The Journal of Systems and Software 209 (2024) 111921

Contents lists available at ScienceDirect

The Journal of Systems & Software

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

A systematic literature review on Windows malware detection: Techniques, research issues, and future directions☆

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

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

ARTICLE INFO

MSC:
00-01
99-00

Keywords:
Malware analysis
Malware detection
Malware dataset
Windows malware
Machine learning
Deep learning

ABSTRACT

The aim of this systematic literature review (SLR) is to provide a comprehensive overview of the current state of Windows malware detection techniques, research issues, and future directions. The SLR was conducted by analyzing scientific literature on Windows malware detection based on executable files (.EXE file format) published between 2009 and 2022. The study presents new insights into the categorization of malware detection techniques based on datasets, features, machine learning and deep learning algorithms. It identifies ten experimental biases that could impact the performance of malware detection techniques. We provide insights on performance evaluation metrics and discuss several research issues that impede the effectiveness of existing techniques. The study also provides recommendations for future research directions and is a valuable resource for researchers and practitioners working in the field of Windows malware detection.

1. Introduction

In the cybersecurity domain, malware refers to any form of malicious program that cyber attackers (hackers) employ to infiltrate computing devices such as computers and smartphones to damage or gain access to sensitive information without the users' knowledge (Chen et al., 2012). Malware can perform a set of malicious activities after compromising the system. For instance, some malware programs such as Trojans are often employed by hackers to steal users' confidential data, modify data, control/manage compromised devices remotely, slow the performance of network and connected devices, delete users' data, or perform other types of malicious actions (Anon, 2021f). Generally, malware covers all malicious programs/harmful code such as viruses, spyware, ransomware, backdoors, and many more (Ye et al., 2017a). Malware programs (files) are designed to target various platforms such as desktop and mobile operating systems (Huang et al., 2021; Qamar et al., 2019) and Internet of Things (IoT) devices (Jeon et al., 2020). Recently, malware programs have evolved into various complex and advanced variants such as ransomware that can encrypt, hijack, and exfiltrate sensitive information (Oz et al., 2021).

Malware has become the most potent automated tool for hackers to conduct/launch different cyber attacks targeting individuals, as well as small, medium, and large businesses around the world. Accordingly, there has been a significant increase in malware attacks over the last decade (CyberEdge Group, 2021). As reported in SOPHOS (2021a), 37% of organizations around the world were affected by ransomware attacks with retail and education being the highly affected sectors, an example of how ransomware attacks greatly hit computers and other network devices. The report in SOPHOS (2021a) has also indicated that the ransom demand has risen rapidly, increasing the average ransom payout by 64% (from $84,116.00 in Q4 2019 to $233,817.30 million in Q3 2020). The overall global average cost to recover from a ransomware attack has more than doubled from $0.761 million in 2020 to $1.85 million in 2021, reflecting a very high financial deficit caused by ransomware cyber-attacks around the globe (SOPHOS, 2021b).

Ransomware has moved beyond straightforward cyber attacks into a realm of sophisticated multi-purpose/stage malware attacks (attack mechanisms that encrypt, exfiltrate, and hold the victim's confidential data until a ransom is paid) (Oz et al., 2021; SOPHOS, 2021a). Cyber attackers have greatly improved ransomware attacks using sophisticated encryption and anonymous payment techniques. Additionally, ransomware programs are now being distributed in the form of ransomware as a service (RaaS) (Oz et al., 2021). For instance, according to the threat report released by Esentire (2021), Colonial Pipeline, the biggest and main oil pipeline that carries gasoline and jet fuel

☆ Editor: Martin Pinzger.

* Correspondence to: Department of Computer Science and Information Technology, School of Engineering and Mathematical Sciences, La Trobe University, Bundoora, Victoria 3086, Australia.
E-mail addresses: P.Maniriho@latrobe.edu.au (P. Maniriho), A.Mahmood@latrobe.edu.au (A.N. Mahmood), m.chowdhury@latrobe.edu.au (M.J.M. Chowdhury).

https://doi.org/10.1016/j.jss.2023.111921
Received 2 February 2023; Received in revised form 22 November 2023; Accepted 5 December 2023
Available online 16 December 2023

to the South-eastern USA was hit by a ransomware attack which was attributed to DarkSide, a RaaS malware variant from a group of cybercriminals based in Eastern Europe. This attack disrupted all pipeline operations, and some of the IT systems were affected, causing chaos and panic. Cisco-Secure (2021) also reported cryptominers and Trojans malware as severe threats, among others.

The risks and damages caused by malware attacks are often different, considering whether the malware has compromised a corporate network, home network, or individual device. As stated in CyberEdge Group (2021), among the organizations that were surveyed, 86.2% were compromised by at least one successful cyber-attack. Sophos (SOPHOS, 2021a) revealed that financial institutions, the healthcare, retail, and education sectors will continue to be the most highly targeted sectors by malware attacks. On the other hand, Dailyhostnews (John, 2022) has recently revealed that most of the new malware attacks (more than 95%) discovered in 2022 target Windows desktop devices, making them the top most devices infected by malware attacks.

Different solutions based on machine learning (ML) and deep learning (DL) algorithms were proposed to address this problem. Ramteke et al. (2021) proposed a Light Gradient Boosting-based model for detecting malware attacks in banking and financial institutions. Li et al. (2021) proposed a CNN-based approach that detects malware attacks by converting executable binary files of benign and malware into decimal arrays. Their experiment was conducted on a dataset of 9458 files belonging to 25 malware families. The performance shows that their detection model's accuracy has reached 97%. Priyadarshan et al. (2021) used bytes features extracted from benign and malware files to implement ML-based classifiers which achieved good detection accuracy while detecting unseen malware attacks. An LSTM-based method for malware classification was developed using system calls extracted from malware while running in the isolated dynamic analysis environment (Or-Meir et al., 2021). With this method, long sequences of system calls invoked by malware can be processed in a short time, and the detection results show an improvement of 6% over other malware detection methods. Raff et al. (2017) also presented a raw byte-based method that detects malware using CNN and recurrent neural network algorithms.

Consequently, there has been a significant rise in the number of work that focuses on malware detection in Windows. Therefore, the aim of this work is to present a comprehensive study on the detection of malware attacks in Windows. We systematically identify and synthesize relevant information presented in the existing works to provide state-of-the-art techniques in malware detection. This work also identifies and discusses the limitations of existing works from various perspectives and provides our recommendations to address them, opening new avenues of research in this area. More specifically, we followed the research protocols for performing SLR presented by Kitchenham et al. (2009), which allowed us to select relevant studies and respond to various research questions defined in Section 4 of this work. Furthermore, each selected paper's quality was assessed based on the assessment criteria defined in Section 4. The following are the main contributions of this work.

- Extracting and synthesizing relevant information from selected high-quality research papers that were focused on malware detection in Windows desktop devices.
- Identifying malware detection techniques and discussing their deployment methods.
- Compiling and reporting public benchmark datasets including the widely used performance evaluation metrics, ML, and DL algorithms.
- Presenting critical experimental biases (such as temporal bias, spatial bias, sample size, etc.) which can lead to inflated experimental results when not taken into consideration while building malware detection techniques.
- Presenting research issues that impede the development of reliable and advanced techniques for detecting malware attacks and providing future research directions.

Organization: The rest of this systematic study is structured as follows. Section 2 provides the background and basic malware analysis and detection concepts. Section 3 discusses related works on malware detection. Section 4 presents the methodology for performing the review. Sections 5, 6, 7, 8, 9, 10, and 11, discuss the research findings and present answers for the research questions, respectively. Section 12 presents the conclusion of this work.

2. Background and basic concepts

This section presents a detailed background on Windows executable files and discusses the prevalence of malicious executable files. It also discusses types of malware analysis, different forms of malware, and steps for building malware detection using DL and ML algorithms.

2.1. Windows portable executable file structure

The portable executable (or PE for short) is the standard file format for executable files/programs used in Microsoft 32-bit and 64-bit Windows OS (Sikorski and Honig, 2012). PE files have different formats such as .exe files, dynamic link libraries (.dlls), BAT/Batch files (.bat), control panel applications (.cpl), kernel modules (.srv), device drivers (.sys), and many more,[1] with the .exe files being the most used of all file formats. The data structure of a PE file has essential information required for the OS Loader to load and execute the executable program into memory. The structure of a PE file begins with a DOS Header at the top followed by a DOS Stub, NT Headers, Section Headers, and finally the Overlay. Located at the top of .exe files, the **DOS Header** is a long structure that occupies the first 64 bytes of the PE file. It makes the file an MS-DOS executable (it identifies the type of the file compatible with MS-DOS). The **DOS Stub** holds a piece of code that is executed by default during program execution. The stub displays a string error message warning the user if the program cannot be executed in MS-DOS. on the other hand, the **NT Headers** (IMAGE_NT_HEADERS) hold the PE signature, file header, and optional header components which are necessary for PE file execution.

- PE signature: This is a 4-byte signature used to identify the file as a portable executable file.
- PE File header: A Common Object File Format (COFF) header comes directly after the PE signature. It contains field information such as machine type (which holds values to specify the type of CPU), the number of sections, the number of symbols, and the size of the operational header, to name a few.
- PE Optional header: This header holds useful information that is necessary to load and run an executable file. While the optional header is important for image files, it is not needed for some types of files such as object files, the reason why it is optional.

The **Section Header** (Section Table) represents an array of Image Section Headers with each header containing information for the section it refers to. A PE file has different section headers, however, it is important to note that the number of sections differs based on the function of the file. The following are examples of section headers observed in .exe files.

- The .text section: This section holds the program's executable code/instructions to be executed by the CPU.
- The .data section: This section holds initialized data defined in the code.

[1] https://docs.fileformat.com/executable/exe/

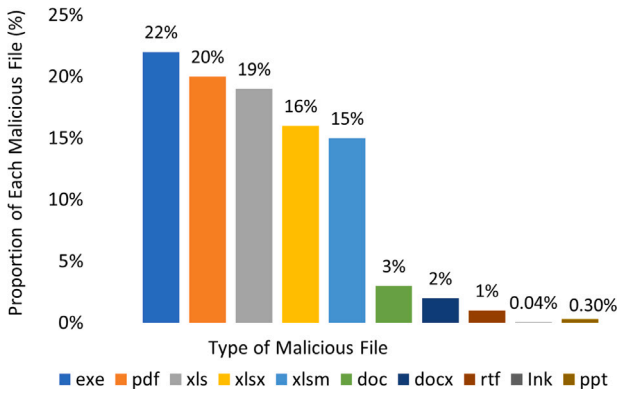

Fig. 1. Top malicious file types encountered over the web (Checkpoint, 2023).

- The .rdata section: It holds information such as constants, string literals, and debug directory information (read-only data).
- The .bss section: All uninitialized data for the application (e.g. static variables) are found in this section. It is referred to as Better Save Space (bss).
- The .idata section: This section contains the import table such as DLLs and Application Programming Interface (API) functions.
- The .edata: This section is used to hold the export table.
- The .reloc: It holds information for image relocation.
- The .rsrc section: Resources such as images, embedded binaries, menus, and icons used by the application are found in this section.
- The .tls: The thread local storage (tls) manages/provides storage for each program's executing thread.

Finally, the **Overlay** contains additional information which is appended at the end of the file. Further details showing the description of each PE file's component can be found in Microsoft documentation.[2]

2.2. Malware executable file against windows

Developed by Microsoft, Windows OS is the most widely used and distributed desktop OS with the highest global market share (Anon, 2023). Consequently, the popularity of Windows OS and its global usage make it the primary target of malware attacks worldwide. As recently reported by Checkpoint,[3] phishing emails and online websites are the main attack vectors used by hackers to spread malicious executable files over the Internet. In addition, statistics also show that malicious executable files (.exe files) are more highly produced by hackers than other file formats (Checkpoint, 2023), making them more challenging to detect. Accordingly, Fig. 1 depicts the proportion of different malicious file formats on the web and as it can be seen, the .exe file formats are ranked on top of other malicious files. The next Section 2.3 discusses how malicious files (malware programs) violate organizational security principles to compromise the target. It also discusses different types of emerging malware.

2.3. Security violation performed by malware

All malevolent activities intended by hackers are performed by malware after gaining unauthorized entry into the compromised system or host. A wide range of malicious activities can be carried out, with the attacks violating or compromising existing organizational security principles such as data confidentiality, integrity, and availability. Specifically, malware attacks violate these security principles as follows.

Confidentiality–Malware can steal organizational or individual confidential data such as users' login credentials/authentication details (username, password, and personal identification number), financial data (bank account number, credit card number, and security code), social security number, health data (patients' records) and other categories of confidential data. Modern malware such as keyloggers steals confidential information by capturing users' keystrokes and storing them in a file that is remotely accessed by cyber attackers. The stolen data can also be sent automatically to cyber attackers through email. Such keystrokes can contain sensitive information like PINs, passwords, usernames, or other forms of confidential messages exchanged between two or many entities. Moreover, some sophisticated keyloggers known as "screen recorders", are also able to capture users' screen activities at random intervals (Tuscano and Koshy, 2021; Solairaj et al., 2016).

Integrity–Malware can modify existing sensitive information through unauthorized access. For example, malware can capture network traffic and modify the packet information such as the source IP address and the information being transmitted. Such an attack is called a man-in-the-attack (MITM) which compromises the integrity of data and it is often hard to detect it (Anon, 2021a).

Availability–A malicious program can launch denial-of-service attacks (DoS) or distributed DoS attacks which initiate a series of unusual requests (generate massive network traffic) to flood the target host or system, preventing it from being accessed by the authorized users (Weisman, 2021). DoS attacks can disrupt the service in the victim's system or crash the entire system. The malicious program can consume computer resources such as the amount of available disk space and random-access memory (RAM) while trying to mine cryptocurrencies. Additionally, malware (e.g., ransomware) can encrypt and lock all sensitive data and demand for a ransom to be paid before the data can be decrypted and accessed again (Reshmi, 2021).

In particular, there exist some advanced malware programs that can carry out more than one malicious activity, i.e., they can perform a combination of different cyber-attacks that can at the same time compromise confidentiality, integrity, and availability of sensitive data (Sikorski and Honig, 2012). For instance, modern ransomware can encrypt and exfiltration sensitive data which could result in availability and confidentiality being compromised (SOPHOS, 2021c). Another example is a keylogger that records keyboard entries such as passwords while updating itself and downloading remote Trojan malware that can launch a DoS attack against the target.

According to Martin (2021) malware cyber-attacks can be mainly accomplished through 7 steps, which are presented under what is called the "Cyber Kill Chain framework (or model)". This model has identified the steps or phases through which hackers can accomplish or execute malicious activities to achieve their objectives. More importantly, the Cyber Kill Chain framework also allows cybersecurity analysts and researchers to gain an understanding of attack mechanisms and strategies, thereby enhancing attack mitigation procedures and increasing the visibility of discovering the attack. These steps are summarized in Fig. 2. Furthermore, Table 1 presents different types of well-known malware.

2.4. Malware analysis techniques

Malware analysis plays a significant role in discovering patterns that can be used to detect and prevent future threats. Important information about the program or file is extracted during the program analysis, and relevant patterns are later selected from the extracted information to represent the suspicious program under analysis. That is, malware analysis is one of the best ways to discover malicious characteristics of malware programs. The main objective of carrying out malware analysis is to extract relevant patterns from any suspicious file that

[2] https://learn.microsoft.com/en-us/windows/win32/debug/pe-format

[3] https://go.checkpoint.com/2022-mid-year-trends/page-global-malware-statistics.php

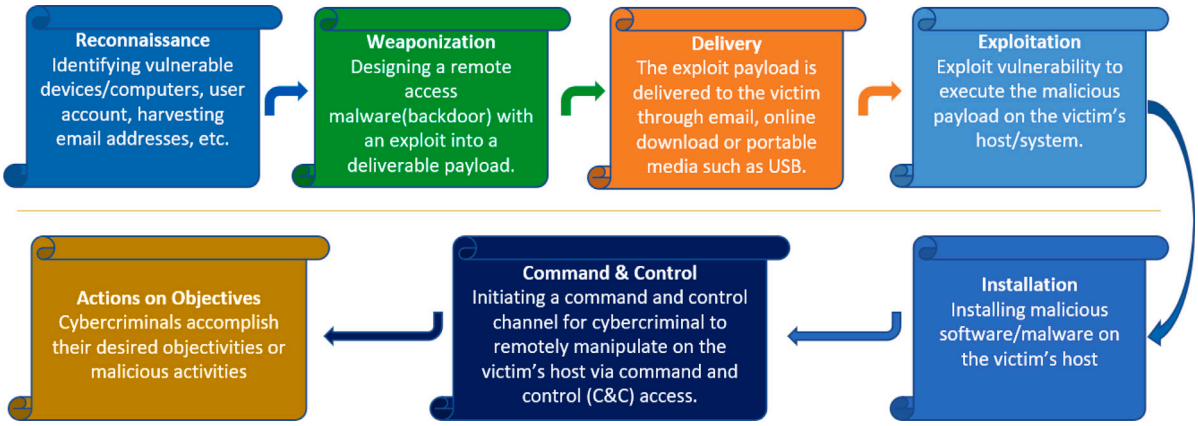

Fig. 2. The cyber kill chain model showing the phases of a cyber attack (Martin, 2021).

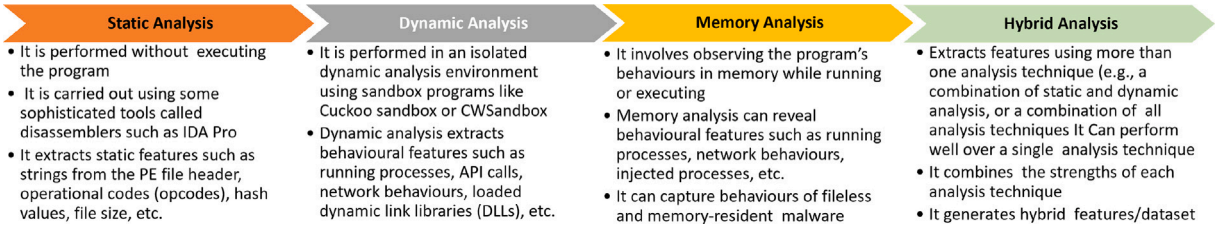

Fig. 3. Different types of malware analysis techniques in Windows malware detection.

Table 1
Types of existing malware and their malicious characteristics.

Malware type	Malicious characteristics	Example of malware
Spyware	Monitors and gathers user's activities without noticing it	Transponder (vx2)
Worms	Perform self-replication over the network and spread to infect other systems	Slammer
Backdoor	Once implanted on the victim's host, it allows hackers to gain full control over the compromised host	ShadowPad
Adware	Potentially unwanted programs (PUP) that can display undesirable adverts and can add spyware on the targeted device, alter the browser's homepage, and bombard the device with massive adverts	Fireball
Virus	Can replicate itself offline and once installed on the victim's host, it modifies and infects other computer programs by inserting its malicious code	Shlayer
Keyloggers	Can monitor users' keystrokes and capture live screenshots without their consent.	PunkeyPOS
Trojan Horses	Pretend to be legitimate software by masquerading themselves. They can collect and send secret data to the cyber attacker	Rakhni Trojan
Rootkits	Infect computers and disguise their presence while still actively performing malicious activities	ZeroAccess
Cryptojacking	Once installed, this malware can mine and steal cryptocurrency wallets from other victims. It also consumes the victim's system resources such as memory or CPU	Coinhive
Ransomware	Encrypts and holds the victim's confidential data until a ransom is paid	RobbinHood
Botnet	Can launch DDoS cyber-attacks, distributed spam and phishing emails	Conficker
Fileless Malware	It is a memory-based malware and utilizes other legitimate programs such as Windows PowerShell and Windows Management Instrumentation (WMI) to perform its malicious activities	Astaroth
Obfuscated malware	It compresses and hides malicious code/component which is unpacked during program execution. Obfuscation allows evading the analysis tools and detection systems	FlashUtil.exe

can be utilized in responding to malware cyber threats. After the analysis task is accomplished, the next step is to build malware detection techniques using the selected patterns or features representing malware and benign programs. The final output is a malware detection technique with the ability to determine whether a particular program or file intends to perform malicious activities or not. Static, dynamic, memory, and hybrid analysis are different types of malware analysis techniques (Singh and Singh, 2021; Maniriho et al., 2021) and details on each technique are summarized in Fig. 3.

2.5. ML and DL algorithms for malware detection

In the early days of cybersecurity, the number of malware attacks was extremely low, where simple detection mechanisms like manually defined pre-execution filtering rules were sufficient to detect most types of malware. Nevertheless, the rapid development of Internet applications led to an explosion of a high number of malware attacks, making it impossible/impractical to manually create detection rules for each malware. This created the need for new advanced security

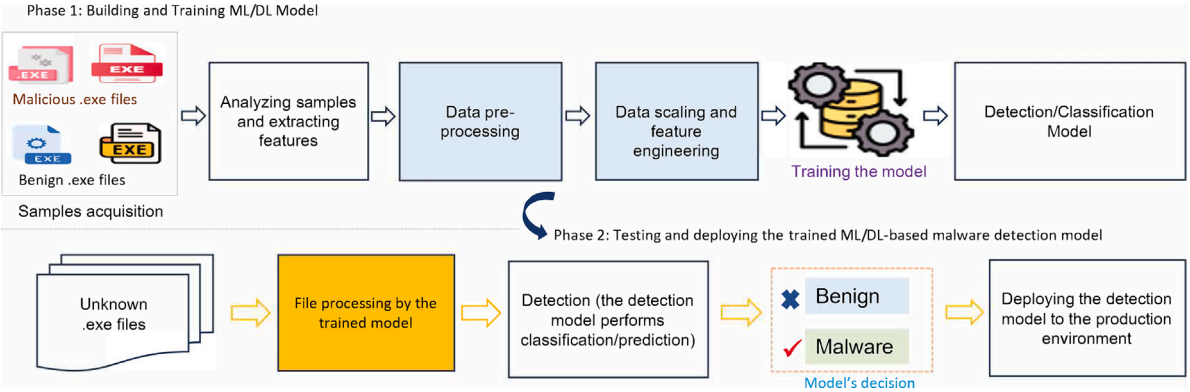

Fig. 4. Machine learning and deep learning-based malware detection life cycle.

technologies. Consequently, global anti-malware security companies shifted to machine learning and deep learning technologies, to enhance the capability of their anti-malware classification products. ML and DL algorithms have become the key components in today's security products such as antivirus (AVs) and other threat-hunting tools (Coombs, 2018). These algorithms decide if a file/program is malicious or normal (benign) based on the data they have been trained and tested on. In Windows OS, these data may include strings, sequences of API calls, file hash signatures, and processes that can be statically or dynamically extracted from benign and malicious executable files.

ML and DL algorithms can solve supervised learning (learning from labeled data) (Karbab and Debbabi, 2019; Maniriho et al., 2023), semi-supervised (learns from a small number of labeled data and a large amount of unlabeled data) (Gao et al., 2020), and unsupervised (learns from unlabeled data) (Mohaisen et al., 2015) malware detection tasks, making them crucial in protecting the current cyber infrastructures. Reinforcement learning is another category of ML algorithm that focuses on the application/use of software agents to make a decision in an environment and has been used for malware detection (Bidoki et al., 2017). In Section 2.6, we discuss steps for building malware detection techniques while current DL and ML-based techniques for malware detection are presented in Sections 7 and 8.

2.6. Steps for building ML and DL-based malware detection techniques

Building techniques for detecting malware attacks involves several steps to be followed. Malware detection techniques are systems designed to determine if a file, program code, or suspected URL has malicious intentions or not. ML and DL algorithms are mostly used to build these systems and constitute an integral part of modern malware detection systems. Therefore, the following are the main steps for building malware detection techniques. These steps show the development life cycle of ML and DL-based malware detection techniques and they are summarized in Fig. 4.

Samples acquisition—This step involves collecting malware and benign samples to be used in the experiment. Benign samples can be collected from online repositories such as CNET site (Anon, 2021b) while malware samples can be collected from well-known public repositories such as VirusTotal.[4]

Analyzing samples and extracting features—Setting up the analysis environment, analyzing collected samples/programs, and extracting raw features to represent benign and malware are performed at this stage. Details of malware analysis approaches are presented in Section 2.4 while different features that can be extracted throughout the analysis are provided in Tables 4, 5, and 6.

