernel model malware, the system must be restored by formatting the infected system after each rootkit malware analysis and installing a new clean operating system.

4.2.2. Sandbox environment

Sandbox environment allows researchers and security analysts to execute and analyse malware programs without affecting the native operating system, its applications, or the hardware on which they are executing. Sandboxes are also employed to protect systems and network infrastructures against malware stealthy attacks or exploits from unknown malware. There exist various automated sandboxes designed for malware analysis such as Cuckoo and Virmon sandboxes [32].

4.2.3. Virtual machine

Installing and running malware analysis tools on a virtual machine (VM) can protect hardware, operating system and different applications of the host system. A virtual machine allows taking a snapshot of a clean system which is used to revert the infected system to the clean state after executing and analysing each malware executable. Therefore, the clean system's snapshot must always be taken and saved. Besides, the system's snapshot can be captured while the malware is running or after infecting the system, and the system can be reverted to the previous state using the captured clean snapshot. There are two possible options to analyse malware inside a virtual machine [31]. The first option is accomplished by including malware and analysis tools inside a virtual machine while the second option involves installing and running the malware analysis tool on the host operating system and then trace the operations performed by the malware from the virtual machine.

4.2.4. System emulation

Another alternative malware testbed is a system emulator that enables to fully separate malware execution from the hardware [31]. An emulator is a type of software that allows a host computer to act or operate as another computer (guest computer). It gives the host computer the ability to execute malware executable as if it is running in the guest machine. As the same hardware is shared between the emulator and the host computer, it is highly advised not to open another program while an emulator is running. Moreover, it is crucial to mention that malware is not executed on physical hardware, but instead, it is executed on an internal-guest software component created by the emulator, and the guest component keeps track of operations performed by the malware.

4.2.5. Volatile memory acquisition

Memory acquisition (MA) involves capturing a memory dump (memory image) while a malware or benign program is running or after being executed and store it to the host or external storage media for further analysis. Any of the testbed methods mentioned above can be used to execute the malware, and a memory acquisition tool must also be installed on the analysis environment. Acquiring memory dump with hardware can be achieved using hardware tools such as "Tribble" (through PCI express card) [33] or "Firewire". On the other hand, software-based acquisition tools are easy to use and user friendly, can be accessed at a low cost and are easy to deploy. Nonetheless, software-based memory acquisition tools can be detected by some of the sophisticated malicious software. In addition, some sophisticated malware utilises anti-debugging techniques that prevent the acquisition tool from capturing volatile memory images. Therefore, an empty memory image will be captured once any anti-debugging and anti-memory dumping are running in memory [33].

4.2.6. Commercial and industrial testbed

Commercial testbed is mostly preferred by security professionals and other systems security analysts. IBM X-Force Command Centre is the first developed commercial malware testbed [34]. It is implemented with a mobile command Cyber Tactical Operations Center that provides various services. The Ixia BreakingPoint and Cyber Exata [35] are other commercially available testbeds. Ixia provides the ability to simulate about 28, 000 live malware attacks and send them to devices being tested while Cyber Exata can provide various cyber range services.

4.2.7. Internet of thing(IoT) testbed

IoT testbeds have also become a necessity as they can help to reveal weaknesses that may exist in the IoT applications, devices and communication, and producing analysis or assessment reports that can be used in the design and development of new malware detection techniques. In [50], an open-source automated testbed for IoT networks was implemented. Their platform can be easily extended and thus, allowing both security professionals and researchers to add new security assessment modules, analysis and testing features. Other testbeds for IoT networks were proposed in [51,52]. Table 2 summarises the merits and demerits of each malware analysis testbed.

4.3. Malware behaviour analysis tools

Some malware analysis stools are sandbox-based while other tools work together with a sandbox in order to monitor and gather different information on the portable executable file while it is being executed. They allow researchers to pinpoint the behavioural artefacts associated with the operations performed by the malware under analysis.Table 3 presents various tools for performing both dynamic malware analysis and memory analysis.

Table 2
Summary of merits and demerits of each malware analysis testbed.

Testbed	Easily deployable	Fast system restore	Hardware damage	Can be Detected by malware	Requires high cost
Bare-metal	✓	x	✓	x	✓
Sandboxing	✓	✓	x	✓	x
Virtual machine	✓	✓	x	✓	x
MA (hardware-based)	x	x	x	✓	✓
MA (software-based)	✓	✓	✓	✓	x
System emulation	✓	Not required	x	✓	x

Table 3
Taxonomy of behaviour-based malware analysis tools including supported platforms-Windows OS (W), Linux OS (L) and macOS (M).

Tools	Description	Analysis		Platform		
		DA	MA	W	L	M
Cuckoo sandbox [32]	Advanced automated sandbox for malware analysis	✓	x	✓	✓	✓
DumpIT [36]	Takes physical memory snapshot of the host and save it to the same location where DumpIT executable is located	x	✓	✓	x	x
Volatility Framework [5]	An advanced framework that examines volatile memory images for malicious activities discovery	✓	✓	✓	✓	✓
Wireshark [37]	A sniffing tool that captures live network traffic	✓	x	✓	✓	✓
ProcMonitor [38]	An advanced windows monitoring tool that displays real time file system, processes and registry activity	✓	✓	✓	x	x
RAMMap [1]	Monitors and analyses memory used by running applications	✓	✓	✓	x	x
InetSim [39]	Advanced network simulator that provides the malware with a virtual internet to interact with during dynamic analysis. Therefore, allowing it to reveal its malicious network activities without affecting any resources	✓	✓	✓	✓	✓
CWSandbox [40]	An automated sandbox that analyses and evaluates actions performed by a malware executable at runtime	✓	x	✓	x	x
TCPDump [41]	Advanced network traffic analysis tool that shows incoming and outgoing TCP/IP packets	✓	x	x	✓	✓
Process Explorer [42]	System and task manager's monitoring tool that displays details of all system's running processes	✓	x	✓	x	x
Rekall [43]	An advanced framework that performs memory analysis	✓	✓	x	✓	x
Ram Capturer [44]	Memory acquisition tool that captures live memory dumps of RAM	✓	✓	✓	x	x
Memoryze [36]	This tool Can capture and save a full volatile memory of VM or physical machine	✓	✓	✓	✓	✓
Regshot [2]	Captures the snapshot of the registry in real time	✓	x	✓	x	x
VirMon [32]	Automated sandbox for malware analysis	✓	x	✓	x	x
WinApi32 tool [45]	This tool has the ability to capture all binary executable's invoked API calls including all arguments, returned values of APIs, the content of registers, etc.	✓	x	✓	x	x
WINAPIOverride32 [46]	A sophisticated API monitoring tool for both 64-bits and 32-bits processes (advanced API hooking tool)	✓	x	✓	x	x
AVClass tool [47]	An advanced online automated tool that accepts a malware executable sample as input and label it to determine its family of belongingness such as adware, Trojan, spyware, etc.)	✓	x	✓	✓	x
Noriben [48]	Windows-based automated sandbox for dynamic malware analysis	✓	✓	✓	x	x
NtTrace [49]	A native API tracer for Windows designed to run on command prompt	✓	x	✓	x	x
Limon [48]	Linux-based sandbox for performing dynamic malware analysis	✓	x	x	✓	x

4.4. *Dynamic malware analysis*

Dynamic malware analysis involves running a malware executable file in an isolated and secured environment using automated tools that allows monitoring and extracting malware's behaviours while being executed. Even though the binary executable's codes or structure may frequently change, in many cases, new malware's behaviours will be similar to the previous ones. Therefore, the majority of new malware families can be detected through the dynamic analysis approach [53]. Nevertheless, some sophisticated malware will not run when they discover that they are being executed in a sandbox or virtual machine environment [54]. A typical workflow of dynamic malware analysis is illustrated in Fig. 2(a).

4.5. *Memory-based malware analysis*

Live response analysis and memory image analysis are two categories of memory analysis employed to collect volatile data from the system under analysis or investigation [55]. In live response, forensic investigators execute their self-compiled trusted binaries in the comprised system in order to retrieve the volatile memory data. However, the live response can significantly modify

Table 4
Different dynamic and memory-based characteristics/features that can be used to represent a malware executable program.

Feature category	Behavioural feature	Description
Systematic behaviours	API call sequences	Suspicious malware's API calls
	API call frequencies	Frequencies of suspicious API calls
	API return values	Values returned by each malicious API call
	Registry operations	Operations on registry keys such as creating, deleting, reading or modifying registry keys
	Loaded DLLs	Information on the list of loaded DLLs
	DLL function calls	All DLLs function imported by the malware while running
	File system Changes (operation on file)	Different changes on the file system (deleting, copying or renaming files)
	Mutexes' operations-based features	Malicious mutex operations created by mutex module (creating, locking or unlocking mutexes)
	Values of register	Returned values for registers
	File changes in suspicious locations	Suspicious file change(creating, renaming deleting, writing or modifying files)
	Suspicious DLL location	Location of loaded DLL (where paths have been predefined)
	Process	Process identifier (PID) and name of suspicious processes
	Threads	Created malicious threads
Network behaviours	Network operations	Opened suspicious ports, sockets or connection to malicious IP address
	Web browsing history	Accessed malicious URLs/domain names

the volatile data and overwrite valuable evidence. Hence, memory image analysis technique was introduced to remove this limitation to help mitigate the risk of losing some valuable volatile memory artefacts such as cached data and terminated processes that could be modified through live response analysis. Memory image analysis reduces the risk of overwriting volatile data and facilitates the analysis as it does not load (create) additional processes in the system like live response [56,57]. Captured memory images can be analysed using automated frameworks/tools such as Volatility framework [12], Redline [44], etc., which reveal any malicious activities performed by the malware. In addition, memory image analysis also helps to capture malware's hooks and malicious code injections. Malware behavioural artefacts that can be extracted from the memory dump of the infected system can be seen from Table 4 and a graphical representation of memory image analysis is presented in Fig. 2(b).

4.6. Benign and malware dataset sources

Behavioural features extracted from benign and malware PE are needed in order to build a dataset that represents normal and malicious activities. This is because the detection model must learn to detect and distinguish benign from malware file. There exist various benign samples and details on their sources can be viewed from [10,58]. On the other hand, knowing where to get malware PE samples helps both researchers and security analysts to mitigate malware attacks. Malware repositories include but not limited to VirusShare [58,59], VirusTotal [10,58], Vxheaven [60,61], CA Technologies [7], Virussign [62], Honeypot [60], Kafan Forum [63], Danny Quist [64], Mal-API-2019 [65], Sami Malware Group [62,66], Contagio [58], AMP ThreatGrid [67], Malicia-project [10,68] and VET Zoo [7,61].

