applied sciences

MDPI

Review

Malware Detection Issues, Challenges, and Future Directions: A Survey

Faitouri A. Aboaoja [1,*], Anazida Zainal [1], Fuad A. Ghaleb [1], Bander Ali Saleh Al-rimy [1], Taiseer Abdalla Elfadil Eisa [2] and Asma Abbas Hassan Elnour [2]

[1] Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia, Johor Bahru 81300, Johor, Malaysia
[2] Department of Information Systems-Girls Section, King Khalid University, Mahayil 62529, Saudi Arabia
* Correspondence: faitouri@graduate.utm.my

Abstract: The evolution of recent malicious software with the rising use of digital services has increased the probability of corrupting data, stealing information, or other cybercrimes by malware attacks. Therefore, malicious software must be detected before it impacts a large number of computers. Recently, many malware detection solutions have been proposed by researchers. However, many challenges limit these solutions to effectively detecting several types of malware, especially zero-day attacks due to obfuscation and evasion techniques, as well as the diversity of malicious behavior caused by the rapid rate of new malware and malware variants being produced every day. Several review papers have explored the issues and challenges of malware detection from various viewpoints. However, there is a lack of a deep review article that associates each analysis and detection approach with the data type. Such an association is imperative for the research community as it helps to determine the suitable mitigation approach. In addition, the current survey articles stopped at a generic detection approach taxonomy. Moreover, some review papers presented the feature extraction methods as static, dynamic, and hybrid based on the utilized analysis approach and neglected the feature representation methods taxonomy, which is considered essential in developing the malware detection model. This survey bridges the gap by providing a comprehensive state-of-the-art review of malware detection model research. This survey introduces a feature representation taxonomy in addition to the deeper taxonomy of malware analysis and detection approaches and links each approach with the most commonly used data types. The feature extraction method is introduced according to the techniques used instead of the analysis approach. The survey ends with a discussion of the challenges and future research directions.

Keywords: malware detection and classification models; malware analysis approaches; malware detection approaches; malware features; feature engineering

Citation: Aboaoja, F.A.; Zainal, A.; Ghaleb, F.A.; Al-rimy, B.A.S.; Eisa, T.A.E.; Elnour, A.A.H. Malware Detection Issues, Challenges, and Future Directions: A Survey. *Appl. Sci.* **2022**, *12*, 8482. https://doi.org/10.3390/app12178482

Academic Editor: David Megías

Received: 31 July 2022
Accepted: 22 August 2022
Published: 25 August 2022

1. Introduction

The constant growth of Internet users and the provision of online services such as banking and shopping services, provide hacking criminals with a suitable environment to perform their cybercrimes, which leads to a rise in the expenses that are paid to protect the systems [1]. The international damage cost that has been caused by cyber maliciousness takes the attention of the researchers due to its rapid growth. In 2021, this cost is predicted to be around 6 USD trillion according to Cybersecurity Ventures Official Annual Cybercrime Report [2]. Malware is considered the biggest threat to cybersecurity and falls under several types such as viruses, worms, trojan horses, rootkits, and ransomware since malware causes direct harm to the systems or steals their sensitive information [3] [4]. In addition, malware represents the most frequent sort of computer, network, or user attacks to cause damage or steal sensitive information [5,6]. Over recent years, the number of malicious software has increased by 22.9%, which reflects an alarming rise in threats to computer

users [7]. The authors of [8] stated that there were around one billion infected files in January 2021. Currently, malware generation has been implemented to escape the detection process via the exploitation of obfuscation and evasion techniques, and thereby produce more sophisticated dynamic malware [9]. A malware analysis process must be conducted to understand the characteristics and major functions of the malware. There are three main analysis approaches: static, dynamic, and hybrid. Static analysis means assessing a program without running it. Unlike static analysis, dynamic analysis means checking the executable's behavior by running it. The superiority and restrictions of both kinds of analysis are mutually complementary. Static analyses are faster, but malware can avoid detection if it is well disguised using obfuscation techniques or encryption strategies. Obfuscated codes and dynamic malicious hardly avoid the dynamic analysis by observing and examining the program during its runtime, but the dynamic analysis is sensitive to evasion techniques. Additionally, because it cannot analyses all the distinct operation paths, dynamic analysis does not provide all risky behaviors [10].