Data pre-processing—This step involves cleaning the raw dataset generated in step 2 to remove any inconsistencies such as missing values, duplicate values, and other unwanted data. Numerical data representation which is supported by machine learning or deep learning algorithms is generated at this stage. This gives a good feature representation, which is passed to the next step to be scaled and select the best features for building ML and DL-based malware detection techniques.

Data scaling and feature engineering—At this stage, the feature set generated in the third step is fed to the data scaling module which transforms/scales the data to bring values in the same intervals wherever necessary. Thereafter, the scaled feature set is passed to the feature engineering module which performs various feature engineering operations to select relevant features. One portion of the data is used to train the detection technique and the remaining portion is used for testing. Feature selection (also referred to as features' dimensionality reduction) can be accomplished using techniques such as truncated singular value decomposition (truncated SVD) (Singh and Singh, 2020). Further details on different feature selection techniques can be found in these studies (Maniriho et al., 2021; Ye et al., 2017b). Nevertheless, it is important to mention that different from traditional ML, feature selection is automatically performed when using DL algorithms, i.e., there is no need to specify a feature selection technique.

Training the model—At this stage, an ML or DL-based model is trained using the training set. All selected features representing malware and benign are used to train the malware detection model. It is worth noting that training also involves hyperparameter tuning to improve the model performance.

Testing the model—This stage involves testing the detection model using unseen samples from the test set. It reveals the efficiency of the trained model. If the model's performance is satisfactory, it is validated and then deployed to the production environment. The performance can be evaluated by measuring several metrics that are crucial for validating the detection technique and details on these metrics are presented in Section 9.

Deploying and monitoring the model—Once the detection model has been tested on unseen data, it is then deployed into the production environment to protect end users' devices from malware attacks. Deployed models will also be monitored to observe any signs of errors or performance degradation over time. Monitoring helps to ensure that newly deployed malware detection models perform as expected in production.

3. Previous surveys on windows malware detection

Due to significant advances in Windows malware detection, there are several ML and DL-based malware detection techniques presented in the literature. Many of these techniques were previously reviewed to understand the research findings and advance the knowledge in the area of malware detection. The examples of such review papers

[4] https://www.virustotal.com/

were presented in Ye et al. (2017b), Shah and Singh (2015), Ucci et al. (2019), Or-Meir et al. (2019), Moussaileb et al. (2021), Gorment et al. (2021), Pachhala et al. (2021), Singh and Singh (2021), Verma and Sharma (2021), James and Sabitha (2021), Yadav et al. (2022). However, new malware detection techniques continually appear in the literature, creating the need for updated surveys. This section discusses these related works (surveys) and highlights their limitations, which motivated us to conduct this SLR study.

The review in Ye et al. (2017b) has presented an overview of malware taxonomy and the need for malware detection in the industrial sectors. The authors have discussed feature extraction approaches (static, dynamic, and hybrid analysis) and other approaches such as the one that represents malware samples with their semantics contents. Different feature selection techniques and various data mining methods (clustering and classification) for malware detection were also presented. Nevertheless, this work did not explore important topics such as malware deployment methods, public benchmark datasets, various experimental biases, and some valuable research issues such as adversarial learning attacks. Having been published in 2017, the survey study in Ye et al. (2017b) lacks valuable knowledge, making it outdated. In addition, the authors of this work did not follow a formal approach such as the SLR approach while conducting their survey.

Some of the existing surveys are misleading and incomplete. For instance, the work in Mane et al. (2022) claims to present a review of malware detection methods based on deep learning, but their work only cited four works that used deep learning. The authors also mistakenly categorize traditional ML algorithms as DL algorithms. Another example is the survey carried out by Yadav et al. (2022) which covered neither ML-based nor DL-based techniques for malware detection. Their work only covered various categories of emerging malware and how they attack network devices.

The work presented by Ucci et al. (2019) elaborated on feature extraction techniques and discussed different features extracted in executable files. This work has also reviewed several ML-based techniques for malware detection including some of their limitations. Despite a large compilation of ML-based related works, their study did not survey any work that used DL algorithms to implement malware detection techniques. Additionally, with their survey having only research articles published until 2018, this survey is also outdated, given new malware detection techniques presented in the literature since 2018.

Or-Meir et al. (2019) conducted a comprehensive survey that only focused on advances in malware detection techniques based on dynamic analysis. Consequently, research studies related to static analysis and hybrid analysis were not reviewed. In addition, some of the topics covered in this work are also incomplete. For example, discussions on topics such as emerging DL algorithms, performance metrics, and experimental bias are missing in their survey. Similar limitations observed in Or-Meir et al. (2019) are also found in the work presented in Singh and Singh (2021). Although the work in Singh and Singh (2021) discusses the use of some performance metrics (five metrics: true positive rate, false positive rate, precision, accuracy, and AUROC), the list is incomplete, with several metrics for evaluating malware detection missing in their survey.

A survey on detecting malware attacks in Windows was presented by Moussaileb et al. (2021). Their systematic review investigated various attack vectors and attack stages employed by ransomware to compromise the victim's system. Different countermeasures were also discussed including research gaps. However, Moussaileb et al.'s work (Moussaileb et al., 2021) did not explore techniques for detecting other categories of malware, limiting their work by only reviewing ransomware detection-related studies.

The work in Maniriho et al. (2021) presented malware analysis testbeds, malware analysis techniques, and different categories of ML and DL algorithms for detecting malware attacks in Windows and Android devices. The authors also presented dynamic malware analysis tools supported by Windows, Android, and Linux operating systems. They have also presented existing repositories for benign and malware samples including features extracted through dynamic analysis in both Windows executable files and Android APK files. While this work significantly explored malware detection, it was only limited to malware detection based on malware dynamic analysis, ignoring relevant works proposed on static and hybrid-based malware detection. Shah and Singh (2015) also limited their work by only covering malware analysis based on the dynamic analysis approach and presenting some dynamic features used for malware detection.

Pachhala et al. (2021) conducted a study that reviewed malware detection techniques developed using machine learning algorithms. Nonetheless, their review did not deeply cover the use of existing DL-based techniques. In addition, hybrid-based techniques were also not reviewed. The works in Mira (2019), Chang et al. (2022) reviewed malware detection in Windows. Nevertheless, their scope was only limited to API call-based techniques for malware detection which is a major limitation, considering many features used to detect malware attacks in Windows. Inayat et al. (2021) presented a survey that covered the types of malware detection techniques. Their survey reviewed ML-based techniques for malware detection. However, this work did not review any DL-based techniques for malware detection. The review of ML-based studies is also incomplete. While the survey presented in James and Sabitha (2021) tried to cover different mitigation strategies (such as categories of malware analysis and some features extracted from executable files) to combat malware attacks, their work is also incomplete as it did not fully cover ML and DL-based techniques for malware detection.

Our Survey: We use the systematic literature review approach to extend the previous surveys and address their limitations. The SLR approach allows us to identify and select high-quality research papers that are thoroughly reviewed to extract and synthesize relevant information on malware detection in Windows desktop devices. Accordingly, 219 research papers published between 2009 and 2022 were selected, and valuable information extracted from these papers is synthesized in this SLR study. First, we presented a detailed background, enabling readers to easily grasp various topics covered in this SLR study. Such topics include malware detection techniques and their deployment methods. Existing public benchmark datasets have been compiled and grouped based on extracted features (features that are used to represent malware and benign files). Moreover, this work presents the most prevalent ML and DL algorithms that are employed for developing malware detection techniques including eighteen (18) performance metrics. We also identified ten (10) critical experimental biases to be avoided when building malware detection techniques. Finally, this work highlights limitations encountered in the existing benchmark datasets and malware detection techniques, and thereafter, it provides recommendations and future directions. It is crucial to mention that other research issues that were not fully explored in the previous surveys were also discussed. Accordingly, Table 2 presents the number of papers reviewed in the previous surveys and this survey. To the best of our knowledge, there is no other SLR in the literature that adequately analyzed and systematized extracted information from the existing works in Windows malware detection as in this SLR.

4. Methodology for performing SLR in this work

In this work, we have followed the methodology presented in Brereton et al. (2007) to conduct our systematic literature review (SLR) on malware detection in Windows. A complete workflow/methodology for performing our SLR is depicted in Fig. 5 and has three main stages that are described below.

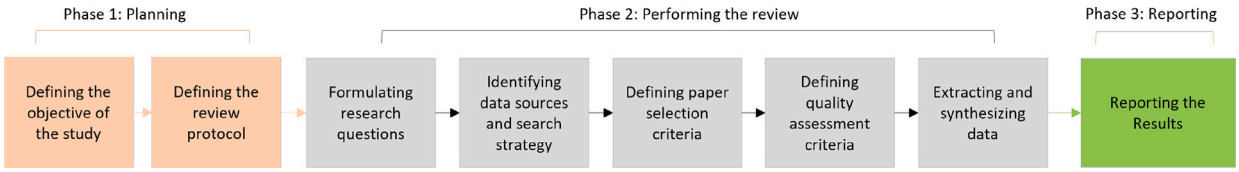

Fig. 5. Steps for performing a systematic literature review on Windows malware detection presented in this work.

Table 2
Number of papers reviewed in the previous surveys and this work.

Survey study presented in	Number of reviewed papers
Ye et al. (2017b)	184
Mane et al. (2022)	13
Yadav et al. (2022)	10
Ucci et al. (2019)	105
Or-Meir et al. (2019)	104
Singh and Singh (2021)	134
Moussaileb et al. (2021)	115
Maniriho et al. (2021)	178
Shah and Singh (2015)	12
Pachhala et al. (2021)	30
Mira (2019)	28
Chang et al. (2022)	47
Inayat et al. (2021)	44
James and Sabitha (2021)	34
This work	219

4.1. Phase 1: Planning

The planning phase involves identifying the study's objective (or goal) and defining appropriate protocols to be followed while performing this review. Thus, we have defined the objective of this study including the review process.

Objective of this study—As previously mentioned, the main objective of this study is to review the state-of-the-art studies on Windows malware detection. This work synthesizes the knowledge from these studies to better understand various malware detection topics and identify new research issues for possible future research directions. More specifically, this SLR study aims to address the research questions defined in Section 4.2.1.

Review protocol—We employed the systematic literature review approach to conduct our review on malware detection. Thus, the SLR presented in this work is an in-depth/detailed structured literature review that answers specific questions relevant to malware detection in Windows. Each reviewed paper is systematically selected and assessed based on a set of evaluation criteria. Specifically, we defined paper search guidelines and criteria for paper inclusion and exclusion and quality assessment, allowing us to identify the best papers and answer specific questions designed in this work. These guidelines and criteria are elucidated below in the second phase of the methodology.

4.2. Phase 2: Performing the review

The process for gathering the main papers from which relevant information is extracted is determined during this phase. Following the guidelines in Brereton et al. (2007), this phase is mainly accomplished through the following five (5) sub-phases.

4.2.1. Formulating research questions

The SLR presented in this work is inspired by seven research questions (RQs). All RQs are related to malware detection in Windows. They help to identify high-quality research papers and identify new future research directions in this area. Consequently, we operationalize the goal of this work by answering all defined RQs. These RQs are presented below, and we also highlight why each research question is relevant to detecting malware attacks.

RQ1: What are the types of malware detection techniques and their deployment methods?

Investigating RQ1 is motivated by a wide range of works presented in the literature. Thus, in order to understand and gain valuable insights into detecting malware attacks in Windows, it is crucial that we first identify categories of malware detection techniques (anti-malware programs) including their strengths and weaknesses. It is also ideal to discover how anti-malware programs are deployed into the end-user devices to combat emerging malware attacks. We explored the correlation observed between the development of malware detection techniques and their deployments. Therefore, establishing such theoretical aspects creates a solid foundation that leads to reaching the goal of this work.

RQ2: Which public benchmark datasets and features are used for malware detection in windows desktop devices?

By exploring RQ2, we are particularly intrigued to explore various public benchmark datasets that are used to train and test malware detection techniques. We limit our scope to identifying datasets with features extracted from Windows .exe files (malware and benign). Additionally, we delve into the limitations of these datasets and provide our insights on how to address them in future works. It is also important to highlight that without benchmark datasets, building malware detection techniques becomes impossible.

RQ3: Which ML algorithms are mostly used for detecting malware attacks?

Having investigated RQ2, it becomes essential to identify different ML algorithms that are used to build malware detection techniques. Thus, throughout RQ3, we thoroughly explore the existing literature to identify emerging ML algorithms for malware detection in Windows. Based on our observations from the literature, we also present the research trends on the application of ML algorithms in this area of malware detection.

RQ4: What current DL algorithms are employed to implement malware detection techniques?

Given the potential of deep learning algorithms in recent years, in RQ4 we are specifically interested in identifying the most prevalent DL algorithms for building malware detection techniques in Windows. Therefore, existing DL-based works are analyzed to determine current trends in the use of these algorithms and derive our recommendations.

RQ5: What are the existing evaluation metrics for assessing the performance of malware detection techniques?

Motivated by the role of performance metrics in examining and validating the performance of ML and DL-based malware detection techniques, RQ5 has been defined to explore different performance metrics presented in the literature. More importantly, we carefully identify these metrics from the literature and then discuss how they are used in various perspectives. We also mention their shortcomings wherever applicable.

RQ6: What are the critical experimental factors/biases in ML and DL-based techniques for detecting malware?

The experimental results from the existing literature show significant performance achieved by ML and DL-based malware detection techniques. As these techniques continue to emerge in malware detection, RQ6 allows us to examine the existing works to discover if they suffer from any experimental bias that could lead to inflated results and affect the sustainability of the associated malware detection technique. Therefore, we identify such biases and provide valuable recommendations to avoid them in future experiments.

RQ7: What are the research challenges in windows malware analysis and detection?

With this RQ, our goal is to rely on the research findings of this SLR to discuss and highlight a set of research issues and possible future directions for researchers in Windows malware detection. These issues serve as new research avenues that could be explored in future works to advance knowledge in this area.

4.2.2. Defining data sources and search strategy

Different sources of data (online databases or libraries) and key terms/keywords to be used for querying each selected database are defined in this sub-phase. Hence, papers to be used throughout our review process were searched from six (6) well-known databases including IEEE Xplore,[5] ACM Digital Library,[6] ScienceDirect,[7] SpringerLink,[8] Web of Science,[9] and Google Scholar.[10] We have also searched papers from top international computer security conferences highlighted in Gu (2022). These databases and conferences include up-to-date research works from a wide range of literature on a particular research field or discipline. Additionally, the selected databases were the same source of information for similar survey papers in the discipline. For example, they have been used in previous works presented in Ullah et al. (2018), Suaboot et al. (2020a), Meijin et al. (2022) and Li et al. (2017). After choosing the databases, we have defined the search terms that are used to search papers in each database. We have applied the Boolean expression-based approach that allows us to combine search terms with expressions. These expressions include "AND", or "OR". Hence, the following search terms were used to query each of the above databases. ("machine learning" OR "deep learning" OR "Windows" OR "static analysis" OR "behaviour analysis" OR "hybrid analysis" AND ("malware detection" OR "malware classification"). After performing all possible search combinations, we obtained 418 papers to go through the next phase.

4.2.3. Paper selection criteria

This stage involves defining the criteria for paper inclusion and exclusion. Hence, five inclusion criteria were defined to select related papers from the list of all papers obtained from the databases. The inclusion criteria are (1) whether the search terms are mentioned either in the title, abstract, or keywords, (2) malware detection introduced in the paper, (3) a new benchmark dataset has been presented in the study, and (4) experimental analysis are presented in the paper (5) the study has covered advances on malware detection. We have also formulated several exclusion criteria that are used to remove/filter out irrelevant studies from the list of papers. The exclusion criteria include:

- Duplicate papers were removed
- All papers that are not written in English were excluded
- Papers whose full text is not available
- Posters/short notes and editorials
- Studies that are not related to Windows malware detection were excluded
- Exclude studies that did not use ML or DL algorithms to build malware detection techniques

All papers that fulfilled the inclusion criteria were included in the set of papers to undergo the next phase while papers that did not fulfil them were directly excluded from the list. Note that this process was performed in order to determine relevant papers to be used to answer all RQs. A total number of 320 papers were selected from the list after this stage.

Table 3
Quality assessment criteria.

#	Defined quality assessment criteria
1	Are the goals of the study clearly stated?
2	Are the categories of malware detection techniques clearly explained?
3	Are malware deployment methods well discussed?
4	Did the study provide details of the dataset?
5	Is the proposed method clearly explained?
6	Are the experimental evaluations clearly presented and discussed?
7	Did the study clearly state performance metrics?
8	Does the research study contribute to this SLR?

4.2.4. Defining quality assessment criteria

This part involves assessing each paper's quality in the selected papers after applying the inclusion and exclusion criteria. Our quality assessment process is based on the work presented in Kitchenham et al. (2009), which specifies how to formulate quality assessment criteria and assign a quality score that specifies whether a particular paper fulfils the assessment criteria or not. The same process was adopted in the previous SLR study in Tummers et al. (2019). Therefore, Table 3 shows our defined quality assessment criteria following these works (Kitchenham et al., 2009; Tummers et al., 2019). We used a Three-Point Scale (Yes, with a score = 1, partially = 0.5, and not covered/No with a score = 0) to assess the quality of each paper. For example, if the study's goal was clearly mentioned in the paper, then we give a score of 1, whereas if it was partially mentioned, a score of 0.5 is given. A score of zero is given when there is no clear objective provided in the study. All papers that scored below 4 were excluded to maintain high-quality papers in the final set. As a result, 219 papers on malware detection based on Windows executable files were selected to be used in the final set of primary papers to review after performing the quality assessment. Please note that the above steps were performed through the collaboration of all authors. That is, the authors have agreed on individual steps of the methodology and inclusion of the papers.

4.2.5. Extracting and synthesizing data

The process for extracting relevant data from each selected study is performed during this phase. Relevant information was extracted based on variables/dimensions presented in Fig. 6, which help to categorize extracted data. After the extraction of data, authors classified all data based on the defined RQs, which resulted in quantitative data synthesized to derive relevant insights from the findings. It is worth mentioning that only RQ1, RQ5, RQ6, and RQ7 require qualitative data instead of quantitative data.

4.3. Phase 3: Reporting the results

We conducted an in-depth SLR, which produced significant results on malware detection in Windows. Thus, the reporting phase summarizes our results/findings and provides answers to all RQs defined in this work. This is very important as answering all RQs ensures

[5] https://ieeexplore.ieee.org/
[6] https://portal.acm.org/
[7] https://www.sciencedirect.com/
[8] https://link.springer.com/
[9] https://www.webofknowledge.com/
[10] https://scholar.google.com/

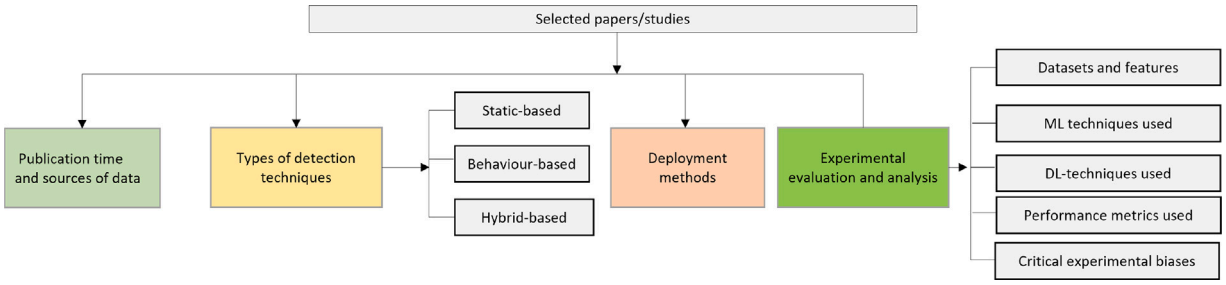

Fig. 6. Different information extracted from the selected papers while performing our SLR.

that our goal has been reached. The results are based on the selected papers published between 2009 and 2022, and our research findings are presented in Sections 5–11. Throughout reporting, we also provide our insights and recommendations relevant to advancing research in this area of malware detection.

5. Types of malware detection and their deployment methods (RQ1)

In this section, we present different types of malware detection techniques including their strengths, weaknesses, and deployment methods. We also provide insights on issues that impede the deployment of malware detection techniques.

5.1. Types of malware detection techniques

Security solutions use different detection techniques to detect and prevent malware programs from achieving their objectives. Following our analysis of existing works, static-based, behaviour-based, and hybrid-based techniques are the main categories of malware detection.

5.1.1. Static signature-based detection technique

This type of malware detection technique mainly relies on a maintained database of known malware static signatures. It detects any suspected program by matching its signatures with those stored in the database (Carlin et al., 2019). Signatures are common characteristics or continuous sequences that are common for certain malware files. Opcode (such as MOV, ADD, SUB, JNZ, JE, JC, and PUSH), hash value (a unique identifier for each file), and information from the executable header (field value, entropy of section, printable strings, size of section), strings and function calls are examples of static signatures that can be extracted from Windows executable (.EXE) files using tools such as IDA Pro disassembler, PeExplorer, Pestudio, FileAlyzer, Exeinfo PE, Hash Generator, PEView, and CFF Explorer (Santos et al., 2013a; Nissim et al., 2014; Singh and Singh, 2021). It is crucial to mention that antivirus (AV) tools depend on signature-based techniques to detect malware. Thus, antivirus software vendors keep updating and refreshing their databases of signatures so that they can accurately keep track of new malware signatures. Signature-based detection techniques can detect known malware with a high detection rate and a low false-positive rate. However, their major disadvantage is that they cannot detect unknown malware as there is no corresponding or similar signature in the database. Moreover, malware development approaches such as polymorphism, obfuscation, metamorphism, and packing can easily evade or bypass signature-based malware detection systems (Catalano et al., 2022; Cesare et al., 2013), making them ineffective for unknown and advanced malware.

5.1.2. Behaviour-based detection technique

Unlike the signature-based detection technique, the behaviour-based technique detects both known and zero-day malware. It uses behavioural activities/patterns to detect malware. Such activities may include but are not limited to deleting, creating, or altering systems file operations. In addition to system file modification, malware can modify network addresses such as DNS, and IP addresses. Analysis tools such as sandboxes (e.g., Cuckoo sandbox), Wireshark, ProcMon, FakeDNS, Regshot, and Process Explorer are used to monitor and capture behavioural characteristics of malware and benign EXE files (Singh and Singh, 2021; Maniriho et al., 2021). Behaviour-based detection techniques flag any deviations from normal activities as malicious, allowing them to detect known and zero-day malware attacks as the detection is based on the program's behaviours instead of static signatures (Abbasi et al., 2022; Li et al., 2022b). Malware behaviour can be captured through dynamic or memory analysis techniques. However, behaviour-based techniques are prone to a high false-positive rate, and they require more training time than signature-based techniques. Different behaviour-based techniques are presented in Sections 7 and 8.

5.1.3. Hybrid detection techniques

Hybrid detection techniques incorporate the functionality of more than one detection technique to detect malware. For example, the work in Susanto and Munawar (2016) has implemented AHMDS, a hybrid malware detection technique that uses both signature-based and behaviour-based techniques to detect malware. A hybrid malware detection system based on deep neural networks for the cloud was proposed in De Paola et al. (2018). Their detection model can deal with big data while detecting different malware variants. The research works in Gupta et al. (2018), Santos et al. (2013b) have also presented hybrid techniques for malware detection. Hybrid techniques can perform well over a single technique, however, they can be computationally expensive.

5.2. Deploying malware detection techniques

In this section, we discuss different techniques for deploying malware detection techniques across organizational computer networks. Accordingly, throughout our review process, we have identified three main categories of deployment methods reported in the existing literature. These categories of deploying malware detection techniques are discussed below and are summarized in Fig. 7. Accordingly, malware detection systems can be deployed using three main approaches, namely, host-based, network-based, and cloud-based approaches.