4.7. Feature selection techniques

Feature selection (FS) is one of the critical steps in the design and implementation of malware detection models [7]. Given a feature vector V of behavioural features x with $V = x_1, x_2, x_3, x_4, \ldots, x_n$ extracted from malware and benign executables during behaviour analysis with some of the features being significant, less significant or useless, knowing which feature to be employed to train a malware detection/classification model is critical [7,69]. A typical feature selection technique selects the most relevant subset of input features from the entire dataset to be used while building the detection model. Dimensionality reduction is another technique that helps exclude some variables/features from the dataset by projecting data distribution into a lower-dimensional space that generates relevant features for building the detection model [58].

The existing feature selection methods include statistical-based methods such as Chi-square [70], Correlation-based Feature Selection [71], Information Gain (IG) [59], Shannon entropy [58], Information Gain Ratio (IGR) [13], Unbalanced Data Feature Selection approach [71], Hierarchical Feature Selection (HFS) and Max-Relevance Algorithm (MRA) [13]. Fisher Score [7], Relief [7] and Minimum Redundancy [7,72] are also statistical-based methods that select a subset of features based on ranking scores. Nevertheless, the selected subset of features may not always result in a good performance of the detection model [7,73]. There are also statistical hybrid feature selection techniques that utilise a combination of more than one feature selection techniques [7]. There exist also dimensionality reduction techniques such as truncated singular value decomposition (truncated SVD) [58], principal component analysis (PCA) [48,70] and singular value decomposition (SVD) [58]. Document Frequency (DF) [13] and Term Frequency–Inverse Document Frequency (TF–IDF) [13,74] are Natural language processing (NLP)-based methods which have also been employed to select relevant features while building a malware detection approach.

5. Machine learning and deep learning techniques

This section elaborates various machine learning and deep learning techniques for behaviour-based malware detection.

5.1. Machine learning techniques

Machine learning algorithms [2,74] have shown promising results in detecting malware attacks. Different machine learning techniques have been presented in previous studies. Nevertheless, most of these studies did not provide a detailed taxonomy of different machine learning techniques specifically applied for behaviour-based malware detection. In this way, the following are Ml techniques grouped based on their learning similarities.

Instance-based Learning Technique–Instance-based (IB) learning (also called lazy-learning or memory-based learning) technique is one of the categories of supervised learning classifiers that perform classification based on some similarity measures. K-nearest neighbour (KNN) [75], Self-organising map (SOM)

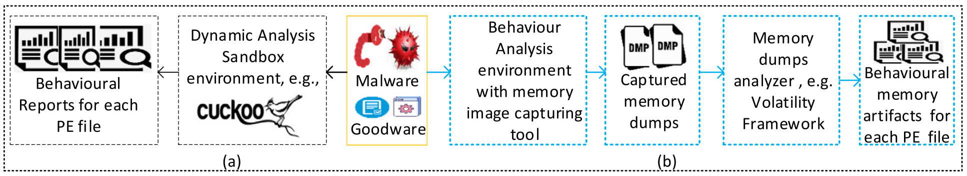
Behavioural Reports for each PE file
Dynamic Analysis Sandbox environment, e.g.,
cuckoo
Malware
Goodware
(a)
Behaviour Analysis environment with memory image capturing tool
DMP
DMP
Captured memory dumps
Memory dumps analyzer, e.g. Volatility Framework
Behavioural memory artifacts for each PE file
(b)

Fig. 2. A typical workflow of (a) Dynamic malware analysis (b) Malware memory image analysis.

[76,77] and support vector machine (SVM) [60,65,78] are families of instance-based learning techniques.

Decision Tree Technique–A Decision Tree (DT) is another variant of the supervised learning technique. Decision tree classification first represents the whole dataset of benign and malware behavioural features as the root node. In the second step, the classifier determines the most relevant features from the dataset using some feature selection techniques such as Information Gain [59]. The third step involves dividing the whole dataset into subsets containing the most relevant attributes and values. In the fourth step, the DT classifier generates internal nodes (decision tree nodes) that contain the best features. In the fifth step (considered the last step), the classifier recursively creates decision trees using the most relevant subset (created in step 3). The same process continues until no further node can be classified, and the last node is known as a leaf node, which denotes the predicted outcome of a given instance which could be benign or malware.

J48 (also called C4.5 decision tree), logistic model tree (LMT), iterative dichotomiser 3 (ID3), and classification and regression tree (CART) are some of the categories of DT classifiers. J48 was applied to build a supervised learning model that distinguishes malware from benign files using the sequence of API calls [59]. Another J48 for malware detection based on dynamic features was proposed in [79]. The LMT [80] is a supervised learning classifier that is implemented with a combination of logistic regression and decision tree classifier. For example, in [80], the LMT was trained to detect malware based on API calls features, and its performance outdid other classifiers such as Naïve Bayes, KNN, and SVM with the accuracy of 98.2857%. CART is another DT-based learning algorithm represented by a binary tree that makes binary classification straightforward [74]. Iterative Dichotomiser 3 (ID3) learns from the dataset to build a decision tree by iteratively dividing features into two or more than two groups at each iteration step. ID3 was used to detect rootkits malware [81].

Regression Technique– Regression classifiers build the detection model based on the input features from the training dataset and utilises the trained model to predict the class of a given new instance. A regression classifier's output depends on what the classifier has learned during the training phase. Logistic regression (LR)/binary logistic regression is a family of regression classifiers implemented for malware detection and is the most commonly used regression classifier [82]. Multinomial Logistic Regression (MLR) is another family of regression classifiers. However, the MLR classifier is used for multi-class classification problems where the classifier's prediction depends on multiple features of the training set [82,83]. The MLR has more than two possible predictions for the target variable, and it is mostly applied when predicting variants of a particular malware executable.

Probabilistic Technique–This type of technique predicts the label/class of unknown samples based on conditional probabilities where the algorithm gives the probability of distribution as a solution [78]. Techniques such as Naïve Bayes (NB), Gaussian Naïve Bayes (GNB), Bernoulli Naïve Bayes (BNB), Multinomial Naïve Bayes (MNB), Bayesian Belief Network (BBN), Bayesian-Logistic Regression (BLR) and Hidden Markov Model (HMM) are examples of this category.

Naïve Bayes is the basic supervised machine learning classifier that operates based on the Bayesian theorem with the "naïve" principle/assumption where every pair of features to be classified is conditionally independent of each other. Given the class attribute's value, Naïve Bayes is built on the assumption that each feature on the dataset makes an independent and equal contribution to the classification outcome [84]. Gaussian Naïve Bayes, Bernoulli Naïve Bayes and Multinomial Naïve Bayes are three variants of Naïve Bayes classifiers that have also been applied on malware detection [85,86].

Being a probabilistic graphical classifier, Bayesian Belief Network/Bayesian network (BN) uses a directed acyclic graph (DAG) to represent a set of malware and benign features, and their conditional dependencies [87,88]. Bayesian-Logistic Regression (BLR) determines if a given event is malicious or benign based on probability [89,90]. The Hidden Markov Model (HMM) is another powerful type of probabilistic classifier applied for behaviour-based malware detection due to its ability to discover hidden structure in a given sequential data [59].

Clustering Technique–Clustering is an unsupervised learning technique that learns from an unstructured dataset. K-means, K-medoids, hierarchical clustering (HC), expectation–maximisation (EM), mean-shift clustering (MSC) and density-based spatial clustering of applications with noise (DBSCAN) are the well-known clustering techniques. K-means is a partitioning technique that produces tighter and efficient/robust clusters [91]. Given several observations n, K-means aims at partitioning them into k set of clusters C (with $C = c_1, c_2, c_3, \ldots.., c_k$, where c_k denotes the last cluster) based on some similarity measures with each observation belonging to the cluster with the closest mean (which serves as a prototype of the cluster). Such similarity between observations can be determined based on some distance measures between observations [91,92].

K-medoids is a variant of partitioning clustering technique, which can efficiently produce tighter clusters on a given set of unlabelled observations [93]. Hierarchical clustering/hierarchical cluster analysis (HCA) is also an unsupervised learning technique that groups given observations or objects into a hierarchy of clusters. Agglomerative and divisive are two categories of hierarchical clustering. Agglomerative follows a bottom-up approach where at the beginning, each observation is assigned to its cluster, and as one moves up, pairs of clusters are merged. In contrast to agglomerative, divisive follows a top-down approach where at the beginning all observations are assigned into one cluster, and as one moves down the hierarchy, cluster splits is performed recursively [91,94]. With EM, the distance between observations is measured based on probability distributions [95]. MSC and DBSCAN have been implemented for malware clustering in [96].

Artificial Neural Network (ANN) Technique–The ANN (also known as neural network) is a type of machine learning algorithm that discovers the underlying relationship between observations in a given dataset by adopting the process that imitates how the human brain works. A typical structure of an ANN model is made up of a set of connected input, hidden and output nodes/neurons that interact. In case of malware binary classification, such output could be malware or benign [97]. Multilayer Perceptron (MLP) [98], Back-Propagation (BP) [99], Stochastic and Gradient, Descent

(SGD) [49] are various ANN techniques, and they have been applied to detect malware.

Ensemble Learning Technique–Ensemble Technique is a category of machine learning (supervised or unsupervised) that combines the potential/merits of different machine learning techniques and various feature sets (known as boosting). Boosting is not a specific learning algorithm, but instead, it is a generic algorithm that intends to combine several weak algorithms (e.g., KNN, SVM, Naïve Bayes, etc.) to improve their learning process.

AdaBoost (also known as discrete AdaBoost) is one of the boosting algorithms for solving classification problems. The weakness of a specific learning algorithm is determined by the error rate computed at each iteration and AdaBoost finds miss-classified observations, increases their weights for the next algorithm to pay attention to them and improves the classification outcome. The same process continues until the necessary results are achieved, and no more classifier can be added. A Dynamic malware detection based on AdaBoost was proposed in [58,100].

Bagging is an ensemble algorithm based on a combination of more than one algorithms to improve the overall prediction outcome. This algorithm splits the training set into various training samples and then builds a classifier for each training sample, resulting in multiple classifiers learning from different training samples originated from one original training set. The predicted outcomes of all classifiers are combined by computing average or majority voting [101]. A bagged-J48 was presented in [102].

Random Forest is a supervised learning algorithm that learns from a given training dataset while building an ensemble of decision trees. It relies on bagging mechanisms while learning from the dataset. Therefore, Random Forest builds an ensemble of multiple learning decision trees whose predictions are merged to improve the overall performance [113]. In [114], the API call sequences extracted during ransomware execution were used to implement Random Forest, and AdaBoost detection approaches. Stacking is another ensemble technique [101].