Several researchers collect data using both static and dynamic analysis and merge it into a single set of features to increase the detection accuracy of malware by utilizing mixed data that is generated through a hybrid analysis approach. On the other hand, the benefits and drawbacks of both static and dynamic analysis have been taken into account by the hybrid analysis approach [11]. Moreover, to protect the systems and users' data, the detection and classification models have been built utilizing three detection approaches: signature-based, behavioral-based, and heuristic-based. With a signature-based approach, a unique signature pattern has to be previously extracted to compare the given testing files' signature to an updated database of signatures and make a final decision based on the matching state [12]. However, only known malware can be detected using this approach due to the signature of the unknown malware having never been extracted yet, so the obfuscation techniques are considered the biggest weakness of this approach [13]. In contrast, the novel malware can be recognized and thus detected in the behavioral-based approach, which is conducted based on the observed behaviors during the runtime of malware in a controlled environment. Further, detecting the malware based on their behaviors is more robust against the obfuscation techniques [14]. However, malware with the ability to distinguish between the real machine environment and the analysis environment can circumvent and evade the behavioral-based approach [1]. To improve malware detection accuracy, several authors utilize manual or automated rules to develop heuristic-based malware classification and detection models. The heuristic-based models, on the other hand, are restricted to only the malicious behaviors that are represented in the general rules.

Furthermore, (ML) machine learning is commonly used to effectively support the business objectives needed and has been very successful because it is able to handle massive amounts of data such as Application Programming Interface (API) calls, Assembly code (Opcode), and Byte code, which is unacceptable for humans [15]. ML techniques offer a great deal of generality and have become an active domain in the field of cybersecurity. For modelling purposes, ML techniques such as Support Vector Machine (SVM), Naive Bayes (NB), Decision Trees (DT), etc., were used to detect harmful programs based on numerous forms of data that influence the overall detection and classification performance.

Even though several studies have introduced malware analysis approaches, there is a lack of reviews that address the relationship between each analysis approach and the used data types. Existing reviews [11,16,17] focus on the methods without relating these methods with the used data types. Furthermore, taxonomies in existing reviews [1,9,18,19] stop at generic detection approaches like signature and behavioral. These taxonomies also overlook data representation methods used by malware analysis and detection research. Additionally, the previous review papers relate the feature extraction to the analysis phase. This does not hold as the outcome from such a phase is the raw data, from which the features are extracted. That is, feature extraction comes after the data collection stage. Therefore, our review bridges these gaps by providing more granular taxonomy that

explores in detail the subcategories under each approach and associates each subcategory approach with the most used data types. This enables the research community to go deeper with more specific categories and provide solutions for the root causes behind the unsatisfactory performance of the existing malware analysis and detection solutions. Our paper introduces the data extraction process based on the used extraction techniques and considers the extraction process separately from the analysis process to highlight the broader picture between the data collection and extraction phases. Additionally, a novel taxonomy for data representation methods is introduced in this paper. In contrast, our survey did not focus on feature selection methods because feature selection methods have been reviewed extensively in the literature. Figure 1 shows the previous studies that are considered in our survey and their distribution over the period of time from 2003 to 2022.

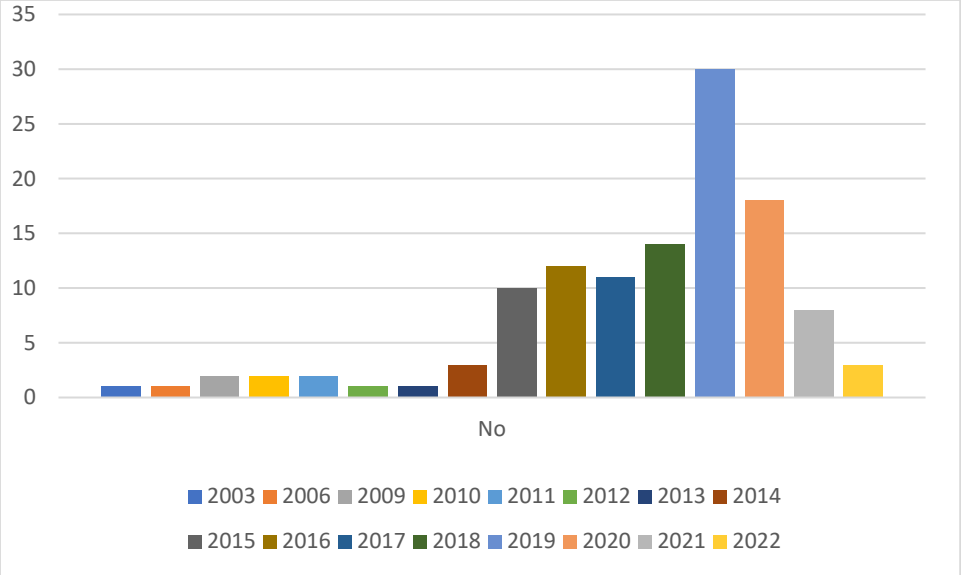

Figure 1. The distribution of the considered papers in the period of time between 2003 and 2022.