5.2.1. Host-based malware detection systems (H-MDSs)

Host-based malware detection relies on a security software/program installed on individual devices that detects and prevents each device from malware infections. An H-MDS monitors programs or applications' real-time activities such as running processes and services, system calls, file system activities (e.g., created files, deleted and modified files, changed paths, and newly installed programs), and other unusual activities such as unexpected disk space consumption or loss,

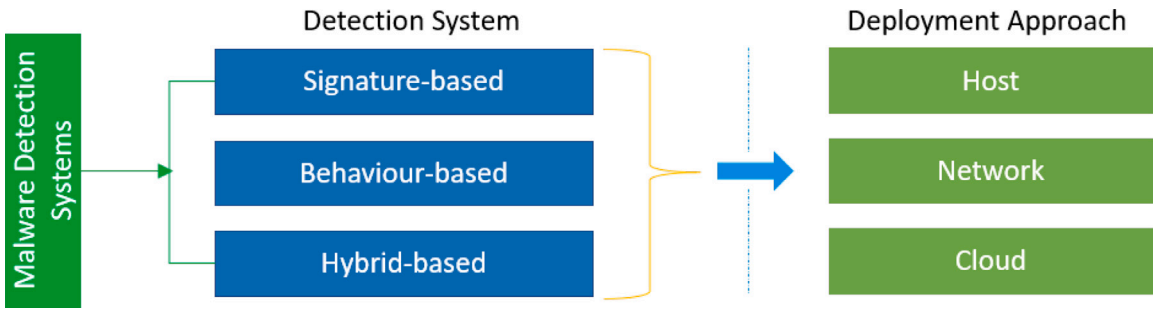

Fig. 7. Types of malware detection techniques and their deployment methods.

excessive use of CPU, and main memory (random access memory), to name a few (Dai and Kuo, 2007; Kolbitsch et al., 2009). They perform routine system scans to detect and remove malware in the system. Programs such as anti-virus (or anti-malware), traditional and modern/next-generation firewalls, and other advanced endpoint detection programs are directly deployed in the device to work at the host level. For example, Windows operating systems (OS) have built-in Windows Defender, a Microsoft anti-malware program (Anon, 2021e). Nonetheless, this type of security solution requires extensive resources as the anti-malware program is typically deployed locally on each device. A host-based malware detection technique has the potential to detect known attacks but may be less effective at detecting and stopping new/unknown malware attacks from infecting the network and devices. Moreover, it is important to mention that as an organization grows, it can be very challenging to scale host-based malware detection solutions. Host-based techniques are also not updated in real-time and it may take longer to get new updates that contain patterns for newly discovered malware.

5.2.2. Network-based malware detection system (N-MDSs)

N-MDSs are deployed on the corporate network or on-premises to monitor all suspicious events that could lead to a compromise (Ollmann, 2020). They can monitor all incoming and outgoing network traffic on a given corporate network and block any suspicious activity that intends to compromise the victim's system. The N-MDSs can be installed on a specialized network router or on another hardware server that monitors the remaining client devices on the network. Some of the N-MDSs are passive (they filter network traffic and generate an alert when unusual traffic is detected and do not block it) while others N-MDSs are active (they filter, detect, and block malicious traffic). An active N-MDS is referred to as a malware prevention system (M-PS) or active N-MDS. An M-PS is often deployed alongside a passive N-MDS. Furthermore, in order to discover advanced malware attacks and increase threat visibility, some organizations use N-MDS in combination with a host-based malware detection system and security information and event management (SIEM) (IBM, 2021), which gathers, aggregates, and analyses multiple security events from different sources.

5.2.3. Cloud-based malware detection systems (cloud-based MDSs)

Cloud-based MDSs are designed and deployed to work in cloud environments to enhance the performance of malware detection for personal computers (laptops and desktops), servers, and mobile devices as well as the overall security of an organization (Yadav, 2019; Aslan et al., 2021). Cloud-based MDSs use powerful detection engines and large databases to detect malware attacks. They rely on remote servers deployed on the cloud to analyze network traffic data and detect potential malware threats. Cloud-based malware detection solutions can detect known and unknown attacks more efficiently. They allow monitoring, tracing, and gathering of behavioural characteristics (traces) of the same malware through multiple executions. This approach also has the potential to identify malicious patterns and behaviours from multiple devices that may reveal an attack. More importantly, cloud-based malware detection solutions do not require extensive resources on each device. Their signature databases and behavioural characteristics are updated in real-time, making them more scalable and efficient than hosted-based solutions (Yadav, 2019). As a result, the cloud anti-malware market is rapidly growing, as many organizations benefit from emerging distributed computing technologies to provide enhanced protection. Bitdefender Total Security, Malwarebytes Nebula, Avast Business Hub, Webroot SecureAnywhere Internet Security, Microsoft Defender for Endpoint, and Kaspersky Security Cloud are examples of cloud-based MDSs. Nevertheless, cloud-based anti-malware programs also have limitations such as privacy and security as sensitive data can be stored and analyzed in the cloud (outside on-premises servers).

5.3. Discussion: From development to deployment

While several malware detection techniques have been proposed and evaluated in the literature, many of these techniques are not deployed in a real-world production environment. Deploying such techniques is very important to ensure their robustness. In this subsection, we discuss the main reasons that prevent many of the existing malware detection techniques from being deployed in real-world environments to allow real-time detection.

- **Limited resources to support deployment:** It takes time and requires many resources (such as computing power and storage) to deploy new technologies such as malware detection techniques in organizational systems. This is because real-world deployment of malware detection techniques is typically cost-effective and requires practical solutions that can be integrated into existing systems. Consequently, many academic researchers lack the resources to deploy their proposed techniques.
- **Lack of scalability:** Many authors have used small datasets generated from controlled/simulated environments such as sandbox-based dynamic analysis environments which generate synthetic datasets that may not represent the characteristics of a malware program in a real-world environment. When such malware detection techniques are tested on datasets from real-world scenarios, they may not effectively be scalable to process high volumes of data extracted from thousands of malware samples. Unfortunately, most of the real-world datasets for developing malware detection techniques are not publicly accessible to security researchers, which limits the scalability of existing techniques.
- **Limited real-world validation:** Malware detection techniques that achieve good performance in a research setting may struggle with the rapidly emerging modern malware landscape. Hackers constantly develop new malware samples that can in many cases widely vary in terms of complexity and sophistication. Thus, without robust validation of existing malware detection techniques in real-world scenarios, there is a high possibility that these techniques proposed in academic research could miss new advanced malware, exposing and making organizations vulnerable to malware attacks. As a result, future works may focus on the development and deployment of the proposed malware detection techniques.

Table 4
Existing public benchmark datasets for training and testing malware detection techniques in Windows.

Reference	Year	Extracted features	No. Benign	No. Malware	Total size
CSDMC2010 dataset (Anon, 2022a; Amer and Zelinka, 2020)	2010	API call sequences	320	68	388
MalImg (Nataraj et al., 2011; Anon, 2022e)	2011	Binary images generated from 25 families of executable files	–	9,339	9,339
Malicia (Nappa et al., 2013)	2013	Binary images of executable files	2,819	11,368	14,187
Ki-dataset by Ki et al. (2015), Anon (2021d)	2015	Sequences of API call	300	23,080	23,380
Microsoft BIG2015 (Ronen et al., 2018)	2015	Static features such as function calls, hash value, strings, hexadecimal representation, symbolic machine code, and metadata	–	More than 20k	–
RISS-Ransomware from Sgandurra et al. (2016), Anon (2022d)	2016	Behavioural features like registry key operations, API calls, directory operations, file system operations, set of file operations performed per file extension, and dropped files extracted from ransomware	942	582	1,524
Paquet-Clouston et al. (2019)	2018	7,222 Bitcoin addresses representing 67 ransomware families	–	–	–
Anderson and Roth (EMBER) (Anderson and Roth, 2018)	2018	Static features from portable executable (PE) header such as general file information (e.g., the file's virtual size, imported and exported functions, list of string, target machine, DLL characteristics, byte entropy, and byte histogram, etc.	550k	550k	110k
Allan and John-dataset (Allan and Ngubiri, 2021)	2018	API call sequences	100	687	787
Ceschin et al. (2018), Anon (2021c)	2018	API call sequences	21,092	29,063	501,55
Dynamic-d (Nunes, 2022)	2018	API call sequences	1000	1000	2000
Kim-D (Kim, 2018) (Amer and Zelinka, 2020)	2018	API call sequences	151	69	220
Oliveira's dynamic dataset (Oliveira and Sassi, 2019b,a)	2019	API call sequences	1,079	42,797	43,876
Oliveira's static dataset (Oliveira, 2019a)	2019	Static features from PE Section Header (hash values, size of data, virtual address, entropy, and virtual size)	1,725	41,568	43,293
Win-API calls by Catak et al. (2020)	2019	API call sequences	–	7,107	7,107
P.Rumao (Rumao, 2022)	2019	Binary hexadecimal and DLL calls	72	301	373
Raw executable grayscale Images (Oliveira, 2019b)	2019	Images of malware and benign generated by converting raw executable byte stream to grayscale images of size 32 by 32 and then flattened to vectors of 1024 bytes for each image	2584	49 376	50,318
MaleVis (Anon, 2022c)	2019	Byte images of 25 classes of malware and 1 class of benign. All images were resized to obtain two different square size resolutions (300 by 300 and 224 by 224). The dataset has 9100 training and 5126 validation images	–	–	14,226
CLaMP (Kumar et al., 2019; Anon, 2022b)	2019	Static features extracted from PE headers. Features include the size of the file, entropy, packer information, section alignment, and size of image	2488	2722	5210
Trinh (Trinh, 2021)	2021	API call sequences	750,000	800,000	1,55M
Saridou et al. (2021)	2021	Images of malicious binaries	–	Over 40,000	–
BODMAS (Yang et al., 2021a)	2021	Different static features	77,142	57,293	134,435
Berrueta (Berrueta et al., 2020)	2021	Behavioural features (network traffic like DNS and TCP data as well as operations performed by the ransomware variant while encrypting shared files' directory)	–	More than 70 ransomware samples	–
Sihwail et al. (2021)	2021	Memory behavioural features including API calls, DLLs, process handles, network traces, injected malicious processes, and privilege feature (process permissions)	966	2,502	3,468
RanSAP (Hirano et al., 2022)	2022	Behavioural features from ransomware and benign files	–	–	–
MalBehavD-V1 (Maniriho et al., 2022)	2022	Sequences of API calls extracted from executable files	1,285	1,285	2,570

(*continued on next page*)

Table 4 (*continued*).

Reference	Year	Extracted features	No. Benign	No. Malware	Total size
Ransomware-dataset (Singh et al., 2022)	2022	Features from process memory images (Read, Read/Write, Read/Execute, Read/Write/Copy, Read/Write/Execute and Read/Write/Execute/Copy)	354	117	471
CIC-MalMem-2022 (Carrier et al., 2022)	2022	Potential injected code, DLLs, processes, handles and API hooks from memory images of obfuscated executable files	29,298	29,298	58,596
Li et al. (2022b)	2022	API call Sequences	2000	2000	4000

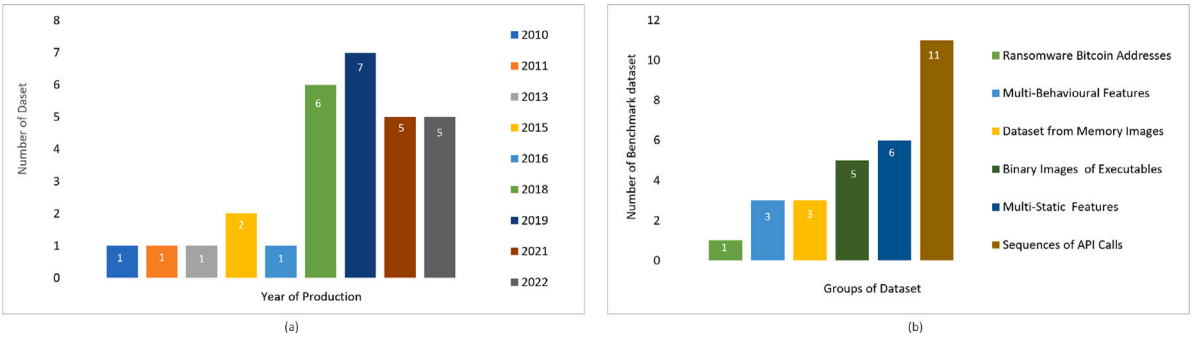

Fig. 8. (a) Number of public benchmark datasets produced per year (from 2011–2022) and (b) Main groups of benchmark dataset.
Source: Data from Table 4.

6. Benchmark datasets and extracted features (RQ2)

This section provides existing datasets for evaluating ML and DL-based malware detection techniques. It also highlights their limitations and provides our recommendations on how to address them.

6.1. Existing datasets for malware detection

A good dataset extracted from benign and malware programs is required to evaluate and validate the performance of malware detection techniques, making the dataset a valuable component in designing and implementing modern anti-malware systems. Nevertheless, most of the datasets used for detecting malware attacks in recent commercial security programs/anti-malware systems are not publicly accessible. This makes acquiring a valuable dataset one of the utmost concerns and in many cases, researchers in the area of malware analysis and detection must create their datasets for training and testing their detection techniques. Moreover, many researchers do not also share their experimental datasets. Consequently, only a few benchmark datasets of benign and malware samples have been made free for public use.

VirusTotal, VirusShare,[11] and other repositories (see Tables 5 and 6) are examples of online repositories where malware samples can be publicly accessed. VirusTotal also offers access to a shared folder for academic researchers, however, researchers must submit an official request in order to be granted access to the folder. VirusTotal also provides academic researchers access via its application programming interface (API).

Sophos and ReversingLabs also contributed to developing malware detection techniques by releasing SoReL-20M,[12] a new public dataset of malware executable program files. There also exist public benchmark datasets extracted from benign and malware executable programs which are freely available for public use. Some of these datasets were shared by Security companies while others were shared by cybersecurity researchers on Github,[13] Kaggle,[14] or other platforms. Accordingly, we have identified 29 public benchmark datasets of benign and malware executable files in Windows. We have grouped these benchmark datasets into six groups based on the types of features (e.g., static or dynamic) extracted from benign and malware executable files. Details on each benchmark dataset can be found in Table 4 which shows when the dataset was released (year), extracted features, and the number of benign and malware files that were analyzed. Table 4 also shows the size of each dataset.

The first group includes 11 datasets of sequences of API calls extracted through the dynamic analysis approach (CSDMC2010 dataset (Anon, 2022a), Ki-dataset by Ki et al. (2015), Allan and John-dataset (Allan and Ngubiri, 2021), Ceschin et al. (Ceschin et al., 2018; Anon, 2021c), Dynamic-d (Nunes, 2022), Kim-D (Kim, 2018), Oliveira's dynamic dataset (Oliveira and Sassi, 2019b,a), Win-API calls by Catak et al. (2020), Trinh (2021), MalBehavD-V1 (Maniriho et al., 2022), and Li et al. (2022b)). The second group has 6 datasets with different static (multi-static) features (Microsoft BIG2015 (Ronen et al., 2018), Anderson and Roth (EMBER) (Anderson and Roth, 2018), Oliveira's static dataset (Oliveira, 2019a), P.Rumao (Rumao, 2022), CLaMP (Kumar et al., 2019), and BODMAS (Yang et al., 2021a)) while the third group has 5 datasets with binary images of executable files (MalImg (Nataraj et al., 2011), Malicia (Nappa et al., 2013), Raw PE grayscale Images (Oliveira, 2019b), MaleVis (Anon, 2022c) and Saridou et al. (2021)). The fourth group has 3 memory analysis-based datasets (Sihwail et al. (2021), Ransomware-dataset (Singh et al., 2022), and CIC-MalMem-2022 (Carrier et al., 2022)) and the fifth group has three datasets with multi-behavioural/dynamic features (RISS-Ransomware from Sgandurra et al. (2016), Berrueta et al. (2020) and RanSAP (Hirano et al., 2022)) while the sixth group has 1 benchmark dataset of ransomware bitcoin addresses (Paquet-Clouston et al. (2019)).

Fig. 8(a) shows the number of datasets produced per year from 2010–2022 (data from Table 4) where several datasets were produced in 2019 (7 datasets) and 2018 (6 datasets). The remaining datasets were produced in 2010 (1 dataset), 2011 (1 dataset), 2013 (1 dataset), 2015 (2 datasets), 2016 (1 dataset), 2021 (5 datasets), and 2022 (5 datasets). On the other hand, Fig. 8(b) shows the above 6 groups of these datasets including the number of datasets in each category where the datasets based on Windows API call sequences were the most produced over other datasets. In Table 4, the dash (-) indicates that the details were not provided in the dataset description.

[11] https://virusshare.com/
[12] https://ai.sophos.com/2020/12/14/
[13] https://github.com/
[14] https://www.kaggle.com/

Table 5
Machine learning-based techniques designed for malware detection in Windows.

Study	Year	Malware collected from	Extracted features	ML algorithm	Detection type		
					Static	Dynamic	Hybrid
Attaluri et al. (2009)	2009	VXHeaven	Opcode sequences	HMM	✓		
Sami et al. (2010)	2010	VxHeavens	Sequences of API calls	RF		✓	
Runwal et al. (2012)	2012	Next Generation Virus Construction Kit (NGVCK)	Opcode sequences	HMM	✓		
Ravi and Manoharan (2012)	2012	VXHeavens	Sequences of API calls (4-grams)	SVM, NB, and DT		✓	
Shabtai et al. (2012)	2012	VXHeavens	Opcode sequences (2-gram features)	RF	✓		
Eskandari et al. (2013)	2013	VirusSign	API calls sequences	DST, SVM (SMO), NB, RT, and RF			✓
Santos et al. (2013b)	2013	VxHeavens	opcode sequences and their frequency, network operations, processes manipulations, threads, file operations, registry keys, and web browsing history	RF, J48, KNN, SVM, and NB	✓	✓	✓
Baysa et al. (2013)	2013	Second Generation Virus Generator (G2) and Next Generation Virus Construction Kit (NGVCK)	Opcode sequences	HMM	✓		
Alazab (2015)	2015	VXHeavens	Sequences of API calls and their frequencies	KNN		✓	
Pluskal (2015)	2015	AVG Company	Running Processes	SVM		✓	
Shijo and Salim (2015)	2015	VirusShare	Printable strings and sequences of API calls	SVM and RF	✓	✓	✓
Pirscoveanu et al. (2015)	2015	VirusShare	Sequences of API calls, DNS requests, mutexes, accessed files, and registry keys	RF		✓	
Annachhatre et al. (2015)	2015	Malicia Project	Opcode sequences	HMM	✓		
Sgandurra et al. (2016)	2016	RISS-Ransomware	File System operations, file extensions, sequences of API calls, registry key operations, directory operations, strings and dropped files	SVM and NB		✓	
Singh et al. (2016)	2016	Malicia Project	Opcode sequences	SVM and HMM	✓		
Huda et al. (2016)	2016	Honeypot and Vxheaven	Static API calls	SVM	✓		
Hansen et al. (2016)	2016	VirusShare and VXHeavens	DLLs and sequences of API calls	RF		✓	
Fan et al. (2016)	2016	Vxheaven	Instruction sequences	All-Nearest Neighbour (improved KNN) classifier	✓		
Vemparala et al. (2016)	2016	–	Dynamic sequences of API calls and opcode sequences	HMM	✓	✓	
Chen et al. (2017a)	2017	VirusShare	API call sequences	RF, SVM, NB, and LR		✓	
Damodaran et al. (2017)	2017	–	API Call sequences and Opcode sequences	HMMs	✓	✓	✓
Huda et al. (2017)	2017	Honeynet Project and VXheavens	Static API calls	SBLR	✓		
Liu et al. (2017)	2017	VX Heavens, ESET NOD32, and Threat Trace Security	Opcodes, import functions, and grayscale images	NB, RF, KNN, GBM, and SVM	✓		
Rathore et al. (2018)	2018	Malicia Project	Opcode frequency	RF	✓		
Shaukat and Ribeiro (2018)	2018	VirusShare	Multi-static and dynamic features from executable files	GBT, LR, SVM with Gaussian-Kernel, RF, and NNs			✓
Jerlin and Marimuthu (2018)	2018	APIMDS-dataset (Ki et al., 2015)	API call sequences	MNB		✓	
Mehnaz et al. (2018)	2018	Open Malware, VirusTotal, Zelster, VXVault, and Malc0de	File type, sequences API call, entropy, and other features	RF, NB, LR, and DT		✓	
Baldwin and Dehghantanha (2018)	2018		Opcode sequences	SVM (SMO)	✓		
Poudyal et al. (2018)	2018	VirusShare, VirusTotal, and ZOO	DLLs and assembly instructions	RF, Adaboost with J48, LR, NB and SVM	✓		
Stiborek et al. (2018)	2018	AMP ThreatGrid	Registry operations, file system changes, network operations, and mutexes	RF		✓	

(*continued on next page*)

Table 5 (*continued*).

Study	Year	Malware collected from	Extracted features	ML algorithm	Detection type		
					Static	Dynamic	Hybrid
Al-rimy et al. (2018)	2018	VirusShare	Sequences of API call	Ensemble of SVM		✓	
Han et al. (2019)	2019	VirusShare	Hybrid sequences of API calls	RF, DT, KNN and XGBoost	✓	✓	✓
Sun et al. (2019)	2019	Vxheaven and Microsoft BIG2015	Opcode sequences	KNN, DT, Adaboost, RF, Bagging, and NNs	✓		
Walker and Sengupta (2019)	2019	–	Sequences of API calls and their frequencies (extracted from Windows PE files)	DT, RF, LDA, LR, KNN, CRT, GNB, and SVM		✓	

(*continued on next page*)

Table 5 (*continued*).

Study	Year	Malware collected from	Extracted features	ML algorithm	Detection type		
					Static	Dynamic	Hybrid
Sihwail et al. (2019)	2019	VirusTotal and Das Malwerk	Sequences of API calls	SVM, NB, DT, RF, and KNN			✓
Rabadi and Teo (2020)	2020	Malshare Website	Sequences of API calls	XGBoost, SVM, DT, and RF		✓	
Ahmed et al. (2020)	2020	VirusTotal and VirusShare	Sequences of API calls	KNN, LR, SVM, RF, and DT		✓	
Amer and Zelinka (2020)	2020	CSDMC2010, Ki-dataset, Allan and John-dataset and Win-API calls	Sequences of API calls	HMM		✓	
Singh and Singh (2020)	2020	VirusShare and Contagio	File operations, modifications from registry key, URLs and network activities, sequences of API calls and printable strings	AdaBoost, Random Forest, DT, GBM and SVM		✓	
Euh et al. (2020)	2020	Malwares	Sequences of API calls, DLLs, Opcode sequences, and WEM (Window Entropy Map) image	Ensemble classifiers (RF, Extra Tree, XGBoost, RT, and Adaboost).	✓		
Suaboot et al. (2020b)	2020	Virusshare	Sequences of API calls	Sub-Curve HMM and RF		✓	
Usman et al. (2021)	2021	Collected through Honeypot	Network activity features	SVM, NB, DT, and Mini Batch K-Means		✓	
Kundu et al. (2021)	2021	EMBER dataset	Static features extracted from executable files	LightGBM	✓		
Kamboj et al. (2022)	2022	Microsoft BIG2015	Various static features	XGBoost, DT, RF, Adaboost, K-Means, and GBM	✓		
Kakisim et al. (2022)	2022	VXHeavens	Opcode sequences	SVM and RF	✓		
Kumar et al. (2022)	2022	MalImg	Global features from binary images	SVM, RF, KNN, NB, and Extra Tree classifier		✓	
Nawaz et al. (2022)	2022	Win-API calls by Catak et al. (2020)	Sequences of API calls	SVM, RF, kNN,NB, J48, NBMText, and SGDText		✓	
Almashhadani et al. (2022)	2022	–	Network traffic features	DT, KNN, SVM, and LDA		✓	
Finder et al. (2022)	2022	VirusTotal	Sequences of API calls and call time interval	SVM, RF, NB and LR		✓	
Park et al. (2022)	2022	KISSA	Multi-static features	XGBoost	✓		
Abbasi et al. (2022)	2022	RISS-Ransomware from Sgandurra et al. (2016)	Various dynamic features	RLR, RF, DT, SVM, and KNN		✓	
Gao et al. (2022b)	2022	EMBER and FFRI	Multi-static features	LightGBM	✓		

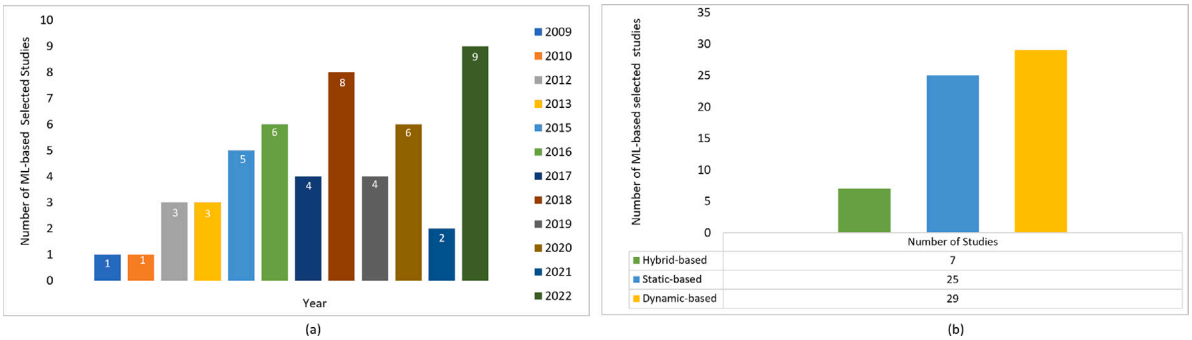

Fig. 9. The number of selected ML-based studies per (a) year of publication and (b) per the type of detection technique (static-based, dynamic-based, and hybrid-based detection). *Source:* Data from Table 5.