Gradient Boosting Machines (GBM) is an ensemble algorithm used for classification and regression. It is based on combining models from different learning algorithms to generate new iterative one. The algorithm initially started with AdaBoost, which later resulted in many ensemble techniques such as GBM, Model-Based Boosting (MBoost), and after that to CatBoost, XGBoost and LightGBM [119]. A lightGBM-based model for malware detection was proposed in [119].

Other machine learning techniques–There exists other categories of machine learning for behaviour-based malware detection including inductive rule learner (e.g., RIPPER) [106], reinforcement learning [38], similarity-based and threshold-matching [45], Fuzzy clustering approach [120] and evolutionary techniques [116].

5.2. Deep learning techniques

Deep learning (DL) is a class of neural networks algorithm which is implemented with multiple hidden layers. Hidden layers enable the model to learn abstractions of benign and malware features. Deep learning models can handle big datasets with high dimensions and perform automatic extraction and selection of high-level abstract features (the process does not require human expertise) [121] and their performance is relatively high compared to the traditional machine learning models. Besides, data labelling is not needed for training DL models as they are capable of learning from both labelled and non-labelled data and produce accurate results with low false-positive [122]. Therefore, in contrast to traditional supervised machine learning techniques that highly depend on human efforts and domain experts to perform feature extraction and feature selection [123], and prone to high false positive rate (FPR) in addition to their naïve assumptions to generate relevant features [124], deep learning models are dominating and have attracted researchers due to their potential to overcome traditional machine leanring drawbacks. Convolutional neural networks (CNNs) [8], recurrent neural networks (RNNs) [118,125], long short-term memory networks (LSTMs) [109], stacked auto-encoders (SAEs) [126], deep feedforward neural networks (DFNNs), also known as feedforward networks (FNs) [108], restricted Boltzmann Machines (RBMs) [127], and deep belief networks (DBNs) [128] are various deep learning techniques for behaviour-based malware detection. These techniques were also implemented in [129–131].

Tables 5 and 6 present comparisons of techniques for behaviour-based malware detection grouped by categories of machine learning and deep learning techniques. In addition, it is crucial to note that in Table 6, we use Ref, ML, DL, IB, DT, Prob, ANN, Ensem, Regr to refer to the reference, machine learning, deep learnng, instance-based, decision trees, probabilistic, ensemble, and regression, respectively.

6. Mobile malware detection

The widespread use of smartphones has accelerated the development of new mobile applications. Developed and released by Google, Android is currently the leading and most popular mobile OS with a global market share of 71.93% [132]. Android is an open-source OS (developed based an open Linux Kernel) that gives developers the opportunity to deploy new applications as well as customising the existing applications. This openness and popularity of the Android OS has made it the target for cyber attackers [133]. As reported in [134], new samples of Android malware reached 482,579 per month, with Android devices mostly being affected by Trojans [3]. Dynamic analysis is the current effective approach to detect zero-day malware attacks in the android systems/devices [135]. Android malware samples/datasets are collected from various sources such as VirusShare [136], AndroZoo [136] and Contagio mobile dataset [137]. Behavioural features can be extracted from Android application packages (Android file with .apk extension) through sandbox environment such as Droidbox [135]. Other analysis environments include emulators such as Nexus One [138], or Dynalog automated framework which is an emulator-based sandbox [139]. The behavioural characteristics of each apk file are monitored during the execution and the output is a behavioural report that can reveal what the malware has performed in the Android device. To address the prevalent complex challenge/problem of Android-based mobile malware attacks, many detection approaches have been implemented with machine learning and deep learning algorithms [138,140]. Table 7 summarises some of the existing behaviour-based Android malware detection frameworks and Fig. 3 depicts a graphical representation of Android malware behaviour analysis.

7. Evaluation metrics for malware detection models

Researchers' evaluation metrics to evaluate the performance of malware detection models and their effectiveness are elucidated in this section. A malware detection model's effectiveness is determined based on the evaluation metric, which is used [157]. Thus, the performance is evaluated based on several evaluation metrics that reveal how well a given malware detection model performs while detecting/classifying known and unknown malicious software. Evaluation metrics include True positive (TP), which refers to PE malicious software samples that are detected and correctly classified as malicious software. When true positive is higher, the model's performance is better. False-positive (FP) indicates PE benign samples which are incorrectly identified as

Table 5
Existing malware detection techniques based on API call features extracted from PE file.

Ref	No. Benign samples	No. Malware samples	API calls features	Behaviour-based detection	Machine learning categories								ML & DL techniques
					Instance-based	Decision trees	Probabilistic	Clustering	Artificial neural networks	Deep learning	Ensemble	Regression	
[103]	3620	16 243	✓	✓	✓								
[102]	1358	1354	✓	✓	✓								KNN
[104]	6434	8634	✓	✓	✓								
[105]	456	1368	✓	✓	✓								
[102]	1358	1354	✓	✓	✓								
[104]	6434	8634	✓	✓	✓								
[106]	100	416	✓	✓	✓								
[60]	760	3490	✓	✓	✓								SVM
[7]	1500	2000	✓	✓	✓								
[105]	456	1368	✓	✓	✓								
[107]	40	60	✓	✓	✓								
[106]	100	416	✓	✓		✓							J48
[80]	–	–	✓	✓		✓							LMT
[106]	100	416	✓	✓			✓						Naïve Bayes
[76]	3620	16 243	✓	✓			✓						Naïve Bayes
[59]	42	648	✓	✓			✓						Sub-curve HMM
[10]	21 116	23 080	✓	✓			✓						Markov Chain
[108]	–	4753	✓	✓			✓						HMM
[98]	68	81	✓	✓				✓					EM
[92]	–	1124	✓	✓				✓					
[92]	–	1124	✓	✓				✓					K-means
[92]	–	1124	✓	✓				✓					Hierarchical clustering
[92]	–	1124	✓	✓				✓					S-DBSCAN
[91]	–	54 628	✓	✓				✓					Agglomerative clustering
[65]	–	1500	✓	✓					✓				MLP
[98]	68	81	✓	✓					✓				
[109]	1352	1430	✓	✓						✓			Bi-R-LSTM
[108]	–	4753	✓	✓						✓			DFN
[108]	–	4753	✓	✓						✓			LSTM
[110]	7860	13 518	✓	✓						✓			B-LSTM
[59]	42	648	✓	✓							✓		
[111]	2951	31,869	✓	✓							✓		
[112]	306	806	✓	✓							✓		RF
[105]	456	1368	✓	✓							✓		
[103]	3620	16 243	✓	✓							✓		
[98]	68	81	✓	✓							✓		LogitBoost
[103]	3620	16 243	✓	✓							✓		Bagging
[91]	–	54 628	✓	✓							✓		Clustering ensemble
[108]	–	4753	✓	✓							✓		Stacked CNN+LSTM
[102]	1358	1354	✓	✓								✓	LR

malicious samples, true negative (TN) represents PE benign samples which are correctly classified as benign. In contrast, false negative (FN) indicates PE malicious samples which are incorrectly classified as benign. Other important metrics include true positive rate (TPR) [5], true negative rate (TNR) [5], false Positive rate (FPR) [64], false-negative rate (FNR) [10], accuracy (ACC) [110], Precision (P) [64], Recall (R) [64], F1-Score [], F-measure [64], and Area under the roc curve (AUR) [5].

Table 6
Existing malware detection techniques based on multi-behavioural features extracted from PE file.

Ref	No. Benign samples	No. Malware samples	Extracted behavioural features												ML techniques	ML categories
			Registry operations	File system changes	web browsing history	Network operations	API call sequences	API return values	Processes	Threads	Loaded DLLs	DLL function calls	values of register	Mutexes		
[58]	8422	16 489	✓	✓	✓	✓									SVM	IB
[66]	1359	3009					✓	✓							SMO	
[115]	1000	1000	✓	✓	✓	✓			✓	✓					KNN and SVM	
[116]	942	582	✓	✓			✓								SVC (SVM)	
[117]	742	673	✓	✓			✓								SVM	DT
[66]	1359	3009					✓	✓							J48	
[115]	1000	1000	✓	✓	✓	✓			✓	✓					J48	
[46]	385	826					✓				✓		✓		J48	
[79]	521	582	✓	✓			✓				✓				J48	
[66]	1359	3009					✓	✓							BLR	Prob
[115]	1000	1000	✓	✓	✓	✓			✓	✓					BN and NB	
[46]	385	826					✓				✓		✓		NB	
[79]	582	521	✓	✓			✓				✓				NB	
[86]	1001	3,265									✓	✓			NB and MNB	
[63]	–	13 600	✓	✓		✓									BP	ANN
[48]	–	1000				✓									NN	
[117]	742	673	✓	✓			✓								ANN	
[46]	385	826					✓				✓		✓		RF	Ensem
[58]	8422	16 489	✓	✓	✓	✓									RF, GB and AdaBoost	
[39]	–	42,000	✓	✓		✓	✓							✓	RF	
[67]	86 707	143 684	✓	✓		✓								✓	RF	
[100]	800	2200	✓	✓		✓	✓								RF, AdaBoost and Bagging	
[118]	13 760	13 760	✓	✓			✓				✓			✓	Stacking	
[100]	800	2200	✓	✓		✓	✓								LR	Regr
[116]	942	582	✓	✓			✓								RLR	

8. Current challenges and future directions

This section discusses some of the current research challenges and future directions in behaviour-based malware detection techniques.

8.1. Sensitive data exfiltration

Data exfiltration cyber-attacks are increasingly becoming more sophisticated due to the sophistication and technological advancement of cybercriminals' attack vectors over the last decades. Data exfiltration usually occurs when a malware program or other internal/external malicious agent performs unauthorised data access by copying or retrieving data from a computer and transferring the stolen sensitive data out of the organisational network into the hands of cybercriminals. These sophisticated cybercriminals' attacks can often remain undetected in the enterprise networks for a remarkable amount of time, even while actively hunting for sensitive enterprise data. Accordingly, the security report released by McAfee [158] has revealed that data exfiltration has had a negative financial impact on various institutions. The existing malware detection techniques cannot detect advanced data exfiltration incidents, especially the ones performed by internal cyber attackers [159]. In addition, malware with data exfiltration behaviours will continue to evolve at a larger scale and therefore, posing a very serious security challenge to both organisations and individual businesses [160].

8.2. Detection of new malware variants

Recently, malware developers are using very sophisticated techniques to develop new malware variants which can evade detection. Evasion technique refers to one of the malware's characteristics which gives them the ability to hide some part of the program codes to avoid execution once loaded in a hostile environment [9,161]. In this way, signature-based detection cannot handle new malware attacks because they fully rely on a database of static signatures and are most prone to various evasion techniques [9].