1.1. Paper Contribution

(i) The association between the analysis approaches and the user data is highlighted.
(ii) A detailed taxonomy that discusses the different types of malware signature and behaviour and distinguishes between manual and automated rules for malware detection is introduced, along with associating each subcategory detection approach with the data types used.
(iii) This survey provides a taxonomy of feature extraction and representation methods based on the techniques used to extract and represent the features.
(iv) A comprehensive understanding of the concepts for both data collection (analysis approaches) and feature extraction processes is presented in this review.
(v) The open issues and the future directions of the research community are introduced in this review.

1.2. Paper Organization

This paper is organized as follows. In Section 2, the research methodology that is followed is presented. In Section 3, this paper discussed the recent review papers. Section 4 presented the malware analysis and detection approaches taxonomy. In Section 5, we have presented the feature extraction and representation methods taxonomy. Section 6 outlined the challenges and open issues. Future directions are described in Section 7.

2. Research Methodology

We followed the methodology which is shown in Figure 2 to introduce this paper. Firstly, we focused on the review papers that have been written recently to identify the

limitations of the existing reviews and then show the need for new literature review papers. Secondly, in addition to the review papers, we used specific key words to collect the relevant experimental papers. Thirdly, according to the analysis and the detection approaches along with the extraction and the representation methods that have been utilized in each single study, the literature review is classified. Four processes, which are reading, understanding, comparing, and criticizing, have been conducted in the last phase to obtain the final results of this survey and highlight the future directions and open issues in the malware detection and classification area. Figure 2 shows the methodology that is followed to write this survey.

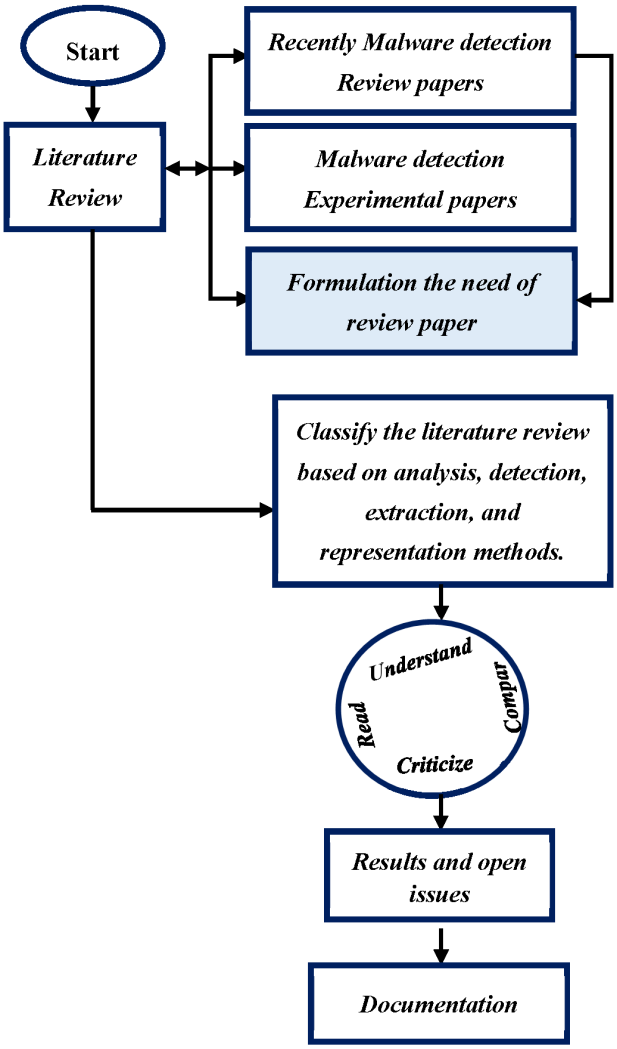

Figure 2. Research Methodology.

3. Related Work

Several review papers have been conducted in the malware detection and classification community to identify specific definitions for malware types, the development phases that are taken by malware to be more sophisticated, and the obfuscation techniques of malware. The authors of [19,20] presented the static, dynamic, and hybrid ways as analysis approaches as well as divided the detection approaches into signature-based, heuristic-based, specification-based, machine learning-based, deep learning-based, and multimodal-based. The challenges that have been faced by malware detection researchers were determined as class imbalance, open and public benchmarks, concept drift, adversarial learning, and interpretability of models. Furthermore, ref. [21] in their survey of malware detection techniques added a sandbox detection approach to those mentioned in [19,20].