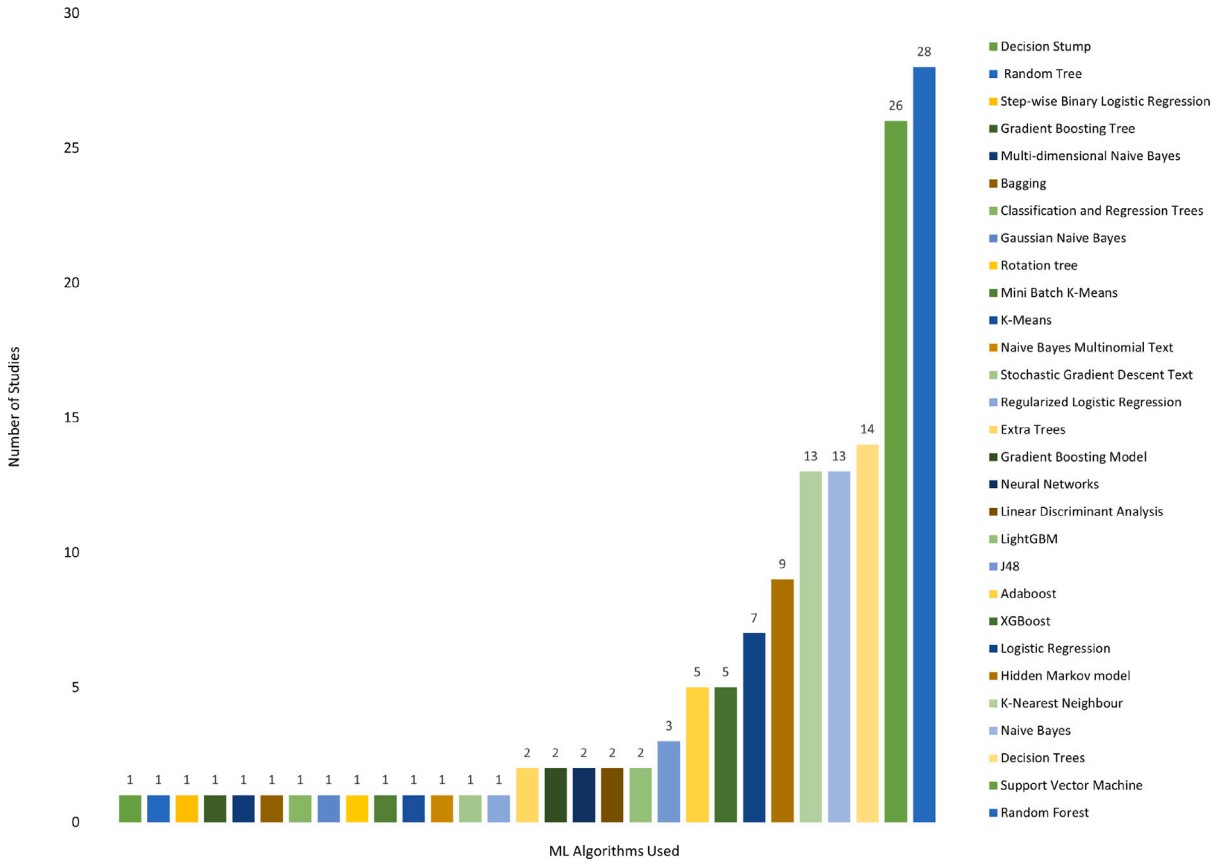

Fig. 10. ML algorithms and the number of studies in which they have been used.
Source: Data from Table 5.

6.2. Discussion: Our insights on existing datasets

Below we highlight different issues encountered in the existing benchmark datasets for malware detection and provide our recommendations, allowing researchers to improve new datasets.

- **Incomplete details on dataset:** We found that some datasets lack valuable details, making them incomplete. For instance, the dataset in Nataraj et al. (2011) did not provide benign images of executable files. Another example can be found in Catak et al. (2020), where the authors have only shared a benchmark dataset of API call sequences that only represent malicious executable files. The description of datasets in Ronen et al. (2018), Paquet-Clouston et al. (2019), Anon (2022c), Berrueta et al. (2020) and Hirano et al. (2022) is also incomplete. Consequently, providing a full description of the dataset must be considered for new dataset generation as failure can lead to inconsistent experimental results.
- **Temporal split of malware samples:** Many of the existing datasets were shared without a timestamp of each malware file. This challenges the time-based split (also called temporal split) for training and testing samples. Moreover, most of the existing datasets are imbalanced (refer to Table 4) where malware samples significantly outnumber benign samples. Unfortunately, such datasets are unrealistic, considering the experimental biases discussed in Section 10.
- **Dataset of API call sequences:** We discovered that the datasets of API call sequences were more produced than other datasets in Windows malware detection, indicating their potential in distinguishing benign from malware executable files.
- **Hybrid benchmark dataset:** We did not come across any hybrid dataset with static, dynamic, or memory features. Thus, we expect this type of dataset to be produced in future works as they are valuable for building hybrid malware detection techniques.
- **Dataset size:** Some datasets in Table 4 have a small number of benign and malware samples, making them unsuitable for evaluating DL-based malware detection techniques as DL algorithms require big datasets when training them to avoid overfitting and underfitting. As a solution to this challenge, data augmentation techniques (Shorten and Khoshgoftaar, 2019) can be used to extend the size of such small datasets. In addition, they can also be extended by adding new features extracted from recent malware variants.
- **Dataset maintenance:** Most of the datasets in Table 4 are not regularly updated to include features extracted from newly discovered malware. Thus, keeping these datasets updated is required to allow the detection of new malware.

7. ML-based malware detection techniques (RQ3)

This section discusses emerging ML techniques for malware detection. It also provides further discussion and our insights on the results in Section 7.2.

7.1. ML-based malware detection techniques

Over the past few years, machine learning algorithms have been widely applied in cybersecurity to detect and classify malware threats. They have achieved significant results, making them one of the prominent solutions to address the challenge of malware threats, which have been devastating many enterprises globally. ML algorithms highly depend on the extracted features representing malware and benign files to detect known and zero-day malware attacks when they are well-trained. In particular, there have been significant works on the use of ML algorithms to design advanced malware detection techniques. Hence, different state-of-the-art studies that used ML algorithms have

been reviewed, and the results are reported in this section. Accordingly, Table 5 summarizes ML-based techniques compiled from the existing works.

As presented in Table 5, each study was carefully reviewed and analyzed based on our SLR guidelines presented in Section 4. Thus, valuable information such as the year of publication, the dataset/malware repository, feature representation, ML algorithms, and the category of each detection technique (static, dynamic, or hybrid-based technique) were extracted. Fig. 9(a) shows the number of selected studies per year (from 2009 to 2022) while Fig. 9(b) shows the number of studies per the type of malware detection. Only one ML-based study was observed in 2009, while many studies were found in 2022, showing continuous interest in ML-based techniques for malware detection in Windows. As shown in Fig. 9(b), among the selected studies, 25 studies are static-based techniques, 29 are dynamic-based techniques, and only seven studies are hybrid-based techniques. It is also worth noting that several ML algorithms have been used in existing work and in many cases, more than one algorithm was used in one study.

Fig. 10 depicts different ML algorithms that were identified and it also shows the number of studies in which the algorithm has been used to implement malware detection techniques. As shown in Fig. 10, these ML algorithms include Decision Stump (DST), Random Tree (RT), Stepwise Binary Logistic Regression (SBRL), Gradient Boosting Tree (GBT), Multi-dimensional Naive Bayes (MNB), Bagging (also called Bagging ensemble), Classification and Regression Trees (CRT), Gaussian Naive Bayes (GNB), Extra Trees (EXT), Rotation Tree (R-tree), K-Means, Mini Batch K-Means, Naive Bayes Multinomial Text (NBMText), Stochastic Gradient Descent Text (SDGText), Regularized Logistic Regression (RLR), Gradient Boosting Model (GBM), Traditional Neural Networks (NNs or Simple NN), Linear Discriminant Analysis (LDA), LightGBM, J48 Decision Tree (J48), Adaboost, XGBoost, Logistic Regression (LR), Hidden Markov model (HMM) and sometimes referred to as Hidden Markov Chain, K-Nearest Neighbour (KNN), Naive Bayes (NB), and Decision Trees (DT).

7.2. Discussion: Our insights on the results

- **ML is gaining more popularity:** As shown in Fig. 9(a), the application of ML algorithms in implementing security solutions has been steadily increasing over time, indicating an ongoing research trend in the use of these algorithms to detect malware attacks in Windows.
- **Most popular ML algorithms:** The results of our investigation show that Support Vector Machine and Random Forest are the most widely used ML algorithms in malware detection. These can be seen from the results presented in Fig. 10.
- **Opcode and API call-based techniques:** We found that several studies have relied on features of opcodes and API calls to implement their malware detection techniques. Thus, we expect to see new malware detection techniques based on these features in future works.
- **Dynamic and Hybrid-based techniques:** While the analysis shows high interest in dynamic analysis-based malware detection over static analysis-based detection, hybrid feature-based malware detection techniques were not fully explored in Windows.
- **Experimental Dataset Missing:** Some authors have mentioned features they used in their works, but the datasets used for training and testing were not specified, which can complicate reproducing their experimental results. Providing a detailed description of the dataset is crucial to allow performance comparison which helps to determine the potential of newly proposed malware detection techniques.

8. DL-based malware detection techniques (RQ4)

This section responds to RQ4. We present emerging DL algorithms that were used to implement malware detection techniques in Windows. We also provide our observations and insights on the research trends on DL-based malware detection.

8.1. DL-based malware detection techniques

Artificial intelligence (AI) has made significant progress over the past decades and continues to emerge in various fields such as computer vision and cybersecurity among others. Deep learning is one of the AI-based techniques mainly created based on traditional neural networks (also called artificial neural networks), another subset of artificial intelligence that mimics how the human brain works. DL algorithms employ advanced neural network architectures to extract relevant patterns from data and have been widely applied to solve different problems in machine translation, robotics, and image classification, to mention a few (Aslan and Yilmaz, 2021; Huang and Karnik, 2022). More importantly, the use of DL algorithms has also emerged in the fields of cybersecurity such as malware detection (Huang and Karnik, 2022; Dixit and Silakari, 2021). The general architecture of DL networks is designed with three main layers: the input layer, which receives numerical input of raw features extracted from benign and malware files, and hidden layers that process the input to extract, select, and learn high abstract/relevant features and then perform classification. Finally, the network architecture has one output layer that reveals the predicted outcome, which could be benign or malware (Sahin and Bahtiyar, 2020).

There exist several types of deep learning algorithms such as recurrent neural networks (RNNs), convolutional neural networks (CNN), autoencoders, and feed-forward networks, to name a few. These algorithms have been used for malware detection in Windows with a variety of benign and malware files' feature representations such as sequences of API calls, operation code (opcode), sequences of bytes, grayscale images of binary executable files, and networks behaviours such as contacted domains names (DNS) and malicious IP addresses (Aslan and Yilmaz, 2021; Kavitha and Muruganantham, 2020; Landman and Nissim, 2021). Interestingly, deep learning has several advantages over conventional ML algorithms. With deep learning, high-level features can be automatically extracted and selected from raw data, removing the need to rely on manual feature engineering/experts with domain knowledge of features. DL algorithms can efficiently deal with unstructured/unlabeled data and can handle the emerging big datasets of malware and benign files.

Furthermore, innovative and advanced malware detection approaches can be built with DL algorithms, allowing organizations and individuals to stay one step ahead of cyber attackers. Thus, many efforts have been made to apply DL algorithms to perform automatic feature extraction and detect different malware variants in Windows. For instance, Sophos,[15] is currently delivering DL-based security solutions as a service to protect organizations against emerging malware attacks. Other security companies such as McAfee[16] Microsoft,[17] Crowdstrike,[18] and Avast,[19] to name a few, also use DL algorithms to implement security solutions.

This section reports our findings on DL-based malware detection from the existing literature. We have analyzed and compiled the most relevant studies to give a general picture of the application of deep learning in malware detection. Table 6 summarizes the details from each selected study. Accordingly, it shows the year of publication, sources of executable files, and experimental datasets that were used. Table 6 also provides the DL algorithm (s) that was used and the type of detection technique. As shown in Figure (a) 11, the selected studies span the period from 2015 to 2022, and we did not come across a relevant study based on DL published before 2015. Figure (b) 11 depicts the number of DL-based studies based on the detection type (static-based and dynamic-based techniques).

[15] https://www.sophos.com/en-us/content/deep-learning-cybersecurity
[16] https://www.mcafee.com/blogs/other-blogs/
[17] https://www.microsoft.com/en-us/security/blog/2020/05/08/
[18] https://www.crowdstrike.com/cybersecurity-101/
[19] https://www.avast.com/en-au/technology/ai-and-machine-learning

Table 6
Deep learning-based techniques designed for detecting malware attacks in Windows.

Study	Year	Malware collected from	Extracted features	DL technique	Detection type	
					Static	Dynamic
Saxe and Berlin (2015)	2015	VXHeavens	Byte entropy, string, and other features from EXE metadata	Deep Neural networks (DNNs)	✓	
Athiwaratkun and Stokes (2017)	2017	–	Sequences of API calls	CNN, LSTM and GRU		✓
Salehi et al. (2017)	2017	VirusTotal and VirusShare	Sequences of API calls	Bi-Residual LSTM model		✓
Ni et al. (2018)	2018	Microsoft BIG2015	Images of executable files	CNN	✓	
Vu et al. (2019)	2019	Virusshare	Images of executable files	CNN	✓	
Vinayakumar et al. (2019)	2019	EMBER dataset and files from VirusTotal and VirusShare	String, byte entropy histogram, raw byte histogram, and binary images	DNN, DeepImageMalDetect (CNN-LSTM)	✓	
Liu and Wang (2019)	2019	VirusTotal	Sequences of API calls	GRU, BiGRU, LSTM, Simple RNN and BiLSTM		✓
Sharma et al. (2019)	2019	Microsoft BIG2015 and dataset from Kan et al. (2018)	Sequence of assembly instructions (bi-grams)	CNN	✓	
Vasan et al. (2020)	2020	MalImg and another real-life malware dataset	Images of executable files	InceptionV3, ResNet50, and VGG16	✓	
Jain et al. (2020)	2020	MalImg	Images of executable files	ELM and CNN	✓	
Darem et al. (2021a)	2021	Microsoft dataset 2015	Opcode sequences (unigram, bigram, trigram, and quadrigram level features)	CNN	✓	
Hemalatha et al. (2021)	2021	Microsoft BIG2015, MalImg, Malicia ,and Malevis	Images of executable files	DenseNet	✓	
Tian et al. (2021)	2021	VirusShare	Byte streams of programs	CNN	✓	
Darem et al. (2021b)	2021	VirusTotal	Sequences of API calls	Sequential Deep Learning		✓
Aslan and Yilmaz (2021)	2021	Malevis, MalIng, and Microsoft BIG 2015 dataset	Images of executable files	AlexNet and ResNet125 (with Malevis)	✓	
Bagane et al. (2021)	2021	–	Binary images of benign and malware	CNN, CNN-LSTM, and CNN-BiLSTM	✓	
Wang et al. (2021)	2021	Microsoft BIG 2015 dataset	Opcode sequences	CNN and GRU	✓	
Sewak et al. (2021)	2021	Malicia dataset	Opcode sequences	LSTM	✓	
Huo et al. (2022)	2022	Microsoft BIG 2015	Static features from executable programs	CNN	✓	
Sharma et al. (2022a)	2022	DikeDataset	Images of executable files	Residual Attention Network	✓	
Gibert et al. (2022)	2022	Microsoft BIG 2015 dataset	Metadata information, operation codes unigram, symbols frequency, register features, pixel intensities, etc.	CNN	✓	
Do Xuan and Huong (2022)	2022	Dataset collected from different sources	Behavioural features such as file operations, registry changes and running processes	DGNs		✓
Olani et al. (2022)	2022	–	Images of executable files	CNN		
Falana et al. (2022)	2022	MalImg, MaleVis and authors generated datasets	Images of executable files	CNN and GAN	✓	
Kim and Cho (2022)	2022	Microsoft BIG 2015 dataset	Images of executable files	CNN-LSTM	✓	
Demırcı et al. (2022)	2022	SOREL-20M dataset	Static features from portable executable files	Stacked BiLSTM	✓	
Li et al. (2022b)	2022	VirusShare	Behavioural sequences of API calls	CNN-BiLSTM		✓
Zhu et al. (2022)	2022	ViruseShare	Entropy of bytes stream and grayscale images of executable files	Siamese Neural Network based on DNN, RNN, ResNet50, VGG16, and InceptionV3)	✓	
Maniriho et al. (2022)	2022	VirusTotal	Sequences of API calls	CNN-BiGRU		✓
Hao et al. (2022)	2022	BODMAS and Microsoft BIG2015	Opcode Sequences	CNN	✓	
Kakisim et al. (2022)	2022	VXHeavens	Opcode Sequences	CNN and LSTM	✓	
Ravi et al. (2022)	2022	Microsoft Big-2015 and other datasets	Multi-static features (API calls, PE-Image, PE-Header, and PE-Imports)	Multi-View deep learning (CNN, DNN and LSTM)	✓	✓
Tekerek and Yapici (2022)	2022	Microsoft BIG2015 and DumpWare10	Images of executable files	GAN+DenseNet	✓	

(*continued on next page*)

Table 6 (*continued*).

Study	Year	Malware collected from	Extracted features	DL technique	Detection type	
					Static	Dynamic
Niu et al. (2022)	2022	–	Network traffic featues (time sequence features)	RESNET-LSTM and PARALLEL-LSTM		✓
Demirkıran et al. (2022)	2022	Oliveira's dataset, VirusShare Win-API calls, and VirusShare	Static and dynamic Sequences of API calls	Deep transformer (an ensemble of pre-trained transformer models such as BERT and CANINE)	✓	✓
Qiang et al. (2022)	2022	VirusTotal	Features from network traffic data packets (One-dimensional behavioural byte sequences)	CNN-LSTM-Attention and CNN-BiLSTM	✓	
Li et al. (2022a)	2022	VirusShare	API call sequences and arguments	GNNs		✓
Fascí et al. (2022)	2022	MalImg	Images of executables	GANs and CVGG3	✓	
Lakshmi et al. (2022)	2022	Malimg	Images of executable files	Siamese neural networks based on CNN	✓	
Finder et al. (2022)	2022	VirusTotal	Sequences of API calls and time interval	LSTM		✓
Shaukat et al. (2022)	2022	VirusShare and VXHeaven	Static sequences of API calls	NN-DeepFool (based on adversarial training samples and deep feed-forward neural network)	✓	
Gibert et al. (2022)	2022	Microsoft BIG2015	Multi-static features (Bytes, image, entropy, and opcode-based features)	CNN	✓	
Falana et al. (2022)	2022	MalImg, MaleVis, and Virushare	Images of executable files	Mal-Detect (GANs and Deep CNN)	✓	
Al-Andoli et al. (2022)	2022	BODMAS and Oliveira's static dataset	Various static and dynamic features	DNNs	✓	✓
Kumar and Janet (2022)	2022	MalImg and Microsoft BIG2015	Images of executable files	Basic CNN, VGG16, VGG19, InceptionV3, and ResNet50	✓	
Yuan et al. (2022)	2022	VirusTotal	Opcode sequences	TextCNN	✓	
Chaganti et al. (2022)	2022	Microsoft BIG2015	Images of executable files	EfficientNetB1	✓	
Zou et al. (2022)	2022	MalImg and Microsoft BIG2015	Images of executable files	CNN	✓	
Chen et al. (2022)	2022	–	Sequence of API calls	BiLSTM and TextCNN		✓
Chai et al. (2022)	2022	VirusTotal	Images of executable files	CNN	✓	
Park et al. (2022)	2022	KISA	Multi-static features	CNN	✓	
Li et al. (2022c)	2022	VirusShare and VirusTotal	Sequences of API Calls	Graph convolutional network		✓
Rizvi et al. (2022)	2022	Collected using honeypots, Kaspersky endpoint security and Kippo Malware collector	Multi-static features	Attention-based Neural Network	✓	
Gao et al. (2022a)	2022	BODMAS	Function name, sequences of API calls, and the starting address of the basic block	Graph Isomorphism Network and MLP	✓	
Mallik et al. (2022)	2022	Microsoft BIG 2015 and MalImg	Images of executable files	VGG16 and Stacked BiLSTM	✓	

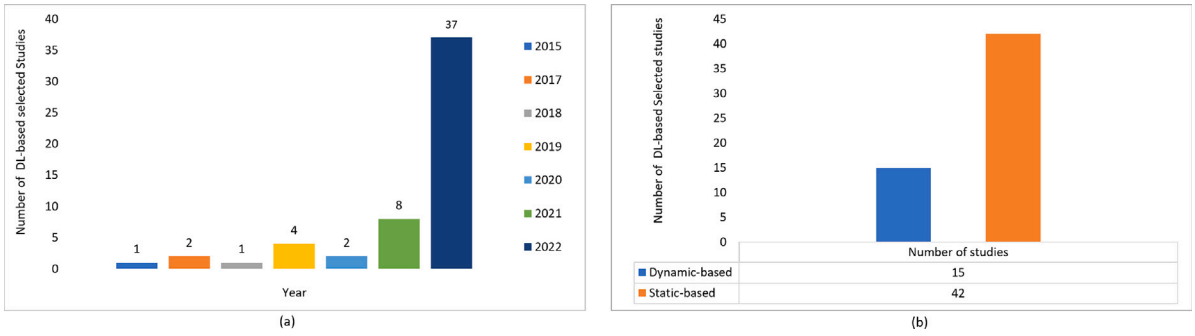

Fig. 11. The number of selected DL-based studies per (a) year of publication and (b) per the type of detection technique (static-based and dynamic-based detection).
Source: Data from Table 6.