Malware detection based on dynamic analysis is a preferable technique with the ability to detect both existing and new

Table 7
Summary of existing behaviour-based Android malware detection frameworks.

Ref	Behavioural features	Detection techniques	ML	DL
Mal-warehouse [141]	Systems resources consumption such as CPU, memory, battery and network usage, process reports	KNN, RF, SVM, NB, and AdaBoost	✓	
Wang et al. [142]	Network traffic operations	C4.5	✓	
Ye et al. [143]	Events of API calls	CNNs		✓
DroidCat [138]	Method calls and inter-component communication (ICC) of intents	RF, KNN, DT, NB and SVM	✓	
SAMADroid [144]	System calls	RF, DT and NB	✓	
DeepAMD [145]	API calls, process logs, log files and battery states	Deep ANN and ML techniques	✓	✓
da Costa et al. [146]	System calls	RF and SVM	✓	
Mobile-Sandbox [147]	API calls	SVM	✓	
Droidward [148]	API calls, network packet size and event flows,	SVM, DTree, and RF	✓	
Yu et al. [149]	System calls	SVM and NB	✓	
CANDYMAN [135]	File access operations, file manipulation, network connections, malicious sms, and API calls	CNN, RNN, LSTM, RF, DTree, Bagging, KNN and SVM	✓	✓
Deep4MalDroid [150]	System calls	Stack autoencoder (SAEs)		✓
DL-Droid [151]	API calls and intents (actions/events)	DL implemented with H2O		✓
Martinelli et al. [152]	System calls	CNNs		✓
DroidDivesDeep [153]	Network data, sensor data, memory and CPU usage	DNNs with Dropout		✓
EnDroid [154]	System call sequences and application level behaviours such as premium service subscription, personal information stealing, etc.	Stacking		✓
Crowdroid [155]	System calls	K-means	✓	
DroidDolphin [156]	API calls	SVM	✓	

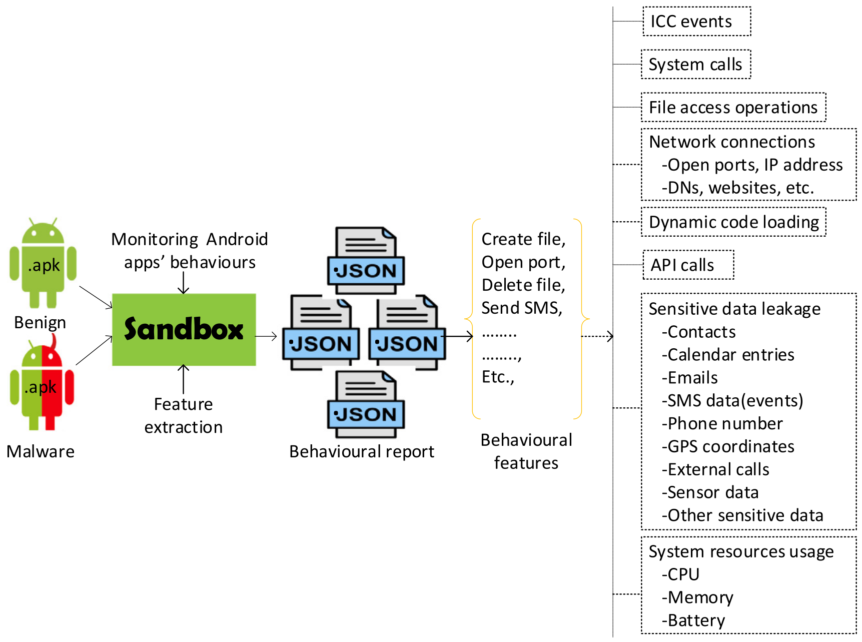
.apk
Benign
.apk
Malware
Monitoring Android apps' behaviours
Sandbox
Feature extraction
JSON
Behavioural report
Create file, Open port, Delete file, Send SMS, ..., Etc.,
Behavioural features
ICC events
System calls
File access operations
Network connections -Open ports, IP address -DNs, websites, etc.
Dynamic code loading
API calls
Sensitive data leakage -Contacts -Calendar entries -Emails -SMS data(events) -Phone number -GPS coordinates -External calls -Sensor data -Other sensitive data
System resources usage -CPU -Memory -Battery

Fig. 3. A graphical representation of Android malware behaviour analysis.

malware variants despite malware evasion which can fool the analysis in some instances [162]. On the other hand, memory analysis-based detection is another alternative for detecting existing and zero-day malware activities. Nevertheless, having automated analysis sandboxes/tools for malware behaviour analysis that cannot be evaded by advanced malware variants is still a matter of the utmost concern. This is because such a sophisticated analysis environment will help to identify and gather any behavioural characteristics that could be used to represent the malware sample which will result in a good dataset for training and testing ML or DL models. More significantly, malware hybrid analysis frameworks would yield better performance over each of these methods.

8.3. Lack of public labelled and benchmark datasets

Generating a significant representative feature set from many unknown portable executable samples is one of the most challenging steps while solving malware detection problems with machine learning and deep learning. Generally, machine learning and deep learning models become sophisticated based on the dataset, which is used to train them. Thus, having a reliable labelled dataset that represents all samples under analysis is more valuable in malware detection [163]. Unfortunately, as the number of malware samples increases at a considerable scale, the manual labelling process becomes time-consuming and unscalable, especially when dealing with supervised learning problems. Therefore, it draws the attention to develop automated labelling frameworks that can perform automatic extraction and labelling of malware's features.

On the other hand, any proposed behaviour-based malware detection model is expected to yield a high detection accuracy. For instance, the malware detection models in [67,107,110] have achieved the accuracy of 97.85%, 99.98% and 95.4%, respectively. Nevertheless, we cannot conclude that one of these three models is better than the others as they have been trained and tested on different datasets. One of the main reasons is that many samples are required for the experiment, and the dataset's composition is rarely public. As a result, a few categories of malware datasets are mostly used [164]. Antivirus companies use the first category of malware samples to test their antivirus products, and they are used to represents modern malware. Unfortunately, these malware samples are not made public for academic researchers. Another category used for research purposes includes the collection of malware sample's repositories available online (see Section 4.6). Nonetheless, it is still challenging to generate a ground truth/benchmark dataset from this large number of samples. Hence, the lack of standard public benchmark malware datasets impedes the comparisons of various malware detection models. As a result, having a standard public benchmark dataset that represents behaviours of different malware variants and benign samples is required to evaluate different behaviour-based malware detection models' performance. To maintain the efficiency of this dataset, regular updates will be required to include behavioural features from new malware variants.

8.4. Adversarial machine learning

The wide deployment of machine learning models has, at the same time drawing the attention of many adversaries who intend to defeat their performance or cause misclassification. An adversary modifies malware executable samples and makes them appear as benign while still having their malicious behaviours and can then be used to defeat a particular existing detection model. This situation where an adversary uses manipulated input samples/features to defeat or fool the performance of a trained ML model is known as "adversarial machine learning", and this type of cyber attack represents a serious threat to malware detection models based on machine learning [165]. Moreover, it is important to note that adversarial samples can defeat traditional machine learning and deep learning techniques because these techniques are not designed to work with adversarial samples [166]. For example, the deep learning model for detecting malware implemented in [167] was defeated by an intelligent adversarial evasion technique proposed in [168] whereby after applying the modified adversarial malware samples the accuracy of the model decreased from 94% to nearly 50%. Data Poisoning [169,170], manipulation of samples' labels [171], model extraction/stealing [172] and model privacy violation [173] are some of the forms of adversarial learning attack vectors that are employed to defeat machine learning and deep learning models. Hence, training robust malware detection and classification models with the ability to detect potentially manipulated adversarial samples remains a serious challenge that needs to be addressed in the future.

8.5. Interpretability of the detection models

As the application of machine learning and deep learning models continues to become ubiquitous in various domains, their interpretability has also become a new open challenge [174,175]. Most of these models are treated as black-boxes that receive input features/observations and generate output through a series of operations that are not easily understandable to humans. Such a way of making a decision is considered obscure, and this could arouse a problem especially in malware detection models as when a false positive alert is triggered, security analysts/end-users would like to understand why such a false alert occurs. A clear interpretation of the results is essential for humans to examine why and how the detection model classifies such a given software file as malware or benign. While many of the existing work has mainly focused on enhancing the models' detection accuracy and runtime, there has been a slight focus on the interpretability of the models' classification results. This has definitely become a serious concern when the latest work demonstrates that sometimes the machine learning models' decisions are biased and inconsistent [174,176]. To date, only a few works on malware detection such as the ones conducted in [177,178] has focused on the interpretability of the proposed machine learning models. Therefore, there is a major research gap in interpretability of machine learning and deep learning models.

9. Conclusion

This paper presented a survey on behaviour-based malware detection. The snowball approach was employed to select a set of most relevant papers which are reviewed throughout this work, and papers were selected from the most well-known databases in computer science and information technology domains. This study has presented malware categories, taxonomy of malware analysis testbeds, including their merits and demerits. It also provides comprehensive coverage of tools for performing dynamic analysis and memory analysis, and a collection of malware and benign dataset repositories, feature selection, machine learning and deep learning techniques. A comprehensive analysis of various performance evaluation metrics and the comparison of behaviour-based malware detection techniques were also presented based on the categories of machine learning and deep learning techniques. This study included current research challenges and issues in behaviour-based malware detection. The emerging challenges such as data exfiltration, detection of new malware variants, adversarial machine learning, lack of public and benchmark and labelled dataset and interpretability of machine learning algorithms were elaborated. This paper is mainly focused on behaviour-based malware detection in Windows and Android OS. However, in future, we want to survey other major and popular desktop and mobile operating systems, such as Linux and iOS.

CRediT authorship contribution statement

Pascal Maniriho: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Abdun Naser Mahmood:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition. **Mohammad Jabed Morshed Chowdhury:** Conceptualization, Writing – review & editing, Supervision.

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.

Acknowledgement

All authors have checked the revised manuscript and agreed to the resubmission. We look forward to your favourable consideration for publication in Future Generation Computer Systems Journal.

References

[1] S.S. 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.

[2] 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) http://dx.doi.org/10.1016/j.jnca.2019.102526.

[3] AV-TEST, Security Report: Facts and Figures, Tech. rep., 2020.

[4] AviraT, Q4 and 2020 Malware Threat Report, Tech. rep., 2021.

[5] A. Cohen, N. Nissim, Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory, Expert Syst. Appl. 102 (2018) 1158–1178, http://dx.doi.org/10.1016/j.eswa.2018.02.039.

[6] A.O.A. El-Mal, M.A. Sobh, A.M. Eldin, Hard-Detours: A new technique for dynamic code analysis, in: IEEE EuroCon 2013, 2013, pp. 46–51, http://dx.doi.org/10.1109/EUROCON.2013.6624964.

[7] S. Huda, R. Islam, J. Abawajy, J. Yearwood, M.M. Hassan, G. Fortino, A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection, Future Gener. Comput. Syst. 83 (2018) 193–207, http://dx.doi.org/10.1016/j.future.2017.12.037.

[8] D. Nahmias, A. Cohen, N. Nissim, Y. Elovici, Deep feature transfer learning for trusted and automated malware signature generation in private cloud environments, Neural Netw. 124 (2020) 243–257, http://dx.doi.org/10.1016/j.neunet.2020.01.003.

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

[10] E. Amer, I. Zelinka, A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence, Comput. Secur. 92 (2020) http://dx.doi.org/10.1016/j.cose.2020.101760.

[11] M. Murthaja, B. Sahayanathan, A. Munasinghe, D. Uthayakumar, L. Rupasinghe, A. Senarathne, An automated tool for memory forensics, in: 2019 International Conference on Advancements in Computing (ICAC), IEEE, 2019, pp. 1–6, http://dx.doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.365.

[12] N. Nissim, O. Lahav, A. Cohen, Y. Elovici, L. Rokach, Volatile memory analysis using the MinHash method for efficient and secured detection of malware in private cloud, Comput. Secur. 87 (2019) http://dx.doi.org/10.1016/j.cose.2019.101590.

[13] Y. Ye, T. Li, D. Adjeroh, S.S. Iyengar, A survey on malware detection using data mining techniques, ACM Comput. Surv. 50 (3) (2017) 1–40, http://dx.doi.org/10.1145/3073559.

[14] K. Shah, D.K. Singh, A survey on data mining approaches for dynamic analysis of malwares, in: Proceedings of the 2015 International Conference on Green Computing and Internet of Things, ICGCIoT 2015, 2015, pp. 495–499, http://dx.doi.org/10.1109/ICGCIoT.2015.7380515.

[15] 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.

[16] D. Deka, N. Sarma, N.J. Panicker, Malware detection vectors and analysis techniques: A brief survey, in: 2016 International Conference on Accessibility to Digital World, ICADW 2016 - Proceedings, 2016, pp. 81–85, http://dx.doi.org/10.1109/ICADW.2016.7942517.

[17] A. Souri, R. Hosseini, A state-of-the-art survey of malware detection approaches using data mining techniques, Hum.-Centric Comput. Inf. Sci. 8 (1) (2018) 533–539, http://dx.doi.org/10.1186/s13673-018-0125-x.

[18] K. Radhakrishnan, R.R. Menon, H.V. Nath, A survey of zero-day malware attacks and its detection methodology, in: TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019, pp. 533–539, http://dx.doi.org/10.1109/TENCON.2019.8929620.

[19] A. Abusitta, M.Q. Li, B.C. Fung, Malware classification and composition analysis: A survey of recent developments, J. Inf. Secur. Appl. 59 (2021) 102828, http://dx.doi.org/10.1016/j.jisa.2021.102828.

[20] B. Yu, Y. Fang, Q. Yang, Y. Tang, L. Liu, A survey of malware behavior description and analysis, Front. Inf. Technol. Electron. Eng. 19 (5) (2018) 583–603, http://dx.doi.org/10.1631/FITEE.1601745.

[21] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, https://doi.org/10.1145/2601248.2601268.

[22] M. Unterkalmsteiner, R. Feldt, T. Gorschek, A taxonomy for requirements engineering and software test alignment, ACM Trans. Softw. Eng. Methodol. (TOSEM) 23 (2) (2014) 1–38, http://dx.doi.org/10.1145/2523088.

[23] Taxonomy | definition of taxonomy by oxford dictionary on lexico.com also meaning of taxonomy, 2021, https://www.lexico.com/definition/taxonomy. (Accessed on 06/07/2021).

[24] R.C. Nickerson, U. Varshney, J. Muntermann, A method for taxonomy development and its application in information systems, Eur. J. Inf. Syst. 22 (3) (2013) 336–359, http://dx.doi.org/10.1057/ejis.2012.26.

[25] A. Vasudevan, R. Yerraballi, SPiKE: Engineering malware analysis tools using unobtrusive binary-instrumentation, in: Proceedings of the 29th Australasian Computer Science Conference, Vol. 48, pp. 311–320. http://dx.doi.org/10.1145/1151699.1151734.

[26] P. Black, I. Gondal, R. Layton, A survey of similarities in banking malware behaviours, Comput. Secur. 77 (2018) 756–772, http://dx.doi.org/10.1016/j.cose.2017.09.013.

[27] A. Case, G.G. Richard, Memory forensics: The path forward, Digit. Investig. 20 (2017) 23–33, http://dx.doi.org/10.1016/j.diin.2016.12.004, Special Issue on Volatile Memory Analysis.

[28] A. Afreen, M. Aslam, S. Ahmed, Analysis of fileless malware and its evasive behavior, in: 2020 International Conference on Cyber Warfare and Security (ICCWS), 2020, pp. 1–8, http://dx.doi.org/10.1109/ICCWS48432.2020.9292376.

[29] W. Han, J. Xue, Y. Wang, Z. Liu, Z. Kong, MalInsight: A systematic profiling based malware detection framework, J. Netw. Comput. Appl. 125 (2019) 236–250, http://dx.doi.org/10.1016/j.jnca.2018.10.022.

[30] D. Kirat, G. Vigna, C. Kruegel, BareBox: Efficient malware analysis on bare-metal, in: ACSAC '11: Proceedings of the 27th Annual Computer Security Applications Conference, 2011, pp. 403–412, http://dx.doi.org/10.1145/2076732.2076790.

[31] O. Or-Meir, N. Nissim, Y. Elovici, L. Rokach, Dynamic malware analysis in the modern era—A state of the art survey, ACM Comput. Surv. 52 (5) (2019) 1–48, http://dx.doi.org/10.1145/3329786.

[32] A. Pektaş, 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.

[33] W. Ahmed, B. Aslam, A comparison of windows physical memory acquisition tools, in: Proceedings - IEEE Military Communications Conference MILCOM, 2015, pp. 1292–1297, http://dx.doi.org/10.1109/MILCOM.2015.7357623.

[34] E. Ukwandu, M.A.B. Farah, H. Hindy, D. Brosset, D. Kavallieros, R. Atkinson, C. Tachtatzis, M. Bures, I. Andonovic, X. Bellekens, A review of cyber-ranges and test-beds: Current and future trends, 2020, arXiv preprint arXiv:2010.06850.

[35] J. Davis, S. Magrath, Technical Report: A Survey of Cyber Ranges and Testbeds, Tech. rep., 2013, p. 29.

[36] F. Biondi, T. Given-Wilson, A. Legay, C. Puodzius, J. Quilbeuf, Tutorial: An overview of malware detection and evasion techniques, in: International Symposium on Leveraging Applications of Formal Methods, Springer, 2018, pp. 565–586, http://dx.doi.org/10.1007/978-3-030-03418-4_34.

[37] V. Ndatinya, Z. Xiao, V.R. Manepalli, K. Meng, Y. Xiao, Network forensics analysis using wireshark, Int. J. Secur. Netw. 10 (2) (2015) 91–106, http://dx.doi.org/10.1504/IJSN.2015.070421.

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

[39] R.S. Pirscoveanu, S.S. Hansen, T.M.T. Larsen, M. Stevanovic, J.M. Pedersen, A. Czech, Analysis of malware behavior: Type classification using machine learning, in: 2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), 2015, http://dx.doi.org/10.1109/CyberSA.2015.7166115.

[40] 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.

[41] N. Hoque, M.H. Bhuyan, R.C. Baishya, D.K. Bhattacharyya, J.K. Kalita, Network attacks: Taxonomy, tools and systems, J. Netw. Comput. Appl. 40 (2014) 307–324, http://dx.doi.org/10.1016/j.jnca.2013.08.001.

[42] E. Gandotra, D. Bansal, S. Sofat, Malware analysis and classification: A survey, J. Inf. Secur. 2014 (2014) http://dx.doi.org/10.4236/jis.2014.52006.

[43] A. Case, G.G. Richard III, Memory forensics: The path forward, Digit. Investig. 20 (2017) 23–33, http://dx.doi.org/10.1016/j.diin.2016.12.004.

[44] I. Korkin, I. Nesterov, Applying memory forensics to rootkit detection, 2015, arXiv preprint arXiv:1506.04129.

[45] 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.

[46] Z. Salehi, A. Sami, M. Ghiasi, Using feature generation from API calls for malware detection, Comput. Fraud Secur. 2014 (9) (2014) 9–18, http://dx.doi.org/10.1016/S1361-3723(14)70531-7.

[47] M. Sebastián, R. Rivera, P. Kotzias, J. Caballero, Avclass: A tool for massive malware labeling, in: International Symposium on Research in Attacks, Intrusions, and Defenses, 2016, pp. 230–253, http://dx.doi.org/10.1007/978-3-319-45719-2_11.

[48] D. Arivudainambi, V.K. KA, P. Visu, et al., 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.

[49] C.W. Kim, Ntmaldetect: A machine learning approach to malware detection using native api system calls, 2018, arXiv preprint arXiv:1802.05412.

[50] P. Gunathilaka, D. Mashima, B. Chen, Softgrid: A software-based smart grid testbed for evaluating substation cybersecurity solutions, in: Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy, 2016, pp. 113–124, http://dx.doi.org/10.1145/2994487.2994494.

[51] V. Sachidananda, J. Toh, S. Siboni, S. Bhairav, A. Shabtai, Y. Elovici, Let the cat out of the bag: A holistic approach towards security analysis of the internet of things, in: IoTPTS 2017 - Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security, 2017, pp. 3–10, http://dx.doi.org/10.1145/3055245.3055251.

[52] M.L. Hale, K. Lotfy, R.F. Gamble, C. Walter, J. Lin, Developing a platform to evaluate and assess the security of wearable devices, Digit. Commun. Netw. 5 (3) (2019) 147–159, http://dx.doi.org/10.1016/j.dcan.2018.10.009.

[53] O.A. Aslan, R. Samet, A comprehensive review on malware detection approaches, IEEE Access 8 (2020) 6249–6271, http://dx.doi.org/10.1109/ACCESS.2019.2963724.

[54] O. Aslan, R. Samet, Investigation of possibilities to detect malware using existing tools, in: 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), 2017, pp. 1277–1284, http://dx.doi.org/10.1109/AICCSA.2017.24.