As part of our SLR, we have identified different DL algorithms such as Convolutional Neural Network (CNN), TextCNN (specifically designed for text-based feature learning such as API call sequences), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM), Stacked BiLSTM, Parallel-Long Short-Term Memory (P-LSTM), CNN-LSTM-Attention, RESNET-LSTM, VGG3 (a 3 layers-deep convolutional neural network), VGG16 (a 16 layers-deep convolutional neural network), VGG19 (a 19 layers-deep convolutional neural networks), CNN-LSTM, CNN-BiLSTM, Graph Convolutional Network (GCN), Gated Recurrent Unit (GRU), Bidirectional Gated Recurrent Unit (BiGRU), CNN-BiGRU, Deep Neural Networks (DNNs or DNN), Siamese Neural Network (SNN), Bidirectional-Residual LSTM model, Extreme Learning Machine (ELM), Sequential

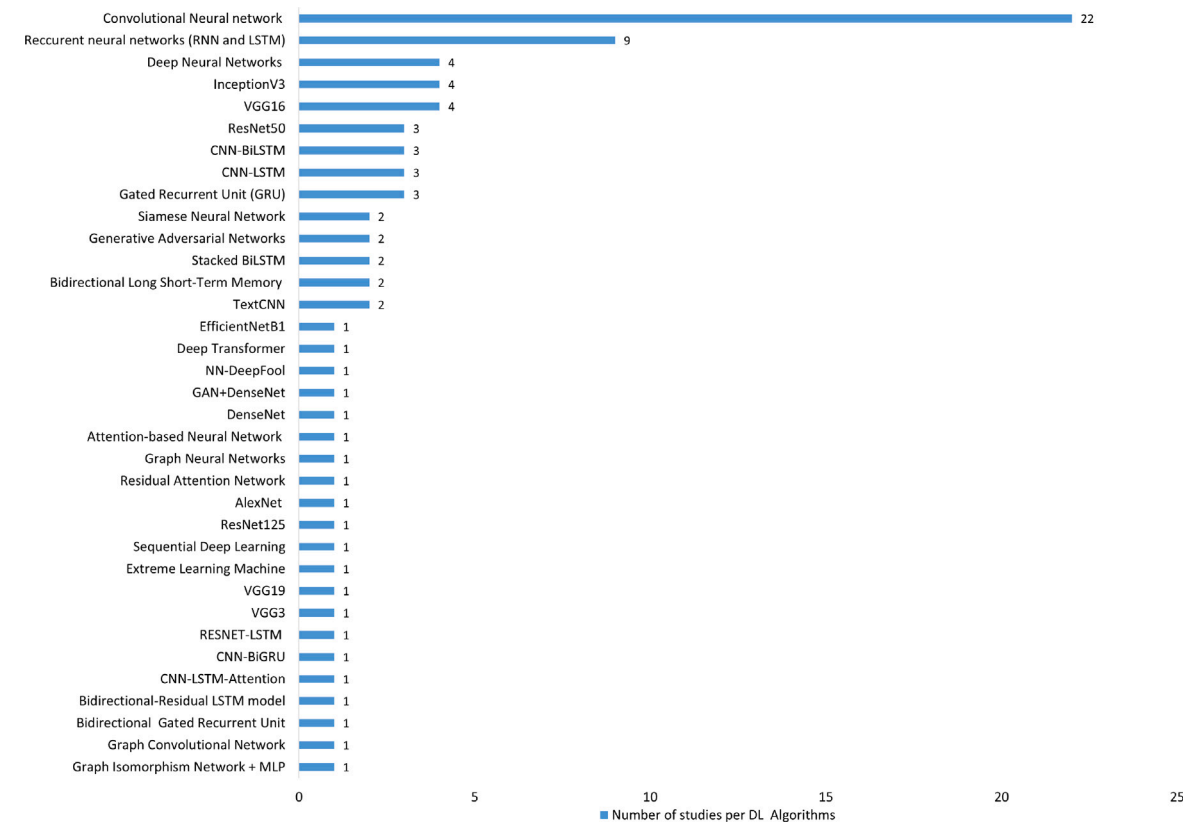

Fig. 12. DL algorithms and the number of studies in which they have been used.
Source: Data from Table 6.

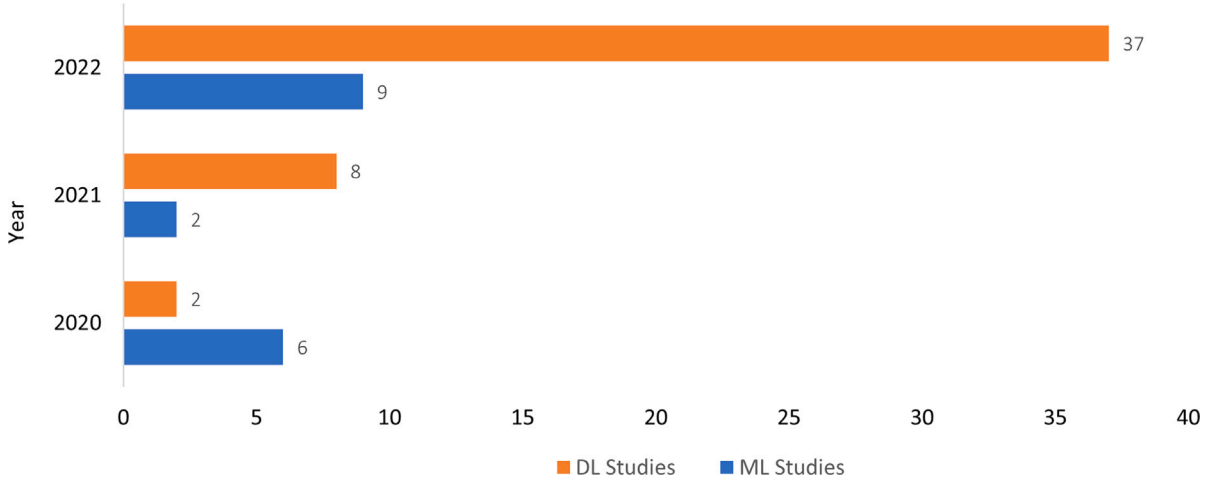

Fig. 13. The use of ML-based studies against DL-based studies over the last three years (between 2020 and 2022).
Source: Data from Tables 5 and 6.

Deep Learning (SDL), ResNet125, AlexNet, Residual Attention Network (RAN), Graph Neural Networks (GNNs), Generative Adversarial Networks (GANs), DenseNet, GAN+DenseNet, NN-DeepFool, ResNet50, Deep Transformer, InceptionV3, Multilayer Perceptron (MLP), Graph Isomorphism Network (GIN), and EfficientNetB1. These algorithms were used to implement malware detection techniques presented in Table 6 and various features such as static signatures, dynamic signatures, and images of binary executable files were used. In addition, all techniques are categorized into two main categories which are static-based and dynamic-based detection techniques. Some of the benchmark datasets presented in Section 6 were used for training and testing the detection techniques while in other works researchers have collected malware samples from various repositories such as VirusShare, VirusTotal, and VXHeavens to construct their datasets.

Fig. 12 (with data from Table 6) summarizes DL algorithms including the number of scientific studies in which they have been used to implement malware detection techniques. Like ML-based studies, some DL-based studies have used more than one DL algorithm. Thus, the analysis in Fig. 12 shows that CNN has been used in 21 studies followed by LSTMs (9 studies) and DNNs (5 studies). Furthermore, it is important to note that many DL-based techniques are observed in 2022, indicating the rapid advancement of DL technologies in malware detection.

		Actual Class	
		Malware (Positive)	Benign (Negative)
Predicted Label /C lass	Malware (Positive)	True Positive (TP)	False Negative (FP)
	Benign (Negative)	False Negative (FN)	True Negative (TN)

Fig. 14. A typical confusion matrix used in the evaluation of a binary classification technique for malware detection).

8.2. Discussion: Our insights on the results

- **The use of ML vs DL-based Algorithms:** There is a high increase in the number of publications on malware detection based on DL algorithms. As depicted in Fig. 13, DL-based techniques were highly produced over ML-based techniques in 2022, indicating ongoing research trends and high interest in DL algorithms for malware detection in Windows desktop devices. Nevertheless, the lowest number of publications based on DL algorithms is observed in 2020. On the other hand, a high number of ML-based studies was found in 2021.
- **Image-based techniques:** As presented in Table 6, malware detection techniques based on static images of Windows executable files are becoming more prevalent in the literature.
- **Most Prevalent DL Algorithms:** During our analysis we discovered that convolutional neural networks and recurrent neural networks are the most popular DL algorithms in Windows malware detection. Thus, these algorithms may continue to emerge in future works.
- **Dynamic/behaviour-based techniques:** As shown in Fig. 11, there is a high research interest in developing DL-based malware detection techniques based on static analysis. Considering that static-based detection techniques are unable to detect obfuscated and other complex malware, the upcoming works should exploit the advantage of Dl algorithms to build many robust dynamic detection techniques as they can detect complex malware.
- **Hybrid-based Techniques:** While most of the existing studies focused on static and dynamic-based malware detection, we did not find hybrid malware detection techniques implemented using DL algorithms. Given their potential, future works should focus on developing new hybrid-based techniques for malware detection. More details on this research limitation are provided in Section 11.8. The next section presents different evaluation metrics identified during our analysis.

9. Performance evaluation metrics for malware detection (RQ5)

This section presents the existing evaluation metrics for malware detection techniques that have been reported in the literature. Moreover, it also provides valuable insights on the use of each metric.

9.1. Performance metrics

We identified various metrics for evaluating both DL and ML-based detection techniques that were measured in the previous studies. For example, a confusion matrix was used to describe the performance of malware classification and detection systems (Rieck et al., 2008). Once computed, the confusion matrix allows the computation of other metrics. The confusion matrix represents the actual class of a given instance (feature vector extracted from a benign or malicious program) and the predicted class/label of that instance. Specifically, it records true positive (TP), false positive (FP), true negative (TN), and false negative (FN). True positive represents the total number of malware instances (applications) that are correctly classified as malware while false positive shows an application such as a file or an active process that is identified and classified as malicious when it is not actually malicious. True negative shows benign or normal programs that are correctly detected as benign. On the other hand, a false negative scenario occurs when a malicious file or process is detected as clean/benign when it is actually malicious. For more clarification, Fig. 14 represents a confusion matrix of a typical binary classification problem for malware detection, where only two classes are defined for the actual and predicted classes, respectively.

There exist other standard metrics such as accuracy (Acc), true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), false negative rate (FNR), precision (PR), recall (RE), F1-score (F1), macro average (macroavg), and weighted average (weightedavg) that are computed based on the confusion matrix. The macro average does not consider class imbalance and can be computed for precision (macroavgPR), recall (macroavgRE), and F1-score (macroavgF1). On the other hand, the weighted average takes into consideration class imbalance and is also computed for recall (weightedavgRE), precision (weightedavgPR), and F1-score (weightedavgF1). Other metrics include Area Under the ROC curve (AUROC), Jaccard coefficient (J), Matthews correlation coefficient (MCC), and Logarithmic loss (Logloss). The AUROC measures the performance by plotting the false positive rate on the X-axis and the true positive rate on the Y-axis and then displays how they vary based on certain decision thresholds. The Jaccard coefficient determines the similarity between malware and benign executable files (e.g. distinguishing malware's opcode sequences from benign ones based on their Jaccard similarity index score). It is worth noting that these metrics are used for both binary classification and multi-class classification tasks. Moreover, Table 7 summarizes all metrics identified in the literature. It also provides details of the mathematical computation of each metric and references for some studies showing where it has been used to examine the performance of malware detection techniques.

9.2. Discussion: Our insights on the use of each metric

Overall, the metrics presented in Table 7 are valuable when implementing malware detection solutions and they provide insights that help organizations enhance the performance of their anti-malware programs. Therefore, in this subsection, we provide additional insights detailing when these metrics should be used.

- **Precision:** It is useful when minimizing the false positives achieved by a malware detection technique. High precision (close or equal to 1) reveals that the detection technique has a low false positive rate, making it more accurate in detecting malware.

Table 7
Different evaluation metrics for malware detection techniques presented in the literature.

Evaluation metric	Computation	Metric used in the studies presented in
Precision (PR)	$PR = \frac{TP}{TP+FP}$	Nawaz et al. (2022), Ahmed et al. (2020), Park et al. (2022), Rizvi et al. (2022), Mallik et al. (2022), Li et al. (2022c), Chen et al. (2022), Chaganti et al. (2022), Yuan et al. (2022), Kumar and Janet (2022), Al-Andoli et al. (2022), Falana et al. (2022), Gao et al. (2022a)
Recall (RE)	$RE = \frac{TP}{TP+FN}$	Nawaz et al. (2022), Amer and Zelinka (2020), Ahmed et al. (2020), Li et al. (2022c), Park et al. (2022), Rizvi et al. (2022), Mallik et al. (2022), Chen et al. (2022), Chaganti et al. (2022), Yuan et al. (2022), Kumar and Janet (2022), Al-Andoli et al. (2022), Falana et al. (2022) Gao et al. (2022a)
True Positive Rate	$TPR = \frac{TP}{TP+FN}$	Amer and Zelinka (2020), Pluskal (2015), Shijo and Salim (2015), Gao et al. (2022b), Shaukat et al. (2022), Kumar and Janet (2022), Falana et al. (2022), Shaukat et al. (2022)
False Positive Rate	$FPR = \frac{FP}{FP+TN}$	Ahmed et al. (2020), Gao et al. (2022b), Pluskal (2015), Shaukat et al. (2022), Kamboj et al. (2022), Shijo and Salim (2015), Amer and Zelinka (2020), Chaganti et al. (2022), Kumar and Janet (2022), Falana et al. (2022)
True Negative Rate	$TNR = \frac{TN}{TN+FP}$	Chen et al. (2017a)
False Negative Rate	$FNR = \frac{FN}{FN+TP}$	Amer and Zelinka (2020) Shaukat et al. (2022)
Accuracy	$Acc = \frac{TP+TN}{TP+TN+FP+FN}$	Nawaz et al. (2022), Kamboj et al. (2022), Park et al. (2022), Abbasi et al. (2022), Shaukat et al. (2022), Amer and Zelinka (2020), Huda et al. (2016), Rizvi et al. (2022), Mallik et al. (2022), Li et al. (2022c), Chai et al. (2022) Chen et al. (2022), Zou et al. (2022), Chaganti et al. (2022), Yuan et al. (2022), Kumar and Janet (2022), Al-Andoli et al. (2022), Falana et al. (2022), Shaukat et al. (2022), Gao et al. (2022a)
F1-score	$F1 = 2 \times \frac{PR \times RE}{PR+RE}$	Ahmed et al. (2020), Park et al. (2022), Rizvi et al. (2022), Kamboj et al. (2022) Mallik et al. (2022), Li et al. (2022c), Chen et al. (2022), Chaganti et al. (2022), Yuan et al. (2022), Kumar and Janet (2022), Al-Andoli et al. (2022), Falana et al. (2022), Gao et al. (2022a)
Macro Average PR	$MacroavgPR = \frac{1}{N}\sum_{i=1}^{N} PR_i$	Wang and Qian (2022), Liu et al. (2021), Chaganti et al. (2022), Tran et al. (2018)
Macro Average RE	$MacroavgRE = \frac{1}{N}\sum_{i=1}^{N} RE_i$	Wang and Qian (2022), Liu et al. (2021), Chaganti et al. (2022), Tran et al. (2018)
Macro Average F1	$MacroavgF1 = \frac{2*MacroavgPR*MacroavgRE}{MacroavgPR+MacroavgRE}$	Wang and Qian (2022), Gibert et al. (2019), Liu et al. (2021), Li and Zheng (2021), Chaganti et al. (2022), Tran et al. (2018), Alageel and Maffeis (2022)
Weighted Average PR	$WeightedavgPR = W_i \sum_{i=1}^{N} PR_i$	Wang and Qian (2022), Liu et al. (2021), Chaganti et al. (2022)
Weighted Average RE	$WeightedavgRE = W_i \sum_{i=1}^{N} RE_i$	Wang and Qian (2022), Liu et al. (2021), Chaganti et al. (2022)
Weighted Average F1	$WeightedavgF1 = \frac{2*WeightedavgPR*WeightedavgRE}{WeightedavgPR+WeightedavgRE}$	Alageel and Maffeis (2022), Chaganti et al. (2022) Wang and Qian (2022), Liu et al. (2021)
Jaccard Coefficient	$J(B,M) = \frac{\lvert B \bigcap M \rvert}{\lvert B \bigcup M \rvert}$	Ling et al. (2021a), Raff and Nicholas (2018)
Matthews Correlation Coefficient	$MCC = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$	Nawaz et al. (2022)
Logarithmic Loss	$Logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j} log_{(p_{i,j})}$	Gibert et al. (2022), Gao et al. (2022b)
Area Under the ROC Curve	$AUROC = \int_0^1 \frac{TP}{TP+FN} d\frac{FP}{TN+FP}$	Zhu et al. (2022), Falana et al. (2022), Kumar and Janet (2022), Chen et al. (2022), Gao et al. (2022b), Rizvi et al. (2022), Goodfellow et al. (2016), Pirscoveanu et al. (2015) Park et al. (2022), Gao et al. (2022a)

- **Recall:** It is used when we want to minimize the misclassified samples (false negative rate) with a low number of false negatives, indicating that a malware detector is more effective in distinguishing malware from benign files.
- **False positive rate:** This metric is used to measure the percentage/rate of false positives out of all actual negative samples. Low FPR means that a malware detection technique has a small number of false alarms, greatly minimizing the time spent on investigating false positives.
- **True negative rate:** This metric indicates the percentage of samples correctly classified as benign out of all actual benign/negative samples. It is valuable when assessing the performance of a malware detection model with a high number of actual negative samples.
- **False negative rate:** This metric determines the proportion of false negatives (malicious files that are not classified as malware but are malware) out of all actual positive samples (malware). It is useful if someone wants to reduce missed detections (misclassified malware samples). A low false negative rate demonstrates that the detection model missed fewer attacks, which decreases the impact and severity of malware infections.
- **Accuracy:** It examines the percentage of correctly classified samples out of all samples and provides the overall performance of a malware detection technique. Nevertheless, accuracy is not always a good metric as it can be biased by datasets with class imbalance.
- **F1-score:** This is the harmonic mean of the detection model's recall and precision. It is used when balancing the trade-off between false positives and false negatives. Essentially, the F1-score is preferable over accuracy when measuring the detection model's performance where the dataset is imbalanced, considering both false negatives and false positives.
- **Matthews correlation coefficient:** It measures the correlation coefficient/difference between predicted outcomes and actual classifications. MCC summarizes the confusion matrix scores, ranging from −1 (total disagreement) to +1 (best agreement), between actual and predicted values. Zero shows no agreement, indicating that the prediction is random based on the actuals. Thus, it is useful when testing binary classifiers with imbalanced datasets.
- **Logarithmic loss:** Also called cross-entropy loss or logarithmic loss, this metric measures the performance of a malware detection model by determining the difference between predicted probabilities and actual values. Log loss can be used for binary and multi-class classification tasks where the malware classification technique assigns a probability to each class for all samples.
- **Area under the ROC curve:** This metric is used to generate a graphical representation of the false positive rate against the true positive rate. It is mostly useful when comparing the performance of different malware classifiers across different thresholds. A higher area

Table 8
Experimental biases (factors) to be considered when building ML and DL-based techniques for detecting malware.

#	Experimental bias	Practical consideration	Reference
1	Generating dataset	Malware samples should be collected from reliable sources and must be collected based on their timestamp to enable time-based split	Mimura (2023), Arp et al. (2022), Mimura and Ito (2022), Rossow et al. (2012)
2	Inaccurate data sampling and labeling	Incorrect labeling and inappropriate samples can lead to inaccurate predictions and can impact the overall performance of the detection technique	Lipton et al. (2018), Arp et al. (2022), Zhu et al. (2020)
3	Sample size	The size of benign samples should be greater than the size of malware samples in the entire dataset	Arp et al. (2022), Mimura (2023)
4	Data snooping	This is observed in a dataset where time dependencies between data samples were ignored during dataset generation. This situation is often referred to as temporal snooping and can lead to temporal bias. Thus, the time-based split of samples should always be considered to prevent malware detection techniques from temporal bias	Arp et al. (2022) Pendlebury et al. (2019)
5	Spatial bias	Spatial bias occurs when malware samples in the test set highly outnumber benign samples. In practice, a realistic portion/ratio of malware samples is required in both test sets,i.e., benign samples should exceed malware samples in the test set to avoid spatial bias	Mimura (2023), Pendlebury et al. (2019), Arp et al. (2022)
6	Selecting parameters	Hyperparameter selection and tuning should be performed properly to avoid inaccurate predictions during training and testing of the ML and DL-based malware detection technique	Sommer and Paxson (2010), Arp et al. (2022)
7	Improper comparison	The performance of a new proposed malware detection technique should be assessed against the previous techniques to determine its improvement. It is also important to use the same dataset and metrics when making comparisons	Arp et al. (2022), Wolpert (1996)
8	Inaccurate performance metrics	Evaluating the performance of malware detection techniques using a single metric such as accuracy can be misleading as it obscures important metrics such as false positives and false negatives. Therefore, several metrics should be measured while assessing the performance of a given detection technique	Arp et al. (2022)
9	Real-world dataset	Malware analysis techniques often collect the behavioural traces of executable files in a controlled analysis environment, and such traces are often different from the ones observed in the wild (collected from a real host). Hence, traces from real hosts should always be preferred to have robust detection techniques. However, it is crucial to mention that getting such a dataset is always challenging and even impossible for security researchers	Avllazagaj et al. (2021)
10	Environmental sensitivity	Malicious executable files exhibit different behaviours when executing in different environments (e.g., A malicious executable file executing in Windows 10 and Windows Server 2016 will exhibit different behavioural characteristics). Therefore, it is better to capture the behaviours of malware while running in multiple execution environments	Avllazagaj et al. (2021)

under the curve shows better performance while a small area shows lower performance.

- **Jaccard coefficient:** It is used to measure the similarity between a set of samples (malware and benign in this case) that the classifier identified as malware and the actual set of malware samples. Its scores range from 0 to 1, with 1 being the perfect similarity score and 0 denoting no similarity at all.
- **Macro average:** It is measured for precision, recall, and F1-score. For instance, it calculates the average recall value over all classes/categories without considering the size of each class (number of samples in each class). Thus, the macro average is preferable when the dataset's classes have equal importance (same number of samples).
- **Weighted average:** In contrast, this metric computes the weighted average for recall, precision, and F1-score. For instance, it computes the weighted precision scores over all classes with the weight being the number of samples in each class. Essentially, computing the weighted averages becomes very useful when classes have different numbers of samples (imbalanced samples between classes) and when someone wants to examine the effect of samples in an imbalanced dataset.

10. Practical experimental bias and recommendation (RQ6)

This section discusses different experimental biases observed in the existing works while carrying out this SLR study. It also highlights our insights on how to avoid them in future experiments.

10.1. Experimental biases encountered in existing studies

Machine learning and deep learning-based malware detection techniques presented in Sections 7 and 8 show promising results, with some techniques achieving high detection accuracy (nearly 100%) (Singh and Singh, 2020; Amer and Zelinka, 2020; Ni et al., 2018). The performance of existing techniques seems to leave no opportunity for improvement and reveals that the problem seems to have been addressed. Nevertheless, various types of experimental biases may greatly affect the overall performance evaluation, causing the detection technique to produce inflated results. Recently, researchers have taken a step forward to address the problem of experimental bias encountered in the existing malware detection techniques (Mimura, 2023; Pendlebury et al., 2019; Arp et al., 2022).

Accordingly, a comprehensive analysis of 30 articles published in top-tier computer security conferences and journals conducted in Arp et al. (2022) revealed that at least three practical pitfalls are encountered in each article, with most of the papers suffering from several experimental biases. The same observation is also observed in many of the studies reviewed in this work, where many of them suffer from spatial bias (refer to Section 10.2 for more details) and other biases. Therefore, this section discusses different types of experimental biases in DL and ML-based malware detection techniques that we encountered while conducting this SLR. We present the types of bias and highlight practical considerations for a realistic/unbiased experiment. Failure to follow such practical aspects may lead to biased experiments and can undermine the detection technique's performance. In light of this, Table 8 summarizes different experimental biases observed in some of the existing works.