[55] A. Aljaedi, D. Lindskog, P. Zavarsky, R. Ruhl, F. Almari, Comparative analysis of volatile memory forensics, live response vs. memory imaging, in: 2011 IEEE International Conference on Privacy, Security, Risk and Trust and IEEE International Conference on Social Computing, PASSAT/SocialCom 2011, 2011, pp. 1253–1258, http://dx.doi.org/10.1109/PASSAT/SocialCom.2011.68.

[56] C. Waits, J.A. Akinyele, R. Nolan, L. Rogers, Computer Forensics: Results of Live Response Inquiry vs. Memory Image Analysis, Tech. rep., 2008.

[57] A. Case, L. Marziale, C. Neckar, G.G. Richard, Treasure and tragedy in kmem-cache mining for live forensics investigation, Digit. Investig. 7 (2010) S41–S47, http://dx.doi.org/10.1016/j.diin.2010.05.006.

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

[59] J. Suaboot, Z. Tari, A. Mahmood, A.Y. Zomaya, W. Li, Sub-curve HMM: A malware detection approach based on partial analysis of API call sequences, Comput. Secur. 92 (2020) 1–15, http://dx.doi.org/10.1016/j.cose.2020.101773.

[60] 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.

[61] S. Huda, J. Abawajy, B. Al-Rubaie, L. Pan, M.M. Hassan, Automatic extraction and integration of behavioural indicators of malware for protection of cyber-physical networks, Future Gener. Comput. Syst. 101 (2019) 1247–1258, http://dx.doi.org/10.1016/j.future.2019.07.005.

[62] 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, pp. 554–559, http://dx.doi.org/10.1109/IConAC.2016.7604978.

[63] Z.-P. Pan, C. Feng, C.-J. Tang, Malware classification based on the behavior analysis and back propagation neural network, in: 3rd Annual International Conference on Information Technology and Applications (ITA 2016), Vol. 7, 2016, pp. 1–5, http://dx.doi.org/10.1051/itmconf/20160702001.

[64] K. Sethi, B.K. Tripathy, S.K. Chaudhary, P. Bera, A novel malware analysis for malware detection and classification using machine learning algorithms, in: SIN '17: Proceedings of the 10th International Conference on Security of Information and Networks, 2017, pp. 107–116, http://dx.doi.org/10.1145/3136825.3136883.

[65] N. Asrafi, D.C.T. Lo, R.M. Parizi, Y. Shi, Y.W. Chen, Comparing performance of malware classification on automated stacking, in: ACM SE '20: Proceedings of the 2020 ACM Southeast Conference, 2020, pp. 307–308, http://dx.doi.org/10.1145/3374135.3385316.

[66] Z. Salehi, A. Sami, M. Ghiasi, MAAR: Robust features to detect malicious activity based on API calls, their arguments and return values, Eng. Appl. Artif. Intell. 59 (2017) 93–102, http://dx.doi.org/10.1016/j.neunet.2018.07.011.

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

[68] A. Nappa, M.Z. Rafique, J. Caballero, The MALICIA dataset: identification and analysis of drive-by download operations, Int. J. Inf. Secur. 14 (1) (2015) 15–33, http://dx.doi.org/10.1007/s10207-014-0248-7.

[69] K. Sethi, R. Kumar, L. Sethi, P. Bera, P.K. Patra, A novel machine learning based malware detection and classification framework, in: 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), 2019, pp. 1–4, http://dx.doi.org/10.1109/CyberSecPODS.2019.8885196.

[70] C.T. Dan Lo, P. Ordóñez, C. Cepeda, Feature selection and improving classification performance for malware detection, in: 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), 2016, pp. 560–566, http://dx.doi.org/10.1109/BDCloud-SocialCom-SustainCom.2016.87.

[71] Q. Jiang, X. Zhao, K. Huang, A feature selection method for malware detection, in: Proceeding of the 2011 IEEE International Conference on Information and Automation (ICIA), 2011, pp. 890–895, http://dx.doi.org/10.1109/ICINFA.2011.5949122.

[72] L.T. Vinh, N.D. Thang, Y.K. Lee, An improved maximum relevance and minimum redundancy feature selection algorithm based on normalized mutual information, in: Proceedings - 2010 10th Annual International Symposium on Applications and the Internet, SAINT 2010 (3), 2010, pp. 395–398, http://dx.doi.org/10.1109/TPAMI.2005.159.

[73] P. O'Kane, S. Sezer, K. McLaughlin, E.G. Im, SVM Training phase reduction using dataset feature filtering for malware detection, IEEE Trans. Inf. Forensics Secur. 8 (3) (2013) 500–509, http://dx.doi.org/10.1109/TIFS.2013.2242890.

[74] E.B. Karbab, M. Debbabi, Maldy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports, Digit. Investig. 28 (2019) S77–S87, http://dx.doi.org/10.1016/j.diin.2019.01.017.

[75] D. Xue, J. Li, T. Lv, W. Wu, J. Wang, Malware classification using probability scoring and machine learning, IEEE Access 7 (2019) 91641–91656, http://dx.doi.org/10.1109/ACCESS.2019.2927552.

[76] T. Kohonen, Self-organized formation of topologically correct feature maps, Biol. Cybernet. 43 (1) (1982) 59–69, http://dx.doi.org/10.1007/BF00337288.

[77] D.C. Le, A.N. Zincir-Heywood, M.I. Heywood, Unsupervised monitoring of network and service behaviour using self organizing maps, J. Cyber Secur. Mobil. 8 (1) (2019) 15–51, http://dx.doi.org/10.13052/jcsm2245-1439.812.

[78] J. Moubarak, T. Feghali, Comparing machine learning techniques for malware detection, in: Proceedings of the 6th International Conference on Information Systems Security and Privacy, 2020, pp. 844–851, http://dx.doi.org/10.5220/0009373708440851.

[79] R.R. Ravula, K.J. Liszka, C.C. Chan, Learning attack features from static and dynamic analysis of malware, Commun. Comput. Inf. Sci. 348 (2013) 109–125, http://dx.doi.org/10.1007/978-3-642-37186-8_7.

[80] A. Dhammi, M. Singh, Behavior analysis of malware using machine learning, in: 2015 Eighth International Conference on Contemporary Computing (IC3), 2015, pp. 481–486, http://dx.doi.org/10.1109/IC3.2015.7346730.

[81] D. Lobo, P. Watters, X.-W. Wu, Identifying rootkit infections using data mining, in: 2010 International Conference on Information Science and Applications, 2010, pp. 1–7, http://dx.doi.org/10.1109/ICISA.2010.5480359.

[82] T. Ghate, C. Pathade, C. Nirhali, K. Patil, N. Korade, Machine learning based malware detection: a boosting methodology, Int. J. Innov. Technol. Explor. Eng. 9 (4) (2020) 2241–2245, http://dx.doi.org/10.35940/ijitee.d1717.029420.