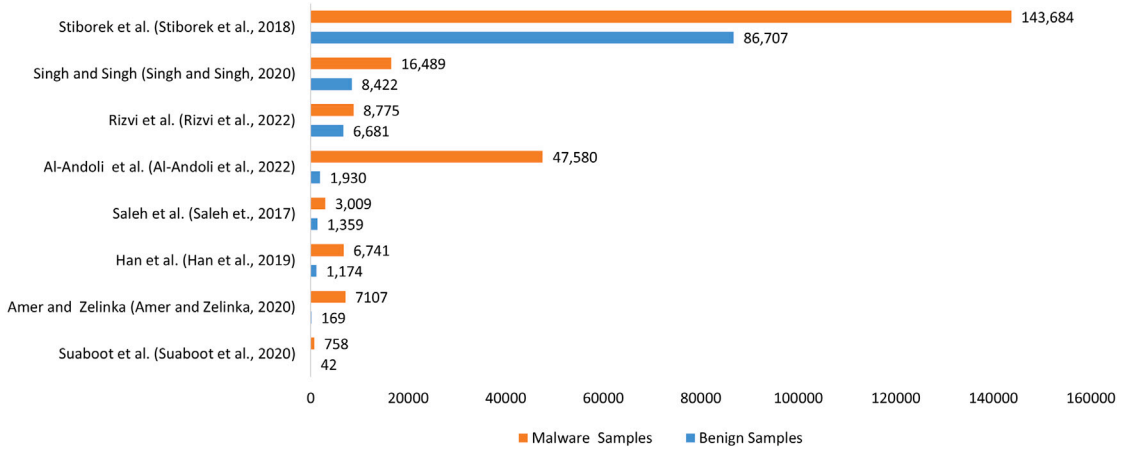

Fig. 15. The example showing the number of benign and malware samples collected for experimental evaluation in some of the previous studies, with the distribution showing that they suffer from spatial bias.

10.2. Discussion: Avoiding experimental bias in future experiments

In several existing works, malware samples extremely exceed the number of benign samples. For instance, the study in Singh and Singh (2020) has used 8422 benign and 16489 malware samples, which resulted in a gap of 8067 samples (16489 - 8422 = 8067). Another example can be taken from the study presented in Sami et al. (2010) which used 2951 benign and 31,869 malware samples, which gives a gap of 28918 samples (31,869-2951 = 31,869). Accordingly, Fig. 15 depicts the distribution of benign and malware samples in some of the existing works, which shows that they suffer from spatial bias where there is no realistic percentage of benign and malware samples in the dataset used. It is crucial to highlight that having a class imbalance in the dataset is not always a problem, especially when the distribution of samples does not violate the practical considerations presented in Table 8. Because only measuring the accuracy precision, and recall can produce biased experimental results, it is important to measure other evaluation metrics such as the F1-score and the weighted average when malware and benign samples are not balanced (which always happens when spatial bias is taken into consideration during testing).

On the other hand, several experiments presented in the existing works were not performed using temporal data, which does not reflect the real-world scenario for malware detection as malware behaviours change over time. Such experiments include the ones presented in Suaboot et al. (2020a), Singh and Singh (2020), Karbab and Debbabi (2019) and Rabadi and Teo (2020), to mention a few. Moreover, the detection techniques based on dynamic analysis have also relied on data from controlled analysis environments (Amer and Zelinka, 2020; Suaboot et al., 2020b; Nawaz et al., 2022; Vemparala et al., 2016). During our analysis, we did not identify any study that used data generated from real hosts while performing experiments. Consequently, a simulated (controlled) analysis environment is the main approach for behaviour-based malware detection in most of the existing works. Thus, new experimental evaluations must be performed taking into consideration all practical and realistic scenarios for malware detection to avoid experimental bias highlighted in Table 8.

11. Research issues and future directions (RQ7)

Despite significant progress in malware detection, existing ML and DL-based constantly face new challenges that limit their performance. Therein, this section provides an in-depth discussion of research issues (challenges) observed in ML and DL-based malware detection in Windows. It also provides future research directions, opening new research avenues in this area.

11.1. High false positive and false negative rate

Both false positive and false negative scenarios occur with many detection techniques including malware endpoint solutions (Microsoft, 2023). When a sophisticated malicious file bypasses a security detection technique, it allows the cybercriminal to carry out the attack, which could result in small or full system breaches, leading to significant service disruption, data theft, ransomware infection, or systems and hardware damage. On the other hand, if a benign file is mistakenly classified as malware in the system (or endpoint device), this can make other applications nonfunctional and can even affect the operating system (Gavrilut et al., 2012). The work in Alahmadi et al. (2022) reported that validating high false positives (false alarms) can be tedious and time-consuming. Cyber attackers can manipulate malware to behave as benign files in order to avoid detection (Yoo et al., 2021), making many malware detection techniques vulnerable to false positives and false negative alarms. Thus, new malware detection techniques must minimize the high number of false positives and false negatives observed in the existing studies.

11.2. Overfitting and underfitting

One of the major/fundamental issues observed in supervised ML and DL techniques is overfitting and underfitting (Azeez et al., 2021), which can lead to poor performance. Overfitting prevents the learning model or classifier from effectively generalizing so that it cannot fit the observed instances of the training dataset and testing set well, i.e., the machine learning model performs well on the training data and achieves poor performance on the test data. This means that the overfitted model cannot discover relevant pieces of information or good data representation in the testing data, which may differ from the training data (Ying, 2019; Brownlee, 2022).

In addition, the overfitted model also tends to memorize all observations/data points including noise in the training set, instead of perfectly learning relevant feature representation from the data. Consequently, it may be complicated to discover the underlying causes behind model overfitting; however, it usually occurs due to three main factors: (1) the classifier's complexity, (2) the presence of noise in the dataset (some features in the dataset are noisy), and (3) the limited size of data (the size of the training data is too small, or the training data has fewer representative data) (Ying, 2019). This results in a situation of a learning model where the noisy data have a great chance to be learned and become the prediction's basis for unseen data. Therefore, a robust ML or DL-based classifier should have the ability to distinguish noise data from good data representation (Paris et al., 2003).

Expansion of the training data (achieved by increasing the size of the training set), network reduction (use techniques like pruning when dealing with decision tree learning algorithms), avoiding early stopping (which prevents the learning model's speed from slowing down at an early stage), regularization (using different regularization techniques to allow the model to fit well on the training set), and using feature selection techniques (which allows to only select relevant features and removing irrelevant (useless) features from the training set) are some of the well-known techniques that can be used to limit or reduce model overfitting (Ying, 2019). Additionally, resampling techniques such as K-Fold cross-validation or keeping back a validation set can also be used to limit overfitting (Brownlee, 2022).

In contrast to overfitting, underfitting refers to a learning situation where the learning model fails to model the training set and cannot also generalize to unseen/new data (IBM, 2022). This creates an underfit malware classification model that is not suitable for prediction as it will achieve poor performance on the training set. Compared to overfitting, underfitting can be easily identified and can be solved by increasing the duration of the training, decreasing regularization, performing feature selection, or trying various learning algorithms (Brownlee, 2022; IBM, 2022).

11.3. Limitations of machine learning algorithms

Machine learning algorithms (classifiers) have different types of limitations. They require large input samples (benign, malware, and potentially unwanted programs) that are correctly labeled to effectively identify malware attacks. Feeding a large quantity of data to the machine learning classifier does not assure that the detection model can correctly detect and classify new malicious files. Therefore, human expert domain knowledge and verification are always required for labeling and manual feature engineering. This process plays a significant role in the performance of machine learning classifiers as even a single incorrect input can lead to incorrect predictions or undermine the prediction to the level of failure. It is worth noting that incorrectly labeled samples can also lead to many false positives (FP) (Kubovič et al., 2018; JARETH, 2022). On the other side, machine learning-based techniques can work well when they are combined with other technologies such as deep learning (Yoo et al., 2021; JARETH, 2022).

Cyber attackers also employ ML algorithms to build sophisticated adversarial cyber attacks (see Section 11.5 for more details). This makes ML algorithms one of the emerging intelligent adversaries used by cyber attackers against ML-based security systems. Steganography is another example of an intelligent adversary used by hackers against ML-based techniques where malicious code is hidden in clean/harmless files such as images (Verma et al., 2022). By deeply exploring the statistical properties of the image and obfuscating the malicious code in the file, the infected file with obfuscated code can fool machine learning classifiers (Verma et al., 2022). Consequently, DL algorithms overcome the limitations of machine learning classifiers and can provide suitable solutions where ML classifiers fail to perform well.

11.4. Advanced automated feature selection

As malware variants against Windows-based systems continue to increase alarmingly (John, 2022; AviraT, 2021), new advanced and efficient DL-based techniques are required to automatically process the emerging high volume of data and generate a good feature representation suitable for training and testing the detection techniques. DL-based techniques with different network architectures will be the best solution for automatic behavioural feature extraction and selection. That is, the use of DL algorithms will continue to emerge as future-generation malware detection techniques will need high-processing DL-based engines with the ability to detect new advanced malware and interpret the predicted outcome. Nevertheless, DL algorithms require graphics processing units (GPUs) to process millions or billions of operations quickly, making advanced and powerful hardware a requirement.

11.5. New attack mechanism against malware detection techniques

While performing this SLR, we discovered that despite the high performance of ML and DL-based malware detection techniques, they could be vulnerable to the threat of adversarial attack, which is becoming more prevalent. Therefore, this subsection discusses what an adversarial attack is, highlights categories of adversarial attacks and frameworks, and then provides defense mechanisms against them.

11.5.1. Adversarial attacks

Machine learning and deep learning technologies are most likely to be used by cyber attackers to empower sophistication in malware development and attack mechanisms. An example is the adversarial machine learning attack (Kianpour and Wen, 2019; Pitropakis et al., 2019; Huang et al., 2019) and deep learning adversarial attack (Kolosnjaji et al., 2018; Papernot et al., 2016b). Adversarial attacks are on the rise and can fool existing malware detection. Cyber attackers can use adversarial samples (inputs) that are intentionally generated to cause the learning technique to make wrong decisions. For example, an AI-based antivirus program of Blackberry Cylance was tricked by security researchers from Skylight Cyber to classify malware files as benign files, indicating how adversarial attacks can circumvent existing security products such as antivirus (Anon, 2022f).

The studies in Ling et al. (2021b), Martins et al. (2020), Demetrio et al. (2022) also show that malware detection techniques based on Windows executable files can be targeted by various adversarial attacks such as feature-space attacks (like data and label poisoning attacks), white-box attacks, black-box attacks, and problem-space attacks with Generative Adversarial Networks, Reinforcement learning, and Gradient-based methods being one of the attack strategies used by cyber attackers. Different adversarial ML and DL-based frameworks have been built. Examples of such frameworks include (L-BFGS) optimization (Szegedy et al., 2013), FGSM (Goodfellow et al., 2014), (JSMA) (Papernot et al., 2016a), Deepfool (Moosavi-Dezfooli et al., 2016), Carlini and Wagner attack (C&W) (Carlini and Wagner, 2017), Generative Adversarial Networks (GANs) (Goodfellow et al., 2020), Wasserstein (WGAN) (Adler and Lunz, 2018), Zeroth-order optimization attack (ZOO) (Chen et al., 2017c), PlausMal-GAN (Won et al., 2022), and Deep Reinforcement Learning-GANs (Randhawa et al., 2022). These frameworks are mostly used to generate perturbed/adversarial samples that can be used to mislead the performance of ML and DL-based malware detection techniques, causing the detection technique to produce incorrect predictions. Fig. 16 summarizes the main types of adversarial attacks against ML and DL-based techniques for malware detection (Ling et al., 2021b; Martins et al., 2020).

11.5.2. Defending against adversarial attacks

By breaking or eluding conventional security endpoints, sophisticated and complex malware attacks pose a serious global security challenge. The existing anti-malware techniques are no longer efficient to defend against these attacks, making the design of new intelligent and predictive malware detection techniques an urgent demand for researchers and security professionals. Accordingly, there has been ongoing research to address the problem of adversarial attacks in ML and DL-based techniques for malware detection and several anti-adversarial learning attacks have been proposed. Such defensive mechanisms include the following techniques.

Adversarial training–One of the defensive mechanisms against adversarial attacks is adversarial training irrespective of the application domains (such as malware detection in Windows, text classification, and image classification, to mention a few). Adversarial training involves re-training ML or DL-based detection techniques using adversarial input samples without or with original samples to enhance their robustness. For instance, existing studies such as Anderson et al. (2018),

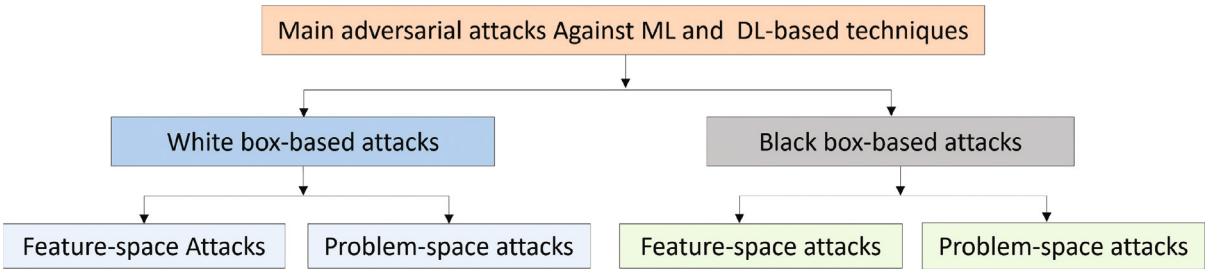

Fig. 16. Main categories of adversarial attacks against Windows executable file-based malware detection techniques.

Hu and Tan (2017), Shaukat et al. (2022) and Chen et al. (2017b) have adopted this approach to address adversarial attacks in Windows.

Other defensive mechanisms–There exist other anti-adversarial learning attack techniques such as defensive distillation (Papernot et al., 2016b), feature squeezing (Xu et al., 2017), universal perturbation approach (Akhtar et al., 2018), MagNet (Meng and Chen, 2017), DAM-ROC (Selvaganapathy and Sadasivam, 2022), and ensemble-based method (Jing et al., 2022; Ahmed et al., 2022). Quiring et al. (2020) also proposed another defensive technique based on executable files.

11.6. Concept drift (data shift)

In machine learning or deep learning, concept drift refers to the situation where the performance of a given detection technique considerably deteriorates over time due to the change in the underlying relationship between data distribution in real-time data (data generated when using the detection technique) and training data (data used for training). The malware detection technique fails to effectively generalize on new samples (Gama et al., 2014). This issue is also known as data shift in the application of ML and DL (Quinonero-Candela et al., 2022). The detection technique can suffer from concept drift due to many factors such as the presence of polymorphic behaviours in malware samples and manipulation of input samples (which causes adversarial drift) (Jordaney et al., 2017).

Generally, in Windows malware detection the behaviours of malware and benign executable files can change over time, causing the executable files to behave differently when running in a production environment (system) (Jordaney et al., 2017; Avllazagaj et al., 2021; Maniriho et al., 2021). This situation can cause poor performance and can greatly degrade the performance of ML and DL-based malware detectors that assume a static relationship between input and output features/variables. In addition, concept drift can affect the detection technique's deployment and scalability (Kuppa and Le-Khac, 2022). Therefore, the detection technique must be kept up-to-date in order to detect or classify new malware attacks. This can be achieved by training and retraining ML or DL-based malware detection techniques using temporal data representing malware and benign file.

Specifically, recent studies revealed that concept drift can be addressed using techniques such as deep transfer learning, evolutionary algorithms, statistical approach, contrastive learning, and time Windows-based sequential approach. For example, Garcıa et al.'s work (García et al., 2023) has proven that transfer learning can handle concept drift in DL-based techniques. Fernando and Komninos (Fernando and Komninos, 2022) have implemented FeSA, a framework that uses genetic algorithm and Information Gain feature selectors to determine and select relevant features that enhance the longevity of ML classifiers for ransomware detection.

Jordaney et al. used a statistical-based approach to build Transcend (Jordaney et al., 2017), a framework that can detect concept drift when it occurs in a malware detection technique during deployment. Their framework can detect concept drift before the detection accuracy decreases to an unsatisfactory level. Barbero et al.'s study has recently proposed TRANSCENDENT (Barbero et al., 2022), an improved framework built based on Transcend (Jordaney et al., 2017), which uses rejection strategies to potentially detect concept drift in malware detection techniques. It is worth mentioning that TRANSCENDENT is also based on a statistical engine, however, new detectors were added to the framework. A new framework that handles concept drift was developed based on contrastive learning in Yang et al. (2021b). A time window-based approach for concept drift detection was recently proposed in Kuppa and Le-Khac (2022) and their method can discover drifted data and can also identify adversarial drifts as they occur.

11.7. New malware evasion mechanisms

Traditional malware infection mechanisms have shifted to new attack chains that are directly performed in memory to avoid being detected by existing conventional security techniques. Cyber attackers also use pre-installed legitimate applications or normal services to compromise the target (legitimate) system (Tancio, 2019). Recently, malware such as ransomware, worms, crypto-miners, and other recent emerging malware such as memory-resident malware (also called fileless malware) use in-memory attack mechanisms to exploit the target (Kumar et al., 2020). Advanced malware evades almost the majority of current antivirus and sandbox security programs (Sharma et al., 2022b).

Hence, scanning the hard drive for detecting malicious activities seems to be inadequate for certain advanced malware. For operating systems (OS) such as Windows, much of the data is never written to disk. Windows OS directly loads executable files from the disk drive into memory; after that, most of the activities occur in memory. Information such as modified runtime code, terminated processes, injected code, API hooked, and data contained in unsaved documents will never be found on disk. Nevertheless, previous studies have not adequately explored memory analysis-based malware detection. In order to protect systems against emerging malware evasion mechanisms, organizations need to discover all malicious indicators of malware programs. Therefore, having sophisticated security approaches that can discover zero-day/ previously unseen malware variants is important. Consequently, performing memory analysis can reveal evasion mechanisms employed by advanced malware (Tancio, 2019).

11.8. Lack of hybrid detection techniques

Some works have presented hybrid malware detection techniques that use static and dynamic analysis features (Ijaz et al., 2019; Shijo and Salim, 2015). Nevertheless, static and dynamic-based methods are only limited to detecting file-based malware attacks and can fail to identify very sophisticated malware such as memory-resident malware programs (Kumar et al., 2020; Saad et al., 2019). In this situation, memory-based malware detection methods become another defensive mechanism that improves the detection (Saad et al., 2019; Sihwail et al., 2021). Unfortunately, the existing literature lacks the development of hybrid malware detection techniques that use static, dynamic, and memory features to detect advanced malware attacks. Such a combination of different malware analysis techniques can exploit these

techniques' strengths and help mitigate severe and hybrid malware attacks. In this regard, the emerging cyberinfrastructures will need these kinds of future-generation security systems to cope with new malware attacks and stay one step ahead of cyber attackers.

12. Conclusion and future work

Malware attacks continue to grow with Windows-based systems being highly targeted by cyber attackers. Machine learning and deep learning techniques are the most effective techniques for detecting malware attacks. This SLR has compiled and analyzed existing public benchmark datasets of malware and benign executable files for Windows. Existing studies have been carefully reviewed to extract and synthesize relevant information such as ML and DL algorithms that were used to build malware detection techniques. Various algorithms have been identified, and the most prevalent ML and DL algorithms for Windows malware detection were highlighted. Different evaluation metrics were presented, and practical experimental biases for building ML and DL techniques were explored including practical considerations. In addition, an in-depth discussion on the current research challenges that affect the development of reliable malware detection techniques in Windows was presented. Recommendations and future direction have been discussed to provide possible research paths for advancing research in Windows malware detection. In future work, we plan to explore current advances in Android malware detection.

CRediT authorship contribution statement

Pascal Maniriho: Conceptualization, Methodology, Formal analysis, 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

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Data availability

No data was used for the research described in the article.

References

Abbasi, M.S., Al-Sahaf, H., Mansoori, M., Welch, I., 2022. Behavior-based ransomware classification: A particle swarm optimization wrapper-based approach for feature selection. Appl. Soft Comput. 121, 108744.

Adler, J., Lunz, S., 2018. Banach Wasserstein gan. Adv. Neural Inf. Process. Syst. 31.

Ahmed, Y.A., Koçer, B., Huda, S., Al-rimy, B.A.S., Hassan, M.M., 2020. A system call refinement-based enhanced minimum redundancy maximum relevance method for ransomware early detection. J. Netw. Comput. Appl. 167, 102753.

Ahmed, U., Lin, J.C.-W., Srivastava, G., 2022. Mitigating adversarial evasion attacks of ransomware using ensemble learning. Comput. Electr. Eng. 100, 107903.

Akhtar, N., Liu, J., Mian, A., 2018. Defense against universal adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3389–3398.

Al-Andoli, M.N., Tan, S.C., Sim, K.S., Lim, C.P., Goh, P.Y., 2022. Parallel deep learning with a hybrid BP-PSO framework for feature extraction and malware classification. Appl. Soft Comput. 131, 109756.

Al-rimy, B.A.S., Maarof, M.A., Prasetyo, Y.A., Shaid, S.Z.M., Ariffin, A.F.M., 2018. Zero-day aware decision fusion-based model for crypto-ransomware early detection. Int. J. Integr. Eng. 10 (6), 82–88.

Alageel, A., Maffeis, S., 2022. EarlyCrow: Detecting APT malware command and control over HTTP (S) using contextual summaries. In: International Conference on Information Security. Springer, pp. 290–316.

Alahmadi, B.A., Axon, L., Martinovic, I., 2022. 99% False positives: A qualitative study of {sOC} analysts' perspectives on security alarms. In: 31st USENIX Security Symposium. USENIX Security 22, pp. 2783–2800.

Alazab, M., 2015. Profiling and classifying the behavior of malicious codes. J. Syst. Softw. 100, 91–102.

Allan, N., Ngubiri, J., 2021. Windows PE API calls for malicious and benign programs. https://www.researchgate.net/publication/336024802_Windows_PE_API_calls_for_malicious_and_benign_programs?channel=doi&linkId=5d8b6a4e92851c33e9395c89&showFulltext=true. (Accessed 16 December 2021).

Almashhadani, A.O., Carlin, D., Kaiiali, M., Sezer, S., 2022. MFMCNS: A multi-feature and multi-classifier network-based system for ransomworm detection. Comput. Secur. 121, 102860.

Amer, E., Zelinka, I., 2020. A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence. Comput. Secur. 92.

Anderson, H.S., Kharkar, A., Filar, B., Evans, D., Roth, P., 2018. Learning to evade static PE machine learning malware models via reinforcement learning. arXiv preprint arXiv:1801.08917.

Anderson, H.S., Roth, P., 2018. Ember: An open dataset for training static pe malware machine learning models. arXiv preprint arXiv:1804.04637.

Annachhatre, C., Austin, T.H., Stamp, M., 2015. Hidden Markov models for malware classification. J. Comput. Virol. Hacking Tech. 11 (2), 59–73.

Anon, 2021a. 5 types of cyber attacks that you are likely to face in 2021 - amvion labs. https://amvionlabs.com/5-types-of-cyberattacks-in-2021/. (Accessed 27 October 2021).

Anon, 2021b. Free software downloads and reviews for Windows, Android, Mac, and iOS – CNET download. https://download.cnet.com/. (Accessed 15 December 2021).

Anon, 2021c. GitHub - fabriciojoc/Brazilian-malware-dataset: Dataset containing thousands of malware and goodware collected in the Brazilian cyberspace over years. https://github.com/fabriciojoc/brazilian-malware-dataset. (Accessed 16 December 2021).

Anon, 2021d. GitHub - leocsato/detector_mw: Optimizer for malware detection. Api calls sequence of benign files are provided.. https://github.com/leocsato/detector_mw. (Accessed 15 December 2021).

Anon, 2021e. Microsoft windows defender antivirus | insightidr documentation. https://docs.rapid7.com/insightidr/microsoft-windows-defender-antivirus/. (Accessed 27 October 2021).

Anon, 2021f. What is a Trojan Virus | Trojan Virus definition | Kaspersky. https://www.kaspersky.com/resource-center/threats/trojans. (Accessed 10 August 2021).

Anon, 2022a. AZSecure-data.org. https://www.azsecure-data.org/. (Accessed 30 December 2022).

Anon, 2022b. Classification of malwares (CLaMP) | Kaggle. https://www.kaggle.com/datasets/saurabhshahane/classification-of-malwares. (Accessed 23 April 2022).

Anon, 2022c. MaleVis dataset home page. https://web.cs.hacettepe.edu.tr/~selman/malevis/. (Accessed 21 April 2022).

Anon, 2022d. Ransomware dataset – RISS. http://rissgroup.org/ransomware-dataset/. (Accessed 17 April 2022).