[83] H. Sayadi, H.M. Makrani, S.M.P. Dinakarrao, T. Mohsenin, A. Sasan, S. Rafatirad, H. Homayoun, 2Smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection, in: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2019, pp. 728–733, http://dx.doi.org/10.23919/DATE.2019.8715080.
[84] E.M. Alkhateeb, M. Stamp, A dynamic heuristic method for detecting packed malware using Naive Bayes, in: 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), IEEE, 2019, pp. 1–6, http://dx.doi.org/10.1109/ICECTA48151.2019.8959765.
[85] N. Kawaguchi, K. Omote, Malware function estimation using API in initial behavior, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. (1) (2017) 167–175, http://dx.doi.org/10.1587/transfun.E100.A.1.
[86] M. Schultz, E. Eskin, F. Zadok, S. Stolfo, Data mining methods for detection of new malicious executables, in: Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, 2001, pp. 38–49, http://dx.doi.org/10.1109/SECPRI.2001.924286.
[87] D. Oyen, B. Anderson, K. Sentz, C. Anderson-Cook, Order priors for Bayesian network discovery with an application to malware phylogeny, Stat. Anal. Data Min. 10 (5) (2017) 343–358, http://dx.doi.org/10.1002/sam.11364.
[88] D. Oyen, B. Anderson, C. Anderson-Cook, Bayesian networks with prior knowledge for malware phylogenetics, in: In the AAAI-16 Workshop on Artificial Intelligence and Cybersecurity, 2016, pp. 185–192.
[89] K. Hughes, Y. Qu, A theoretical model: Using logistic regression for malware signature based detection, in: The 10th International Conference on Dependable, Autonomic, and Secure Computing (DASC-2012), 2012.
[90] S.L. Darshan, M.A. Kumara, C.D. Jaidhar, Windows malware detection based on cuckoo sandbox generated report using machine learning algorithm, in: 2016 11th International Conference on Industrial and Information Systems (ICIIS), 2016, pp. 534–539, http://dx.doi.org/10.1109/ICIINFS.2016.8262998.
[91] Y. Zhang, C. Rong, Q. Huang, Y. Wu, Z. Yang, J. Jiang, Based on multi-features and clustering ensemble method for automatic malware categorization, in: 2017 IEEE Trustcom/BigDataSE/ICESS, 2017, pp. 73–82, http://dx.doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.222.
[92] Y. Fang, W. Zhang, B. Li, F. Jing, L. Zhang, Semi-supervised malware clustering based on the weight of bytecode and API, IEEE Access 8 (2020) 2313–2326, http://dx.doi.org/10.1109/ACCESS.2019.2962198.
[93] S. Hou, L. Chen, E. Tas, I. Demihovskiy, Y. Ye, Cluster-oriented ensemble classifiers for intelligent malware detection, in: Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015), Vol. 1, 2015, pp. 189–196, http://dx.doi.org/10.1109/ICOSC.2015.7050805.
[94] O.B. Boţocan, G. Czibula, HACGA: An artifacts-based clustering approach for malware classification, in: 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 2017, pp. 5–12, http://dx.doi.org/10.1109/ICCP.2017.8116976.
[95] S. Pai, F. Di Troia, C.A. Visaggio, T.H. Austin, M. Stamp, Clustering for malware classification, J. Comput. Virol. Hack. Tech. 13 (2) (2017) 95–107, http://dx.doi.org/10.1007/s11416-016-0265-3.
[96] S.M.K. Raza, J. Caballero, Malware traffic classification: Evaluation of algorithms and an automated ground-truth generation pipeline, 2020, arXiv preprint arXiv:2010.11627.
[97] C. Pascariu, I.-D. Barbu, Dynamic analysis of malware using artificial neural networks: Applying machine learning to identify malicious behavior based on parent process hirarchy, in: 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), IEEE, 2017, pp. 1–5, http://dx.doi.org/10.1109/ECAI.2017.8166505.
[98] K.O. Babaagba, S.O. Adesanya, A study on the effect of feature selection on malware analysis using machine learning, in: ICEIT 2019: Proceedings of the 2019 8th International Conference on Educational and Information Technology, 2019, pp. 51–55, http://dx.doi.org/10.1145/3318396.3318448.
[99] L.E. Gonzalez, R.A. Vazquez, Malware classification using euclidean distance and artificial neural networks, in: 2013 12th Mexican International Conference on Artificial Intelligence, 2013, pp. 103–108, http://dx.doi.org/10.1109/MICAI.2013.18.
[100] M. Ijaz, M.H. Durad, M. Ismail, Static and dynamic malware analysis using machine learning, in: 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 2019, pp. 687–691, http://dx.doi.org/10.1109/IBCAST.2019.8667136.
[101] J. Bai, J. Wang, Improving malware detection using multi-view ensemble learning, Secur. Commun. Netw. 9 (17) (2016) 4227–4241, http://dx.doi.org/10.1002/sec.1600.
[102] Y.A. Ahmed, B. Koçer, S. Huda, B.A. Saleh Al-rimy, M.M. Hassan, A system call refinement-based enhanced minimum redundancy maximum relevance method for ransomware early detection, J. Netw. Comput. Appl. 167 (2020) http://dx.doi.org/10.1016/j.jnca.2020.102753.
[103] S. Sheen, A. Yadav, Ransomware detection by mining API call usage, in: 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, pp. 983–987, http://dx.doi.org/10.1109/ICACCI.2018.8554938.
[104] J. Singh, J. Singh, Assessment of supervised machine learning algorithms using dynamic API calls for malware detection, Int. J. Comput. Appl. 29 (3) (2020) 1–8, http://dx.doi.org/10.1080/1206212X.2020.1732641.
[105] R. Tian, R. Islam, L. Batten, S. Versteeg, Differentiating malware from cleanware using behavioural analysis, in: 2010 5th International Conference on Malicious and Unwanted Software, 2010, pp. 23–30, http://dx.doi.org/10.1109/MALWARE.2010.5665796.
[106] F. Ahmed, H. Hameed, M.Z. Shafiq, M. Farooq, Using spatio-temporal information in API calls with machine learning algorithms for malware detection, in: AISec '09: Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence, 2009, pp. 55–62, http://dx.doi.org/10.1145/1654988.1655003.
[107] N. Asrafi, Comparing performances of graph mining algorithms to detect malware, in: ACM SE '19: Proceedings of the 2019 ACM Southeast Conference, 2019, pp. 268–269, http://dx.doi.org/10.1145/3299815.3314485.
[108] B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, Deep learning for classification of malware system call sequences, in: Australasian Joint Conference on Artificial Intelligence, 2016, pp. 137–149, http://dx.doi.org/10.1007/978-3-319-50127-7_11.
[109] 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.
[110] Y. Liu, Y. Wang, A robust malware detection system using deep learning on API calls, in: 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2019), IEEE, 2019, pp. 1456–1460, http://dx.doi.org/10.1109/ITNEC.2019.8728992.
[111] A. Sami, B. Yadegari, H. Rahimi, N. Peiravian, S. Hashemi, A. Hamze, Malware detection based on mining API calls, in: SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing, 2010, pp. 1020–1025, http://dx.doi.org/10.1145/1774088.1774303.
[112] M. Ahmadi, A. Sami, H. Rahimi, B. Yadegari, Malware detection by behavioural sequential patterns, Comput. Fraud Secur. 2013 (8) (2013) 11–19, http://dx.doi.org/10.1016/S1361-3723(13)70072-1.
[113] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32, http://dx.doi.org/10.1023/A:1010933404324.
[114] F. Ullah, Q. Javaid, A. Salam, M. Ahmad, N. Sarwar, D. Shah, M. Abrar, Modified decision tree technique for ransomware detection at runtime through API calls, Sci. Program. 2020 (2020) http://dx.doi.org/10.1155/2020/8845833.
[115] I. Santos, J. Devesa, F. Brezo, J. Nieves, P.G. Bringas, Opem: A static-dynamic approach for machine-learning-based malware detection, in: International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions, 2013, pp. 271–280, http://dx.doi.org/10.1007/978-3-642-33018-6_28.
[116] M.S. Abbasi, H. Al-Sahaf, I. Welch, Particle swarm optimization: A wrapper-based feature selection method for ransomware detection and classification, in: International Conference on the Applications of Evolutionary Computation (Part of EvoStar), 2020, pp. 181–196, http://dx.doi.org/10.1007/978-3-030-43722-0_12.
[117] Y.A. Ahmed, B. Koçer, B.A.S. Al-rimy, Automated analysis approach for the detection of high survivable ransomware, KSII Trans. Internet Inf. Syst. (TIIS) 14 (5) (2020) 2236–2257, http://dx.doi.org/10.3837/tiis.2020.05.021.
[118] C. Jindal, C. Salls, H. Aghakhani, K. Long, C. Kruegel, G. Vigna, Neurlux: dynamic malware analysis without feature engineering, in: ACSAC '19: Proceedings of the 35th Annual Computer Security Applications Conference, 2019, pp. 444–455, http://dx.doi.org/10.1145/3359789.3359835.
[119] M. Al-kasassbeh, M.A. Abbadi, A.M. Al-Bustanji, Lightgbm algorithm for malware detection, in: Science and Information Conference, 2020, pp. 391–403, http://dx.doi.org/10.1007/978-3-030-52243-8_28.
[120] M. Alaeiyan, A. Dehghantanha, T. Dargahi, M. Conti, S. Parsa, A multilabel fuzzy relevance clustering system for malware attack attribution in the edge layer of cyber-physical networks, ACM Trans. Cyber-Phys. Syst. 4 (3) (2020) 1–22, http://dx.doi.org/10.1145/3351881.
[121] S. Yuan, X. Wu, Deep learning for insider threat detection: Review, challenges and opportunities, Comput. Secur. 104 (2021) 102221, http://dx.doi.org/10.1016/j.cose.2021.102221.
[122] M.M. Najafabadi, F. Villanustre, T.M. Khoshgoftaar, N. Seliya, R. Wald, E. Muharemagic, Deep learning applications and challenges in big data analytics, J. Big Data 2 (1) (2015) 1–21, http://dx.doi.org/10.1186/s40537-014-0007-7.
[123] X. Wang, S.M. Yiu, A multi-task learning model for malware classification with useful file access pattern from API call sequence, 2016, arXiv:1610.05945.

[124] R. Pascanu, J.W. Stokes, H. Sanossian, M. Marinescu, A. Thomas, Malware classification with recurrent networks, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 1916–1920, http://dx.doi.org/10.1109/ICASSP.2015.7178304.

[125] P. Dixit, S. Silakari, Deep learning algorithms for cybersecurity applications: A technological and status review, Comput. Sci. Rev. https://doi.org/10.1016/j.cosrev.2020.100317.

[126] M. Sewak, S.K. Sahay, H. Rathore, An investigation of a deep learning based malware detection system, in: ARES 2018: Proceedings of the 13th International Conference on Availability, Reliability and Security, Vol. 26, 2018, pp. 1–5, http://dx.doi.org/10.1145/3230833.3230835.

[127] R. Benchea, D.T. Gavriluţ, Combining restricted boltzmann machine and one side perceptron for malware detection, in: International Conference on Conceptual Structures, 2014, pp. 93–103, http://dx.doi.org/10.1007/978-3-319-08389-6_9.

[128] O.E. David, N.S. Netanyahu, Deepsign: Deep learning for automatic malware signature generation and classification, in: 2015 International Joint Conference on Neural Networks (IJCNN), IEEE, 2015, pp. 1–8, http://dx.doi.org/10.1109/IJCNN.2015.7280815.

[129] A. Pinhero, A. M L, V. P, C. Visaggio, A. N, A. S, A. S, Malware detection employed by visualization and deep neural network, Comput. Secur. 105 (2021) 102247, http://dx.doi.org/10.1016/j.cose.2021.102247.

[130] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, T. Yagi, Malware detection with deep neural network using process behavior, in: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Vol. 2, 2016, pp. 577–582, http://dx.doi.org/10.1109/COMPSAC.2016.151.

[131] R. Vinayakumar, M. Alazab, K.P. Soman, P. Poornachandran, S. Venkatraman, Robust intelligent malware detection using deep learning, IEEE Access 7 (2019) 46717–46738, http://dx.doi.org/10.1109/ACCESS.2019.2906934.

[132] S. o'Dea, Mobile OS market share 2021, 2021, https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/. (Accessed on 06/08/2021).

[133] A. Qamar, A. Karim, V. Chang, Mobile malware attacks: Review, taxonomy & future directions, Future Gener. Comput. Syst. 97 (2019) 887–909, http://dx.doi.org/10.1016/j.future.2019.03.007.

[134] J. Johnson, Global Android malware volume 2020, 2021, https://www.statista.com/statistics/680705/global-android-malware-volume/. (Accessed on 06/08/2021).

[135] A. Martín, V. Rodríguez-Fernández, D. Camacho, CANDYMAN: Classifying android malware families by modelling dynamic traces with Markov chains, Eng. Appl. Artif. Intell. 74 (2018) 121–133, http://dx.doi.org/10.1016/j.engappai.2018.06.006.

[136] V. Kouliaridis, G. Kambourakis, D. Geneiatakis, N. Potha, Two anatomists are better than one—Dual-level android malware detection. https://doi.org/10.3390/sym12071128.

[137] R. Nix, J. Zhang, Classification of Android apps and malware using deep neural networks, in: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 1871–1878, http://dx.doi.org/10.1109/IJCNN.2017.7966078.

[138] H. Cai, N. Meng, B. Ryder, D. Yao, DroidCat: Effective android malware detection and categorization via app-level profiling, IEEE Trans. Inf. Forensics Secur. 14 (6) (2019) 1455–1470, http://dx.doi.org/10.1109/TIFS.2018.2879302.

[139] M.K. Alzaylaee, S.Y. Yerima, S. Sezer, Dynalog: an automated dynamic analysis framework for characterizing android applications, in: 2016 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), 2016, pp. 1–8, http://dx.doi.org/10.1109/CyberSecPODS.2016.7502337.

[140] Z. Wang, Q. Liu, Y. Chi, Review of android malware detection based on deep learning, IEEE Access 8 (2020) 181102–181126, http://dx.doi.org/10.1109/ACCESS.2020.3028370.

[141] V. Kouliaridis, K. Barmpatsalou, G. Kambourakis, G. Wang, Malwarehouse: A data collection-as-a-service of mobile malware behavioral patterns, in: 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 2018, pp. 1503–1508, http://dx.doi.org/10.1109/SmartWorld.2018.00260.