Anon, 2022e. Signal processing for malware analysis | Vision Research Lab. https://vision.ece.ucsb.edu/research/signal-processing-malware-analysis. (Accessed 17 April 2022).

Anon, 2022f. Skylight cyber | cylance, I kill you!. https://skylightcyber.com/2019/07/18/cylance-i-kill-you/. (Accessed 23 December 2022).

Anon, 2023. Operating globalsystem market share. https://netmarketshare.com/operating-system-market-share.aspx. (Accessed 21 August 2023).

Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K., 2022. Dos and don'ts of machine learning in computer security. In: Proc. of the USENIX Security Symposium.

Aslan, Ö., Ozkan-Okay, M., Gupta, D., 2021. A review of cloud-based malware detection system: Opportunities, advances and challenges. Eur. J. Eng. Technol. Res. 6 (3), 1–8.

Aslan, Ö., Yilmaz, A.A., 2021. A new malware classification framework based on deep learning algorithms. IEEE Access 9, 87936–87951.

Athiwaratkun, B., Stokes, J.W., 2017. Malware classification with LSTM and GRU language models and a character-level CNN. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, pp. 2482–2486.

Attaluri, S., McGhee, S., Stamp, M., 2009. Profile hidden Markov models and metamorphic virus detection. J. Comput. Virol. 5 (2), 151–169.

AviraT, 2021. Q4 and 2020 Malware Threat Report. Tech. Rep..

Avllazagaj, E., Zhu, Z., Bilge, L., Balzarotti, D., Dumitraş, T., 2021. When malware changed its mind: An empirical study of variable program behaviors in the real world. In: 30th USENIX Security Symposium. USENIX Security 21, pp. 3487–3504.

Azeez, N.A., Odufuwa, O.E., Misra, S., Oluranti, J., Damaševičius, R., 2021. Windows PE malware detection using ensemble learning. In: Informatics, vol. 8, (no. 1), Multidisciplinary Digital Publishing Institute, p. 10.

Bagane, P., Joseph, S.G., Singh, A., Shrivastava, A., Prabha, B., Shrivastava, A., 2021. Classification of malware using deep learning techniques. In: 2021 9th International Conference on Cyber and IT Service Management. CITSM, pp. 1–7.

Baldwin, J., Dehghantanha, A., 2018. Leveraging support vector machine for opcode density based detection of crypto-ransomware. In: Cyber Threat Intelligence. Springer, pp. 107–136.

Barbero, F., Pendlebury, F., Pierazzi, F., Cavallaro, L., 2022. Transcending TRANSCEND: Revisiting malware classification in the presence of concept drift. In: 2022 IEEE Symposium on Security and Privacy. SP, pp. 805–823. http://dx.doi.org/10.1109/SP46214.2022.9833659.

Baysa, D., Low, R.M., Stamp, M., 2013. Structural entropy and metamorphic malware. J. Comput. Virol. Hacking Tech. 9 (4), 179–192.
Berrueta, E., Morato, D., Magaña, E., Izal, M., 2020. Open repository for the evaluation of ransomware detection tools. IEEE Access 8, 65658–65669.
Bidoki, S.M., Jalili, S., Tajoddin, A., 2017. PbMMD: A novel policy based multi-process malware detection. Eng. Appl. Artif. Intell. 60, 57–70.
Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M., 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Software 80 (4), 571–583.
Brownlee, J., 2022. Overfitting and underfitting with machine learning algorithms. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/. (Accessed 05 January 2022).
Carlin, D., O'Kane, P., Sezer, S., 2019. A cost analysis of machine learning using dynamic runtime opcodes for malware detection. Comput. Secur. 85, 138–155.
Carlini, N., Wagner, D., 2017. Towards evaluating the robustness of neural networks. In: 2017 Ieee Symposium on Security and Privacy. Sp, Ieee, pp. 39–57.
Carrier, T., Victor, P., Tekeoglu, A., Lashkari, A.H., 2022. Detecting obfuscated malware using memory feature engineering.. In: ICISSP. pp. 177–188.
Catak, F.O., Yazı, A.F., Elezaj, O., Ahmed, J., 2020. Deep learning based sequential model for malware analysis using windows exe API calls. PeerJ Comput. Sci. 6, e285.
Catalano, C., Chezzi, A., Angelelli, M., Tommasi, F., 2022. Deceiving AI-based malware detection through polymorphic attacks. Comput. Ind. 143, 103751. http://dx.doi.org/10.1016/j.compind.2022.103751.
Cesare, S., Xiang, Y., Zhou, W., 2013. Malwise—An effective and efficient classification system for packed and polymorphic malware. IEEE Trans. Comput. 62 (6), 1193–1206.
Ceschin, F., Pinage, F., Castilho, M., Menotti, D., Oliveira, L.S., Gregio, A., 2018. The need for speed: An analysis of Brazilian malware classifiers. IEEE Secur. Privacy 16 (6), 31–41.
Chaganti, R., Ravi, V., Pham, T.D., 2022. Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification. J. Inf. Secur. Appl. 69, 103306.
Chai, Y., Du, L., Qiu, J., Yin, L., Tian, Z., 2022. Dynamic prototype network based on sample adaptation for few-shot malware detection. IEEE Trans. Knowl. Data Eng..
Chang, K., Zhao, N., Kou, L., 2022. A survey on malware detection based on API calls. In: 2022 9th International Conference on Dependable Systems and their Applications. DSA, pp. 464–471.
Checkpoint, 2023. Global analysis of top malicious file types. https://go.checkpoint.com/2022-mid-year-trends/page-global-malware-statistics.php#0. (Accessed on 20 August 2023).
Chen, X., Hao, Z., Li, L., Cui, L., Zhu, Y., Ding, Z., Liu, Y., 2022. CruParamer: Learning on parameter-augmented API sequences for malware detection. IEEE Trans. Inf. Forensics Secur. 17, 788–803.
Chen, Z.-G., Kang, H.-S., Yin, S.-N., Kim, S.-R., 2017a. Automatic ransomware detection and analysis based on dynamic API calls flow graph. In: Proceedings of the International Conference on Research in Adaptive and Convergent Systems. pp. 196–201.
Chen, Z., Roussopoulos, M., Liang, Z., Zhang, Y., Chen, Z., Delis, A., 2012. Malware characteristics and threats on the internet ecosystem. J. Syst. Softw. 85 (7), 1650–1672.
Chen, L., Ye, Y., Bourlai, T., 2017b. Adversarial machine learning in malware detection: Arms race between evasion attack and defense. In: 2017 European Intelligence and Security Informatics Conference. EISIC, pp. 99–106. http://dx.doi.org/10.1109/EISIC.2017.21.
Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.-J., 2017c. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. pp. 15–26.
Cisco-Secure, 2021. Cyber Security Threat Trends: Phishing, Crypto Top the List. Tech. Rep..
Coombs, T., 2018. Artificial Intelligence & Cybersecurity for Dummies. John Wiley & Sons, Inc.
CyberEdge Group, 2021. Cyberthreat Defense Report.
Dai, S.-Y., Kuo, S.-Y., 2007. MAPMon: A host-based malware detection tool. In: 13th Pacific Rim International Symposium on Dependable Computing. PRDC 2007, pp. 349–356.
Damodaran, A., Di Troia, F., Visaggio, C.A., Austin, T.H., Stamp, M., 2017. A comparison of static, dynamic, and hybrid analysis for malware detection. J. Comput. Virol. Hacking Tech. 13 (1), 1–12.
Darem, A., Abawajy, J., Makkar, A., Alhashmi, A., Alanazi, S., 2021a. Visualization and deep-learning-based malware variant detection using opcode-level features. Future Gener. Comput. Syst. 125, 314–323.
Darem, A.A., Ghaleb, F.A., Al-Hashmi, A.A., Abawajy, J.H., Alanazi, S.M., Al-Rezami, A.Y., 2021b. An adaptive behavioral-based incremental batch learning malware variants detection model using concept drift detection and sequential deep learning. IEEE Access 9, 97180–97196.
De Paola, A., Gaglio, S., Re, G.L., Morana, M., 2018. A hybrid system for malware detection on big data. In: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops. INFOCOM WKSHPS, pp. 45–50.
Demetrio, L., Biggio, B., Roli, F., 2022. Practical attacks on machine learning: A case study on adversarial windows malware. IEEE Secur. Privacy 20 (5), 77–85.
Demırcı, D., Şahın, N., Şirlanci, M., Acarturk, C., 2022. Static malware detection using stacked BiLSTM and GPT-2. IEEE Access.
Demirkıran, F., Çayır, A., Ünal, U., Dağ, H., 2022. An ensemble of pre-trained transformer models for imbalanced multiclass malware classification. Comput. Secur. 121, 102846.
Dixit, P., Silakari, S., 2021. Deep learning algorithms for cybersecurity applications: A technological and status review. Comp. Sci. Rev. 39, 100317.
Do Xuan, C., Huong, D., 2022. A new approach for APT malware detection based on deep graph network for endpoint systems. Appl. Intell. 1–20.
Esentire, 2021. Six Ransomware Gangs Claim 290+ New Victims in 2021, Potentially Reaping $45 Million for the Hackers. Tech. Rep..
Eskandari, M., Khorshidpour, Z., Hashemi, S., 2013. HDM-analyser: A hybrid analysis approach based on data mining techniques for malware detection. J. Comput. Virol. Hacking Tech. 9 (2), 77–93.
Euh, S., Lee, H., Kim, D., Hwang, D., 2020. Comparative analysis of low-dimensional features and tree-based ensembles for malware detection systems. IEEE Access 8, 76796–76808.
Falana, O.J., Sodiya, A.S., Onashoga, S.A., Badmus, B.S., 2022. Mal-detect: An intelligent visualization approach for malware detection. J. King Saud Univ.-Comput. Inf. Sci..
Fan, Y., Ye, Y., Chen, L., 2016. Malicious sequential pattern mining for automatic malware detection. Expert Syst. Appl. 52, 16–25.
Fascí, L.S., Fisichella, M., Lax, G., Qian, C., 2022. Disarming visualization-based approaches in malware detection systems. Comput. Secur. 103062.
Fernando, D.W., Komninos, N., 2022. FeSA: Feature selection architecture for ransomware detection under concept drift. Comput. Secur. 116, 102659.
Finder, I., Sheetrit, E., Nissim, N., 2022. Time-interval temporal patterns can beat and explain the malware. Knowl.-Based Syst. 241, 108266.
Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A., 2014. A survey on concept drift adaptation. ACM Comput. Surv. (CSUR) 46 (4), 1–37.
Gao, Y., Hasegawa, H., Yamaguchi, Y., Shimada, H., 2022a. Malware detection by control-flow graph level representation learning with graph isomorphism network. IEEE Access 10, 111830–111841.
Gao, Y., Hasegawa, H., Yamaguchi, Y., Shimada, H., 2022b. Malware detection using LightGBM with a custom logistic loss function. IEEE Access 10, 47792–47804.
Gao, X., Hu, C., Shan, C., Liu, B., Niu, Z., Xie, H., 2020. Malware classification for the cloud via semi-supervised transfer learning. J. Inf. Secur. Appl. 55, 102661.
García, D.E., DeCastro-García, N., Castañeda, A.L.M., 2023. An effectiveness analysis of transfer learning for the concept drift problem in malware detection. Expert Syst. Appl. 212, 118724.
Gavrilut, D., Benchea, R., Vatamanu, C., 2012. Optimized zero false positives perceptron training for malware detection. In: 2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. pp. 247–253. http://dx.doi.org/10.1109/SYNASC.2012.34.
Gibert, D., Mateu, C., Planes, J., Vicens, R., 2019. Using convolutional neural networks for classification of malware represented as images. J. Comput. Virol. Hacking Tech. 15 (1), 15–28.
Gibert, D., Planes, J., Mateu, C., Le, Q., 2022. Fusing feature engineering and deep learning: A case study for malware classification. Expert Syst. Appl. 117957.
Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2020. Generative adversarial networks. Commun. ACM 63 (11), 139–144.
Goodfellow, I.J., Shlens, J., Szegedy, C., 2014. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.
Gorment, N.Z., Selamat, A., Krejcar, O., 2021. A recent research on malware detection using machine learning algorithm: Current challenges and future works. In: International Visual Informatics Conference. Springer, pp. 469–481.
Gu, G., 2022. Security conference ranking and statistic. https://people.engr.tamu.edu/guofei/sec_conf_stat.htm. (Accessed 06 December 2022).
Gupta, S., Bansal, P., Kumar, S., 2018. ULBP-RF: A hybrid approach for malware image classification. In: 5th IEEE International Conference OnParallel, Distributed and Grid Computing. PDGC-2018, IEEE, pp. 115–119.
Han, W., Xue, J., Wang, Y., Huang, L., Kong, Z., Mao, L., 2019. MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics. Comput. Secur. 83, 208–233.
Hansen, S.S., Larsen, T.M.T., Stevanovic, M., Pedersen, J.M., 2016. An approach for detection and family classification of malware based on behavioral analysis. In: 2016 International Conference on Computing, Networking and Communications. ICNC, pp. 1–5.
Hao, J., Luo, S., Pan, L., 2022. EII-MBS: Malware family classification via enhanced adversarial instruction behavior semantic learning. Comput. Secur. 122, 102905.
Hemalatha, J., Roseline, S.A., Geetha, S., Kadry, S., Damaševičius, R., 2021. An efficient densenet-based deep learning model for malware detection. Entropy 23 (3), 344.
Hirano, M., Hodota, R., Kobayashi, R., 2022. RanSAP: An open dataset of ransomware storage access patterns for training machine learning models. Forensic Sci. Int.: Digit. Invest. 40, 301314.

Hu, W., Tan, Y., 2017. Generating adversarial malware examples for black-box attacks based on GAN. arXiv preprint arXiv:1702.05983.

Huang, C., Karnik, A., 2022. The rise of deep learning for detection and classification of malware | McAfee blogs. https://www.mcafee.com/blogs/other-blogs/mcafee-labs/the-rise-of-deep-learning-for-detection-and-classification-of-malware/. (Accessed 04 January 2022).

Huang, X., Ma, L., Yang, W., Zhong, Y., 2021. A method for windows malware detection based on deep learning. J. Signal Process. Syst. 93 (2), 265–273.

Huang, Y., Verma, U., Fralick, C., Infantec-Lopez, G., Kumar, B., Woodward, C., 2019. Malware evasion attack and defense. In: 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops. DSN-W, pp. 34–38.

Huda, S., Abawajy, J., Abdollahian, M., Islam, R., Yearwood, J., 2017. A fast malware feature selection approach using a hybrid of multi-linear and stepwise binary logistic regression. Concurr. Comput.: Pract. Exper. 29 (23), e3912.

Huda, S., Abawajy, J., Alazab, M., Abdollalihian, M., Islam, R., Yearwood, J., 2016. Hybrids of support vector machine wrapper and filter based framework for malware detection. Future Gener. Comput. Syst. 55, 376–390.

Huo, D., Li, X., Li, L., Gao, Y., Li, X., Yuan, J., 2022. The application of 1D-CNN in microsoft malware detection. In: 2022 7th International Conference on Big Data Analytics. ICBDA, pp. 181–187.

IBM, 2021. Security information and event management (SIEM)? | IBM. https://www.ibm.com/topics/siem. (Accessed 27 October 2021).

IBM, 2022. Underfitting: earn how to avoid underfitting, so that you can generalize data outside of your model accurately. https://www.ibm.com/cloud/learn/underfitting. (Accessed 05 January 2022).

Ijaz, M., Durad, M.H., Ismail, M., 2019. Static and dynamic malware analysis using machine learning. In: 2019 16th International Bhurban Conference on Applied Sciences and Technology. IBCAST, pp. 687–691.

Inayat, U., Zia, M.F., Ali, F., Ali, S.M., Khan, H.M.A., Noor, W., 2021. Comprehensive review of malware detection techniques. In: 2021 International Conference on Innovative Computing. ICIC, pp. 1–6.

Jain, M., Andreopoulos, W., Stamp, M., 2020. Convolutional neural networks and extreme learning machines for malware classification. J. Comput. Virol. Hacking Tech. 16 (3), 229–244.

James, A.V., Sabitha, S., 2021. Malware attacks: A survey on mitigation measures. In: Second International Conference on Networks and Advances in Computational Technologies. Springer, pp. 1–11.

JARETH, 2022. The pros, cons and limitations of AI and machine learning in antivirus software - Emsisoft | security blog. https://blog.emsisoft.com/en/35668/the-pros-cons-and-limitations-of-ai-and-machine-learning-in-antivirus-software/. (Accessed 02 January 2022).

Jeon, J., Park, J.H., Jeong, Y.-S., 2020. Dynamic analysis for IoT malware detection with convolution neural network model. IEEE Access 8, 96899–96911.

Jerlin, M.A., Marimuthu, K., 2018. A new malware detection system using machine learning techniques for API call sequences. J. Appl. Secur. Res. 13 (1), 45–62.

Jing, C., Wu, Y., Cui, C., 2022. Ensemble dynamic behavior detection method for adversarial malware. Future Gener. Comput. Syst. 130, 193–206.

John, S.A., 2022. 95% Of new malware threats target windows OS. https://www.dailyhostnews.com/malware-threats-aimed-at-windows. (Accessed 04 December 2022).

Jordaney, R., Sharad, K., Dash, S.K., Wang, Z., Papini, D., Nouretdinov, I., Cavallaro, L., 2017. Transcend: Detecting concept drift in malware classification models. In: 26th USENIX Security Symposium. USENIX Security 17, pp. 625–642.

Kakisim, A.G., Gulmez, S., Sogukpinar, I., 2022. Sequential opcode embedding-based malware detection method. Comput. Electr. Eng. 98, 107703.

Kamboj, A., Kumar, P., Bairwa, A.K., Joshi, S., 2022. Detection of malware in downloaded files using various machine learning models. Egypt. Inf. J..

Kan, Z., Wang, H., Xu, G., Guo, Y., Chen, X., 2018. Towards light-weight deep learning based malware detection. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference, Vol. 01. COMPSAC, pp. 600–609.

Karbab, E.B., Debbabi, M., 2019. MalDy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports. Digit. Invest. 28, S77–S87.

Kavitha, P., Muruganantham, B., 2020. A study on deep learning approaches over malware detection. In: 2020 IEEE International Conference on Advances and Developments in Electrical and Electronics Engineering. ICADEE, pp. 1–5.

Ki, Y., Kim, E., Kim, H.K., 2015. A novel approach to detect malware based on API call sequence analysis. Int. J. Distrib. Sens. Netw. 11 (6), 659101.

Kianpour, M., Wen, S.-F., 2019. Timing attacks on machine learning: State of the art. In: Proceedings of SAI Intelligent Systems Conference. pp. 111–125.

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

Kim, J.-Y., Cho, S.-B., 2022. Obfuscated malware detection using deep generative model based on global/local features. Comput. Secur. 112, 102501.

Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S., 2009. Systematic literature reviews in software engineering–A systematic literature review. Inf. Softw. Technol. 51 (1), 7–15.

Kolbitsch, C., Comparetti, P.M., Kruegel, C., Kirda, E., Zhou, X.-y., Wang, X., 2009. Effective and efficient malware detection at the end host. In: USENIX Security Symposium, vol. 4, (no. 1), pp. 351–366.

Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Giacinto, G., Eckert, C., Roli, F., 2018. Adversarial malware binaries: Evading deep learning for malware detection in executables. In: 2018 26th European Signal Processing Conference. EUSIPCO, pp. 533–537.

Kubovič, O., Košinár, P., Jánošík, J., 2018. Can artificial intelligence power future malware. ESET White Pap..

Kumar, S., Janet, B., 2022. DTMIC: Deep transfer learning for malware image classification. J. Inf. Secur. Appl. 64, 103063.

Kumar, S., Janet, B., Neelakantan, S., 2022. Identification of malware families using stacking of textural features and machine learning. Expert Syst. Appl. 118073.

Kumar, A., Kuppusamy, K., Aghila, G., 2019. A learning model to detect maliciousness of portable executable using integrated feature set. J. King Saud Univ.-Comput. Inf. Sci. 31 (2), 252–265.

Kumar, S., et al., 2020. An emerging threat fileless malware: A survey and research challenges. Cybersecurity 3 (1), 1–12.

Kundu, P.P., Anatharaman, L., Truong-Huu, T., 2021. An empirical evaluation of automated machine learning techniques for malware detection. In: Proceedings of the 2021 ACM Workshop on Security and Privacy Analytics. pp. 75–81.

Kuppa, A., Le-Khac, N.-A., 2022. Learn to adapt: Robust drift detection in security domain. Comput. Electr. Eng. 102, 108239.

Lakshmi, T.S., Govindarajan, M., Sreenivasulu, A., 2022. Malware visual resemblance analysis with minimum losses using siamese neural networks. Theoret. Comput. Sci..

Landman, T., Nissim, N., 2021. Deep-Hook: A trusted deep learning-based framework for unknown malware detection and classification in Linux cloud environments. Neural Netw. 144, 648–685.

Li, L., Bissyandé, T.F., Papadakis, M., Rasthofer, S., Bartel, A., Octeau, D., Klein, J., Traon, L., 2017. Static analysis of android apps: A systematic literature review. Inf. Softw. Technol. 88, 67–95.

Li, C., Cheng, Z., Zhu, H., Wang, L., Lv, Q., Wang, Y., Li, N., Sun, D., 2022a. DMalNet: Dynamic malware analysis based on API feature engineering and graph learning. Comput. Secur. 122, 102872.

Li, X., Li, X., Wang, F., Li, W., Li, A., 2021. A malware detection method based on machine learning and ensemble of regression trees. In: 2021 2nd International Conference on Artificial Intelligence and Information Systems. pp. 1–6.

Li, C., Lv, Q., Li, N., Wang, Y., Sun, D., Qiao, Y., 2022b. A novel deep framework for dynamic malware detection based on API sequence intrinsic features. Comput. Secur. 116, 102686.

Li, C., Zheng, J., 2021. API call-based malware classification using recurrent neural networks. J. Cyber Secur. Mobil. 617–640.

Li, S., Zhou, Q., Zhou, R., Lv, Q., 2022c. Intelligent malware detection based on graph convolutional network. J. Supercomput. 78 (3), 4182–4198.

Ling, Y.T., Sani, N.F.M., Abdullah, M.T., Hamid, N.A.W.A., 2021a. Structural features with nonnegative matrix factorization for metamorphic malware detection. Comput. Secur. 104, 102216.

Ling, X., Wu, L., Zhang, J., Qu, Z., Deng, W., Chen, X., Wu, C., Ji, S., Luo, T., Wu, J., et al., 2021b. Adversarial attacks against windows PE malware detection: A survey of the state-of-the-art. arXiv preprint arXiv:2112.12310.

Lipton, Z., Wang, Y.-X., Smola, A., 2018. Detecting and correcting for label shift with black box predictors. In: International Conference on Machine Learning. PMLR, pp. 3122–3130.

Liu, Y., Wang, Y., 2019. A robust malware detection system using deep learning on API calls. In: 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference. ITNEC, IEEE, pp. 1456–1460.

Liu, L., Wang, B.-s., Yu, B., Zhong, Q.-x., 2017. Automatic malware classification and new malware detection using machine learning. Front. Inf. Technol. Electron. Eng. 18 (9), 1336–1347.

Liu, J., Zhuge, C., Wang, Q., Guo, X., Li, Z., 2021. Imbalance malware classification by decoupling representation and classifier. In: International Conference on Artificial Intelligence and Security. Springer, pp. 85–98.

Mallik, A., Khetarpal, A., Kumar, S., 2022. ConRec: Malware classification using convolutional recurrence. J. Comput. Virol. Hacking Tech. 1–17.

Mane, T., Nimase, P., Parihar, P., Chandankhede, P., 2022. Review of malware detection using deep learning. In: Soft Computing for Security Applications: Proceedings of ICSCS 2021. Springer, pp. 255–262.

Maniriho, P., Mahmood, A.N., Chowdhury, M.J.M., 2021. A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges. Future Gener. Comput. Syst. 130, 1–18.

Maniriho, P., Mahmood, A.N., Chowdhury, M.J.M., 2022. MalDetConv: Automated behaviour-based malware detection framework based on natural language processing and deep learning techniques.