[142] A mobile malware detection method using behavior features in network traffic, J. Netw. Comput. Appl. 133 (2019) 15–25, http://dx.doi.org/10.1016/j.jnca.2018.12.014.

[143] C.-W. Yeh, W.-T. Yeh, S.-H. Hung, C.-T. Lin, Flattened data in convolutional neural networks: Using malware detection as case study, in: Proceedings of the International Conference on Research in Adaptive and Convergent Systems, 2016, pp. 130–135, http://dx.doi.org/10.1145/2987386.2987406.

[144] S. Arshad, M.A. Shah, A. Wahid, A. Mehmood, H. Song, H. Yu, SAMADroid: A novel 3-level hybrid malware detection model for android operating system, IEEE Access 6 (2018) 4321–4339, http://dx.doi.org/10.1109/ACCESS.2018.2792941.

[145] S.I. Imtiaz, S. ur Rehman, A.R. Javed, Z. Jalil, X. Liu, W.S. Alnumay, DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network, Future Gener. Comput. Syst. 115 (2021) 844–856, http://dx.doi.org/10.1016/j.future.2020.10.008.

[146] V.G.T. da Costa, S. Barbon, R.S. Miani, J.J.P.C. Rodrigues, B.B. Zarpelão, Detecting mobile botnets through machine learning and system calls analysis, in: 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1–6, http://dx.doi.org/10.1109/ICC.2017.7997390.

[147] M. Spreitzenbarth, T. Schreck, F. Echtler, D. Arp, J. Hoffmann, Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques, Int. J. Inf. Secur. 14 (2) (2015) 141–153, http://dx.doi.org/10.1007/s10207-014-0250-0.

[148] Y. Yang, Z. Wei, Y. Xu, H. He, W. Wang, Droidward: an effective dynamic analysis method for vetting android applications, Cluster Comput. 21 (1) (2018) 265–275, http://dx.doi.org/10.1007/s10586-016-0703-5.

[149] W. Yu, H. Zhang, L. Ge, R. Hardy, On behavior-based detection of malware on Android platform, in: 2013 IEEE Global Communications Conference (GLOBECOM), 2013, pp. 814–819, http://dx.doi.org/10.1109/GLOCOM.2013.6831173.

[150] S. Hou, A. Saas, L. Chen, Y. Ye, Deep4MalDroid: A deep learning framework for android malware detection based on linux kernel system call graphs, in: 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW), 2016, pp. 104–111, http://dx.doi.org/10.1109/WIW.2016.040.

[151] M.K. Alzaylaee, S.Y. Yerima, S. Sezer, DL-Droid: Deep learning based android malware detection using real devices, Comput. Secur. 89 (2020) 101663, http://dx.doi.org/10.1016/j.cose.2019.101663, URL https://www.sciencedirect.com/science/article/pii/S0167404819300161.

[152] F. Martinelli, F. Marulli, F. Mercaldo, Evaluating convolutional neural network for effective mobile malware detection, Procedia Comput. Sci. 112 (2017) 2372–2381, http://dx.doi.org/10.1016/j.procs.2017.08.216.

[153] P. Faruki, B. Buddhadev, B. Shah, A. Zemmari, V. Laxmi, M.S. Gaur, Droid-DivesDeep: Android malware classification via low level monitorable features with deep neural networks, in: International Conference on Security & Privacy, 2019, pp. 125–139, http://dx.doi.org/10.1007/978-981-13-7561-3_10.

[154] P. Feng, J. Ma, C. Sun, X. Xu, Y. Ma, A novel dynamic android malware detection system with ensemble learning, IEEE Access 6 (2018) 30996–31011, http://dx.doi.org/10.1109/ACCESS.2018.2844349.

[155] I. Burguera, U. Zurutuza, S. Nadjm-Tehrani, Crowdroid: behavior-based malware detection system for android, in: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, 2011, pp. 15–26, http://dx.doi.org/10.1145/2046614.2046619.

[156] W.-C. Wu, S.-H. Hung, DroidDolphin: a dynamic android malware detection framework using big data and machine learning, in: Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems, 2014, pp. 247–252, http://dx.doi.org/10.1145/2663761.2664223.

[157] M.F. Ab Razak, N.B. Anuar, R. Salleh, A. Firdaus, The rise of "malware": bibliometric analysis of malware study, J. Netw. Comput. Appl. 75 (2016) 58–76, http://dx.doi.org/10.1016/j.jnca.2016.08.022.

[158] McAfee, Data exfiltration study: actors, tactics, and detection, 2018, URL https://www.mcafee.com/enterprise/en-us/assets/reports/rp-data-exfiltration.pdf.

[159] H. AlKilani, M. Nasereddin, A. Hadi, S. Tedmori, Data exfiltration techniques and data loss prevention system, in: 2019 International Arab Conference on Information Technology (ACIT), 2019, pp. 124–127, http://dx.doi.org/10.1109/ACIT47987.2019.8991131.

[160] Check Point Research, Cyber security report, 2020, URL https://pages.checkpoint.com/cyber-security-report-2020.html.

[161] M. Gaudesi, A. Marcelli, E. Sanchez, G. Squillero, A. Tonda, Malware obfuscation through evolutionary packers, in: GECCO Companion '15: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, 2015, pp. 757–758, http://dx.doi.org/10.1145/2739482.2764940.

[162] A. Afianian, S. Niksefat, B. Sadeghiyan, D. Baptiste, Malware dynamic analysis evasion techniques: A survey, ACM Comput. Surv. 52 (6) (2019) 1–28, http://dx.doi.org/10.1145/3365001.

[163] M. Al-Kasassbeh, S. Mohammed, M. Alauthman, A. Almomani, Feature selection using a machine learning to classify a malware, in: Handbook of Computer Networks and Cyber Security, 2020, pp. 889–904, http://dx.doi.org/10.1007/978-3-030-22277-2_36.

[164] G. Liang, J. Pang, Z. Shan, R. Yang, Y. Chen, Automatic benchmark generation framework for malware detection, Secur. Commun. Netw. https://doi.org/10.1155/2018/4947695.

[165] D.J. Miller, Z. Xiang, G. Kesidis, Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks, Proc. IEEE 108 (3) (2020) 402–433, http://dx.doi.org/10.1109/EISIC.2017.21.

[166] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, G. Loukas, A taxonomy and survey of attacks against machine learning, Comp. Sci. Rev. 34 (2019) http://dx.doi.org/10.1016/j.cosrev.2019.100199.

[167] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, C. Nicholas, Malware detection by eating a whole exe, 2017, arXiv preprint arXiv:1710.09435.

[168] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, F. Roli, Adversarial malware binaries: Evading deep learning for malware detection in executables, in: 2018 26th European Signal Processing Conference (EUSIPCO), 2018, pp. 533–537, http://dx.doi.org/10.23919/EUSIPCO.2018.8553214.

[169] V. Tolpegin, S. Truex, M.E. Gursoy, L. Liu, Data poisoning attacks against federated learning systems, in: European Symposium on Research in Computer Security, 2020, pp. 480–501, http://dx.doi.org/10.1007/978-3-030-58951-6_24.

[170] N.M. Müller, D. Kowatsch, K. Böttinger, Data poisoning attacks on regression learning and corresponding defenses, 2020, arXiv preprint arXiv:2009.07008.

[171] R. Taheri, R. Javidan, M. Shojafar, Z. Pooranian, A. Miri, M. Conti, On defending against label flipping attacks on malware detection systems, Neural Comput. Appl. https://doi.org/10.1007/s00521-020-04831-9.

[172] F. Tramèr, F. Zhang, A. Juels, M.K. Reiter, T. Ristenpart, Stealing machine learning models via prediction apis, in: SEC'16: Proceedings of the 25th USENIX Conference on Security Symposium), 2016, pp. 601–618.

[173] X. Wang, Y. Xiang, J. Gao, J. Ding, Information laundering for model privacy, 2020, arXiv preprint arXiv:2009.06112.

[174] R. Moraffah, M. Karami, R. Guo, A. Raglin, H. Liu, Causal interpretability for machine learning-problems, methods and evaluation, ACM SIGKDD Explor. Newsl. 22 (1) (2020) 18–33, http://dx.doi.org/10.1145/3400051.3400058.

[175] L.H. Gilpin, D. Bau, B.Z. Yuan, A. Bajwa, M. Specter, L. Kagal, Explaining explanations: An overview of interpretability of machine learning, in: 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), 2018, pp. 80–89, http://dx.doi.org/10.1109/DSAA.2018.00018.

[176] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A survey on bias and fairness in machine learning, 2019, arXiv preprint arXiv:1908.09635.

[177] M.Q. Li, B. Fung, P. Charland, S.H. Ding, I-MAD: A novel interpretable malware detector using hierarchical transformer, 2019, arXiv preprint arXiv:1909.06865.

[178] A. Mills, T. Spyridopoulos, P. Legg, Efficient and interpretable real-time malware detection using random-forest, in: 2019 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (Cyber SA), 2019, pp. 1–8, http://dx.doi.org/10.1109/CyberSA.2019.8899533.

Pascal Maniriho received his B.Tech with Honors in Information and Communication Technology from Umutara Polytechnic, Rwanda and Master's degree in Computer Science from Institut Teknologi Sepuluh Nopember (ITS), Indonesia, in 2013 and 2018, respectively. He has been working in academia in Information Technology since 2019. He is currently pursuing his Ph.D. degree in cybersecurity at La Trobe University, Australia. His research interests include malware detection, data theft prevention, information security, machine learning and deep learning.

Abdun Naser Mahmood received the B.Sc. degree in applied physics and electronics, and the M.Sc. (research) degree in computer science from the University of Dhaka, Bangladesh, in 1997 and 1999, respectively, and the Ph.D. degree from the University of Melbourne, Australia, in 2008. He is currently an Associate Professor with the Department of Computer Science, School of Engineering and Mathematical Sciences, La Trobe University. His research interests include data mining techniques for scalable network traffic analysis, anomaly detection, and industrial SCADA security. He is a senior member of the IEEE.

Dr. Mohammad Jabed Morshed Chowdhury is currently working as Associate Lecturer at La Trobe University, Melbourne, Australia. He has earned his Ph.D. Candidate at Swinburne University of Technology, Melbourne, Australia. He has earned his double Masters in Information Security and Mobile Computing from Norwegian University of Science and Technology, Norway and University of Tartu, Estonia under the European Union's Erasmus Mundus Scholarship Program. He has published his research in top venues including TrustComm, HICSS, and REFSQ. He is currently working with Security, Privacy and Trust. He has published research work related to blockchain and cyber security in different top venues.