Maniriho, P., Mahmood, A.N., Chowdhury, M.J.M., 2023. API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques. J. Netw. Comput. Appl. 103704.

Martin, L., 2021. Cyber kill chain® | lockheed Martin. https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html. (Accessed on 27 October 2021).

Martins, N., Cruz, J.M., Cruz, T., Abreu, P.H., 2020. Adversarial machine learning applied to intrusion and malware scenarios: A systematic review. IEEE Access 8, 35403–35419.

Mehnaz, S., Mudgerikar, A., Bertino, E., 2018. Rwguard: A real-time detection system against cryptographic ransomware. In: International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, pp. 114–136.

Meijin, L., Zhiyang, F., Junfeng, W., Luyu, C., Qi, Z., Tao, Y., Yinwei, W., Jiaxuan, G., 2022. A systematic overview of android malware detection. Appl. Artif. Intell. 36 (1), 2007327.

Meng, D., Chen, H., 2017. Magnet: A two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 135–147.

Microsoft, 2023. Address false positives/negatives in Microsoft defender for endpoint | Microsoft learn. https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/defender-endpoint-false-positives-negatives?view=o365-worldwide. (Accessed 18 June 2023).

Mimura, M., 2023. Impact of benign sample size on binary classification accuracy. Expert Syst. Appl. 211, 118630.

Mimura, M., Ito, R., 2022. Applying NLP techniques to malware detection in a practical environment. Int. J. Inf. Secur. 21 (2), 279–291.

Mira, F., 2019. A review paper of malware detection using API call sequences. In: 2019 2nd International Conference on Computer Applications & Information Security. ICCAIS, pp. 1–6.

Mohaisen, A., Alrawi, O., Mohaisen, M., 2015. AMAL: High-fidelity, behavior-based automated malware analysis and classification. Comput. Secur. 52, 251–266.

Moosavi-Dezfooli, S.-M., Fawzi, A., Frossard, P., 2016. Deepfool: A simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2574–2582.

Moussaileb, R., Cuppens, N., Lanet, J.-L., Bouder, H.L., 2021. A survey on windows-based ransomware taxonomy and detection mechanisms. ACM Comput. Surv. 54 (6), 1–36.

Nappa, A., Rafique, M.Z., Caballero, J., 2013. Driving in the Cloud: An Analysis of Drive-by Download Operations and Abuse Reporting. In: Proceedings of the 10th Conference on Detection of Intrusions and Malware & Vulnerability Assessment.

Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S., 2011. Malware images: Visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security. pp. 1–7.

Nawaz, M.S., Fournier-Viger, P., Nawaz, M.Z., Chen, G., Wu, Y., 2022. MalSPM: Metamorphic malware behavior analysis and classification using sequential pattern mining. Comput. Secur. 118, 102741.

Ni, S., Qian, Q., Zhang, R., 2018. Malware identification using visualization images and deep learning. Comput. Secur. 77, 871–885.

Nissim, N., Moskovitch, R., Rokach, L., Elovici, Y., 2014. Novel active learning methods for enhanced PC malware detection in windows OS. Expert Syst. Appl. 41 (13), 5843–5857.

Niu, W., Zhou, J., Zhao, Y., Zhang, X., Peng, Y., Huang, C., 2022. Uncovering APT malware traffic using deep learning combined with time sequence and association analysis. Comput. Secur. 120, 102809.

Nunes, M., 2022. Dynamic malware analysis kernel and user-level calls | Zenodo. https://zenodo.org/record/1203289#.YluoQuhBxaT. (Accessed 17 April 2022).

Olani, G., Wu, C.-F., Chang, Y.-H., Shih, W.-K., 2022. Deepware: Imaging performance counters with deep learning to detect ransomware. IEEE Trans. Comput..

Oliveira, A., 2019a. Malware analysis datasets: PE section headers | IEEE DataPort. https://ieee-dataport.org/open-access/malware-analysis-datasets-pe-section-headers. (Accessed 19 August 2021).

Oliveira, A., 2019b. Malware analysis datasets: Raw PE as image | Kaggle. https://www.kaggle.com/datasets/ang3loliveira/malware-analysis-datasets-raw-pe-as-image. (Accessed 17 April 2022).

Oliveira, A., Sassi, R., 2019a. Behavioral malware detection using deep graph convolutional neural networks. TechRxiv.

Oliveira, A., Sassi, R.J., 2019b. Malware analysis datasets: API call sequences | IEEE DataPort. https://ieee-dataport.org/open-access/malware-analysis-datasets-api-call-sequences. (Accessed 19 August 2021).

Ollmann, G., 2020. How to Interpret Network-Based Malware Detection-The Impact of Malware Acquisition and Processing on Network Detection and Threat Classification Systems. Tech. Rep., Vectra Threat Research Lab.

Or-Meir, O., Cohen, A., Elovici, Y., Rokach, L., Nissim, N., 2021. Pay attention: Improving classification of PE malware using attention mechanisms based on system call analysis. In: 2021 International Joint Conference on Neural Networks. IJCNN, pp. 1–8.

Or-Meir, O., Nissim, N., Elovici, Y., Rokach, L., 2019. Dynamic malware analysis in the modern era—A state of the art survey. ACM Comput. Surv. 52 (5), 1–48.

Oz, H., Aris, A., Levi, A., Uluagac, A.S., 2021. A survey on ransomware: Evolution, taxonomy, and defense solutions. arXiv preprint arXiv:2102.06249.

Pachhala, N., Jothilakshmi, S., Battula, B.P., 2021. A comprehensive survey on identification of malware types and malware classification using machine learning techniques. In: 2021 2nd International Conference on Smart Electronics and Communication. ICOSEC, pp. 1207–1214. http://dx.doi.org/10.1109/ICOSEC51865.2021.9591763.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A., 2016a. The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, pp. 372–387.

Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A., 2016b. Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on Security and Privacy. SP, IEEE, pp. 582–597.

Paquet-Clouston, M., Haslhofer, B., Dupont, B., 2019. Ransomware payments in the bitcoin ecosystem. J. Cybersecur. 5 (1), tyz003.

Paris, G., Robilliard, D., Fonlupt, C., 2003. Exploring overfitting in genetic programming. In: International Conference on Artificial Evolution (Evolution Artificielle). Springer, pp. 267–277.

Park, S., Jeon, S., Kim, H.K., 2022. HMLET: Hunt malware using wavelet transform on cross-platform. IEEE Access 10, 124821–124834.

Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L., 2019. {TesseraCT}: Eliminating experimental bias in malware classification across space and time. In: 28th USENIX Security Symposium. USENIX Security 19, pp. 729–746.

Pirscoveanu, R.S., Hansen, S.S., Larsen, T.M.T., Stevanovic, M., Pedersen, J.M., Czech, A., 2015. Analysis of Malware Behavior: Type Classification using Machine Learning. In: 2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment. CyberSA.

Pitropakis, N., Panaousis, E., Giannetsos, T., Anastasiadis, E., Loukas, G., 2019. A taxonomy and survey of attacks against machine learning. Comp. Sci. Rev. 34.

Pluskal, O., 2015. Behavioural malware detection using efficient SVM implementation. In: Proceedings of the 2015 Conference on Research in Adaptive and Convergent Systems. pp. 296–301.

Poudyal, S., Subedi, K.P., Dasgupta, D., 2018. A framework for analyzing ransomware using machine learning. In: 2018 IEEE Symposium Series on Computational Intelligence. SSCI, pp. 1692–1699.

Priyadarshan, P., Sarangi, P., Rath, A., Panda, G., 2021. Machine learning based improved malware detection schemes. In: 2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence). pp. 925–931.

Qamar, A., Karim, A., Chang, V., 2019. Mobile malware attacks: Review, taxonomy & future directions. Future Gener. Comput. Syst. 97, 887–909.

Qiang, W., Yang, L., Jin, H., 2022. Efficient and robust malware detection based on control flow traces using deep neural networks. Comput. Secur. 102871.

Quinonero-Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N.D., 2022. Dataset Shift in Machine Learning. Mit Press.

Quiring, E., Pirch, L., Reimsbach, M., Arp, D., Rieck, K., 2020. Against all odds: Winning the defense challenge in an evasion competition with diversification. arXiv preprint arXiv:2010.09569.

Rabadi, D., Teo, S.G., 2020. Advanced windows methods on malware detection and classification. In: Annual Computer Security Applications Conference. pp. 54–68.

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

Raff, E., Nicholas, C., 2018. Lempel-Ziv Jaccard distance, an effective alternative to ssdeep and sdhash. Digit. Investig. 24, 34–49.

Ramteke, R., Padhye, A., Dutt, A.S., Dholay, S., 2021. Malware detection in banking and financial sector using light gradient boosting model. In: 2021 International Conference on Communication Information and Computing Technology. ICCICT, pp. 1–8.

Randhawa, R.H., Aslam, N., Alauthman, M., Khalid, M., Rafiq, H., 2022. Deep reinforcement learning based evasion generative adversarial network for botnet detection. arXiv preprint arXiv:2210.02840.

Rathore, H., Agarwal, S., Sahay, S.K., Sewak, M., 2018. Malware detection using machine learning and deep learning. In: International Conference on Big Data Analytics. Springer, pp. 402–411.

Ravi, V., Alazab, M., Selvaganapathy, S., Chaganti, R., 2022. A multi-view attention-based deep learning framework for malware detection in smart healthcare systems. Comput. Commun. 195, 73–81.

Ravi, C., Manoharan, R., 2012. Malware detection using windows api sequence and machine learning. Int. J. Comput. Appl. 43 (17), 12–16.

Reshmi, T., 2021. Information security breaches due to ransomware attacks - A systematic literature review. Int. J. Inf. Manag. Data Insights 1 (2).

Rieck, K., Holz, T., Willems, C., Düssel, P., Laskov, P., 2008. Learning and classification of malware behavior. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, pp. 108–125.

Rizvi, S.K.J., Aslam, W., Shahzad, M., Saleem, S., Fraz, M.M., 2022. PROUD-MAL: Static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable. Complex Intell. Syst. 8 (1), 673–685.

Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., Ahmadi, M., 2018. Microsoft malware classification challenge. arXiv preprint arXiv:1802.10135.

Rossow, C., Dietrich, C.J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., Bos, H., Van Steen, M., 2012. Prudent practices for designing malware experiments: Status quo and outlook. In: 2012 IEEE Symposium on Security and Privacy. IEEE, pp. 65–79.

Rumao, P., 2022. Malware executable detection | Kaggle. https://www.kaggle.com/datasets/piyushrumao/malware-executable-detection. (Accessed 17 April 2022).

Runwal, N., Low, R.M., Stamp, M., 2012. Opcode graph similarity and metamorphic detection. J. Comput. Virol. 8 (1), 37–52.

Saad, S., Mahmood, F., Briguglio, W., Elmiligi, H., 2019. Jsless: A tale of a fileless javascript memory-resident malware. In: International Conference on Information Security Practice and Experience. Springer, pp. 113–131.

Sahin, M., Bahtiyar, S., 2020. A survey on malware detection with deep learning. In: 13th International Conference on Security of Information and Networks. pp. 1–6.

Salehi, Z., Sami, A., Ghiasi, M., 2017. MAAR: Robust features to detect malicious activity based on API calls, their arguments and return values. Eng. Appl. Artif. Intell. 59, 93–102.

Sami, A., Yadegari, B., Rahimi, H., Peiravian, N., Hashemi, S., Hamze, A., 2010. Malware detection based on mining API calls. In: SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing. pp. 1020–1025.

Santos, I., Brezo, F., Ugarte-Pedrero, X., Bringas, P.G., 2013a. Opcode sequences as representation of executables for data-mining-based unknown malware detection. Inform. Sci. 231, 64–82.

Santos, I., Devesa, J., Brezo, F., Nieves, J., Bringas, P.G., 2013b. Opem: A static-dynamic approach for machine-learning-based malware detection. In: International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions. pp. 271–280.

Saridou, B., Rose, J., Shiaeles, S., Papadopoulos, B., 2021. 48,240 Malware samples and binary visualisation images for machine learning anomaly detection. http://dx.doi.org/10.21227/vs0r-8s26.

Saxe, J., Berlin, K., 2015. Deep neural network based malware detection using two dimensional binary program features. In: 2015 10th International Conference on Malicious and Unwanted Software. MALWARE, pp. 11–20.

Selvaganapathy, S.G., Sadasivam, S., 2022. Defense against adversarial malware using robust classifier: DAM-ROC. Sādhanā 47 (4), 1–49.

Sewak, M., Sahay, S.K., Rathore, H., 2021. LSTM hyper-parameter selection for malware detection: Interaction effects and hierarchical selection approach. In: 2021 International Joint Conference on Neural Networks. IJCNN, IEEE, pp. 1–9.

Sgandurra, D., Muñoz-González, L., Mohsen, R., Lupu, E.C., 2016. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020.

Shabtai, A., Moskovitch, R., Feher, C., Dolev, S., Elovici, Y., 2012. Detecting unknown malicious code by applying classification techniques on opcode patterns. Secur. Inform. 1 (1), 1–22.

Shah, K., Singh, D.K., 2015. 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, pp. 495–499. http://dx.doi.org/10.1109/ICGCIoT.2015.7380515.

Sharma, R., Deshmukh, S., Mannava, A., Birla, P., 2022a. Deep learning based residual attention network for malware detection in CyberSecurity. In: 2022 6th International Conference on Intelligent Computing and Control Systems. ICICCS, IEEE, pp. 851–856.

Sharma, A., Gupta, B.B., Singh, A.K., Saraswat, V., 2022b. Orchestration of APT malware evasive manoeuvers employed for eluding anti-virus and sandbox defense. Comput. Secur. 115, 102627.

Sharma, A., Malacaria, P., Khouzani, M., 2019. Malware detection using 1-dimensional convolutional neural networks. In: 2019 IEEE European Symposium on Security and Privacy Workshops. EuroS PW, pp. 247–256.

Shaukat, K., Luo, S., Varadharajan, V., 2022. A novel method for improving the robustness of deep learning-based malware detectors against adversarial attacks. Eng. Appl. Artif. Intell. 116, 105461.

Shaukat, S.K., Ribeiro, V.J., 2018. RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning. In: 2018 10th International Conference on Communication Systems Networks. COMSNETS, pp. 356–363.

Shijo, P., Salim, A., 2015. Integrated static and dynamic analysis for malware detection. Procedia Comput. Sci. 46, 804–811.

Shorten, C., Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. J. Big Data 6 (1), 1–48.

Sihwail, R., Omar, K., Ariffin, K.A.Z., 2021. An effective memory analysis for malware detection and classification. CMC-Comput. Mater. Continua 67 (2), 2301–2320.

Sihwail, R., Omar, K., Ariffin, K.A.Z., Al Afghani, S., 2019. Malware detection approach based on artifacts in memory image and dynamic analysis. Appl. Sci. 9 (18).

Sikorski, M., Honig, A., 2012. Practical Malware Analysis: The Hands-on Guide To Dissecting Malicious Software. no starch Press.

Singh, T., Di Troia, F., Corrado, V.A., Austin, T.H., Stamp, M., 2016. Support vector machines and malware detection. J. Comput. Virol. Hacking Tech. 12 (4), 203–212.

Singh, A., Ikuesan, R.A., Venter, H., 2022. Ransomware detection using process memory. arXiv preprint arXiv:2203.16871.

Singh, J., Singh, J., 2020. Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms. Inf. Softw. Technol. 121.

Singh, J., Singh, J., 2021. A survey on machine learning-based malware detection in executable files. J. Syst. Archit. 112, 101861.

Solairaj, A., Prabanand, S.C., Mathalairaj, J., Prathap, C., Vignesh, L.S., 2016. Keyloggers software detection techniques. In: 2016 10th International Conference on Intelligent Systems and Control. ISCO, pp. 1–6.

Sommer, R., Paxson, V., 2010. Outside the closed world: On using machine learning for network intrusion detection. In: 2010 IEEE Symposium on Security and Privacy. IEEE, pp. 305–316.

SOPHOS, 2021a. Threat Report: Navigating Cybersecurity in an Uncertain World. Tech. Rep..

SOPHOS, 2021b. Security Report: The State of Ransomware. Tech. Rep..

SOPHOS, 2021c. Threat Report: Navigating Cybersecurity in an Uncertain World. Tech. Rep..

Stiborek, J., Pevny, T., Rehak, M., 2018. Multiple instance learning for malware classification. Expert Syst. Appl. 93, 346–357.

Suaboot, J., Fahad, A., Tari, Z., Grundy, J., Mahmood, A.N., Almalawi, A., Zomaya, A.Y., Drira, K., 2020a. A taxonomy of supervised learning for idss in scada environments. ACM Comput. Surv. 53 (2), 1–37.

Suaboot, J., Tari, Z., Mahmood, A., Zomaya, A.Y., Li, W., 2020b. Sub-curve HMM: A malware detection approach based on partial analysis of API call sequences. Comput. Secur. 92, 1–15.

Sun, Z., Rao, Z., Chen, J., Xu, R., He, D., Yang, H., Liu, J., 2019. An opcode sequences analysis method for unknown malware detection. In: Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis. pp. 15–19.

Susanto, A., Munawar, A.Z., 2016. AHMDS: Advanced hybrid malware detector system. In: 2016 International Conference on Data and Software Engineering. ICoDSE, pp. 1–6.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R., 2013. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.

Tancio, B., 2019. Hunting for ghosts in fileless attacks | SANS institute. https://www.sans.org/white-papers/38960/. (Accessed 23 October 2022).

Tekerek, A., Yapici, M.M., 2022. A novel malware classification and augmentation model based on convolutional neural network. Comput. Secur. 112, 102515.

Tian, D., Ying, Q., Jia, X., Ma, R., Hu, C., Liu, W., 2021. MDCHD: A novel malware detection method in cloud using hardware trace and deep learning. Comput. Netw. 198, 108394.

Tran, D., Mac, H., Tong, V., Tran, H.A., Nguyen, L.G., 2018. A LSTM based framework for handling multiclass imbalance in DGA botnet detection. Neurocomputing 275, 2401–2413.

Trinh, Q., 2021. 1.55M API import dataset for malware analysis | IEEE DataPort. https://ieee-dataport.org/open-access/155m-api-import-dataset-malware-analysis. (Accessed 19 August 2021).

Tummers, J., Kassahun, A., Tekinerdogan, B., 2019. Obstacles and features of farm management information systems: A systematic literature review. Comput. Electron. Agric. 157, 189–204.

Tuscano, A., Koshy, T.S., 2021. Types of keyloggers technologies–survey. In: ICCCE 2020.ICCCE 2020. Lecture Notes in Electrical Engineering, vol. 698. pp. 11–22.

Ucci, D., Aniello, L., Baldoni, R., 2019. Survey of machine learning techniques for malware analysis. Comput. Secur. 81, 123–147. http://dx.doi.org/10.1016/j.cose.2018.11.001.

Ullah, F., Edwards, M., Ramdhany, R., Chitchyan, R., Babar, M.A., Rashid, A., 2018. Data exfiltration: A review of external attack vectors and countermeasures. J. Netw. Comput. Appl. 101, 18–54.

Usman, N., Usman, S., Khan, F., Jan, M.A., Sajid, A., Alazab, M., Watters, P., 2021. Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics. Future Gener. Comput. Syst. 118, 124–141.

Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B., Zheng, Q., 2020. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. Comput. Netw. 171, 107138.

Vemparala, S., Di Troia, F., Corrado, V.A., Austin, T.H., Stamo, M., 2016. Malware detection using dynamic birthmarks. In: Proceedings of the 2016 ACM on International Workshop on Security and Privacy Analytics. pp. 41–46.

Verma, V., Muttoo, S.K., Singh, V., 2022. Detecting stegomalware: Malicious image steganography and its intrusion in Windows. In: Security, Privacy and Data Analytics. Springer, pp. 103–116.

Verma, A.K., Sharma, S.K., 2021. Malware detection approaches using machine learning techniques- Strategic survey. In: 2021 3rd International Conference on Advances in Computing, Communication Control and Networking. ICAC3N, pp. 1958–1962.

Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Venkatraman, S., 2019. Robust intelligent malware detection using deep learning. IEEE Access 7, 46717–46738.

Vu, D.-L., Nguyen, T.-K., Nguyen, T.V., Nguyen, T.N., Massacci, F., Phung, P.H., 2019. A convolutional transformation network for malware classification. In: 2019 6th NAFOSTED Conference on Information and Computer Science. NICS, pp. 234–239.

Walker, A., Sengupta, S., 2019. Insights into malware detection via behavioral frequency analysis using machine learning. In: MILCOM 2019 - 2019 IEEE Military Communications Conference. MILCOM, pp. 1–6.

Wang, H., Long, H., Wang, A., Liu, T., Fu, H., 2021. Deep learning and regularization algorithms for malicious code classification. IEEE Access 9, 91512–91523.

Wang, Q., Qian, Q., 2022. Malicious code classification based on opcode sequences and textCNN network. J. Inf. Secur. Appl. 67, 103151.

Weisman, S., 2021. What are denial of service (DoS) attacks? DoS attacks explained. https://us.norton.com/internetsecurity-emerging-threats-dos-attacks-explained.html. (Accessed 27 October 2021).

Wolpert, D.H., 1996. The lack of a priori distinctions between learning algorithms. Neural Comput. 8 (7), 1341–1390.

Won, D.-O., Jang, Y.-N., Lee, S.-W., 2022. PlausMal-GAN: Plausible malware training based on generative adversarial networks for analogous zero-day malware detection. IEEE Trans. Emerg. Top. Comput. 1. http://dx.doi.org/10.1109/TETC.2022.3170544.

Xu, W., Evans, D., Qi, Y., 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. arXiv preprint arXiv:1704.01155.

Yadav, R.M., 2019. Effective analysis of malware detection in cloud computing. Comput. Secur. 83, 14–21.

Yadav, D., Kumar, G., Lakshmi Kameshwari, D., Gunjan, V.K., Kumar, S., 2022. Malware techniques and its effect: A survey. In: ICCCE 2021. Springer, pp. 1215–1225.

Yang, L., Ciptadi, A., Laziuk, I., Ahmadzadeh, A., Wang, G., 2021a. BODMAS: An open dataset for learning based temporal analysis of PE malware. In: 2021 IEEE Security and Privacy Workshops. SPW, pp. 78–84.
Yang, L., Guo, W., Hao, Q., Ciptadi, A., Ahmadzadeh, A., Xing, X., Wang, G., 2021b. {CaDE}: Detecting and explaining concept drift samples for security applications. In: 30th USENIX Security Symposium. USENIX Security 21, pp. 2327–2344.
Ye, Y., Li, T., Adjeroh, D., Iyengar, S.S., 2017a. A survey on malware detection using data mining techniques. ACM Comput. Surv. 50 (3), 1–40.
Ye, Y., Li, T., Adjeroh, D., Iyengar, S.S., 2017b. A survey on malware detection using data mining techniques. ACM Comput. Surv. 50 (3), 1–40.
Ying, X., 2019. An overview of overfitting and its solutions. J. Phys.: Conf. Ser. 1168 (2), 022022.
Yoo, S., Kim, S., Kim, S., Kang, B.B., 2021. AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification. Inform. Sci. 546, 420–435.
Yuan, C., Cai, J., Tian, D., Ma, R., Jia, X., Liu, W., 2022. Towards time evolved malware identification using two-head neural network. J. Inf. Secur. Appl. 65, 103098.
Zhu, J., Jang-Jaccard, J., Singh, A., Welch, I., Harith, A.-S., Camtepe, S., 2022. A few-shot meta-learning based siamese neural network using entropy features for ransomware classification. Comput. Secur. 117, 102691.
Zhu, S., Shi, J., Yang, L., Qin, B., Zhang, Z., Song, L., Wang, G., 2020. Measuring and modeling the label dynamics of online {anti-malware} engines. In: 29th USENIX Security Symposium. USENIX Security 20, pp. 2361–2378.
Zou, B., Cao, C., Tao, F., Wang, L., 2022. IMCLNet: A lightweight deep neural network for image-based malware classification. J. Inf. Secur. Appl. 70, 103313.

Pascal Maniriho received his B.Tech with Honors in Information and Communication Technology from Umutara Polytechnic, Rwanda, and a 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.