In addition, static, dynamic, and hybrid were highlighted as feature extraction methods in [22], along with connecting each extraction method to its most used data types. Furthermore, the feature extraction and feature selection processes were explained from different perspectives in [16,17,23,24]. The authors of [23] reviewed machine learning-based malware detection models by outlining their phases; the feature extraction phase was presented according to the extracted data types. The authors of [16] classified the feature extraction process based on the employed techniques and methods and listed N-gram, graph-based, and dataset-based feature extraction methods. Furthermore, ref. [17] surveyed the literature and classified the features of the extraction methods based on the used analysis approaches as static, dynamic, and hybrid feature extraction methods. In their review, the analysis phase was explored from two aspects, which were steps and strategies. The analysis steps were identified as assembly, representation, and classification.

Furthermore, the family's analysis, similarities analysis, and variants analysis were determined as analysis strategies. Different selection and detection techniques were explored to explain how they affect the malware detection model's performance. Subsequently, the impact of the feature types and the classification techniques have been investigated to prove that there is no one approach capable of detecting all the malware types. The author of [24] presented a complete overview of modern methods and strategies for the feature selection phase based on the data perspective. Therefore, they divided the feature selection methods into theoretical, spare learning, and statistical approaches based on similarity.

Some of the researchers focused on the evaluation of the evolution and the complications that have occurred in modern sophisticated malware. The authors of [1,9] addressed the second generation of malware in detail, along with their development steps to evaluate the evolution of malicious with the associated detection approaches, which were specified as signature-based, behavior-based, heuristic-based, specification-based, energy-based, bio-inspired-based, and machine/deep learning-based to show reliance between the malware development and the detection techniques. As more complicated malware, evasive malware is discussed by [25] through investigating the FFRI dataset to identify the ratio of evasive malware and explain the trend of employed evasion techniques as well as the effect of anti-analysis operations on the analysis and detection processes, but their results include the anti-analysis operations used by malware through considering only the API calls, so the other anti-analysis operations which need to be understood using other features were not covered by their study.

Some researchers survived state-of-the-art malware detection according to a particular attack or using a specific technique. The authors of [26] investigated several characteristics of specific malware attacks on smart home networks. A taxonomy of those attacks was presented based on smart home architecture, smart home central hub, and smart home physical security. Furthermore, VPN filter malware has been addressed in terms of its vulnerability, impact on router vendors, and effect on the network of smart homes. Likewise, Ref. [27] studied the advanced persistent threats (APTs) in detail to explore their characteristics, models, payload delivery methods, and advanced evasion techniques. Furthermore, the analysis approaches and the existing application hardening techniques that have been used to mitigate the malware were taxonomized. The solutions that have been adopted to design a secure region against APTs were stated.

On the other hand, Ref. [28] introduced a survey of data mining techniques-based malware detection approaches. Signature-based and behavior-based were introduced as malware detection approaches along with their frameworks to explain how both use machine learning algorithms. Furthermore, the rate of using the machine learning algorithms in the literature review was illustrated and the challenges were summarized. A summary of data mining-based malware detection approaches was provided by [29] in their work. In addition to the signature-based approach, they added heuristic-based and specification-based malware detection approaches as well as the advantages and disadvantages of each mentioned approach.

However, this study introduces a comprehensive review including a sufficient number of studies to provide a taxonomy for malware analysis and detection approaches, along with highlighting the most frequently used data types for each approach. Additionally, unlike the existing taxonomies that stop at generic detection approaches like signature and behavioral, our review provides a deeper taxonomy to introduce the known detection approaches in detail by presenting novel subcategories under each approach as well as associating each subcategory with the most used data types. This enables the research community to go deeper with a better understanding of the existing detection approaches. In contrast to the existing reviews, which focused on the used data types and the analysis approach (static, dynamic, and hybrid) when they identified the extraction methods, this survey presents the feature extraction phase from the perspective of which technique is used to achieve the extraction process, to introduce a clearer concept that highlights the differences between the data collection process that is conducted during the analysis phase and the feature extraction process that is performed after the analysis phase. Thus, our survey forms the border between those two phases. Furthermore, modern feature extraction methods are added to the ones that existed in the literature, as well as those methods are discussed and compared in this review.

Moreover, a novel taxonomy of feature representation methods is presented in this paper to fill up the gap left by the existing reviews that have never included this taxonomy. Further, tackling those representation methods and identifying distinct definitions for each one as well as explaining the weaknesses leads to a more precise understanding among the research community's authors. Unlike earlier reviews, which treat ML/Deep techniques as a distinct detection approach, this study defines ML/Deep techniques as algorithms that have been widely utilized to enhance the performance of whatever detection approach is used, such as signature-based, behavioral-based, or heuristic-based detection. Table 1 shows the comparison between this survey and the previous review papers based on the list of the contributions.

Table 1. A list of the contributions to this survey with a comparison to the previous review papers.

Review Paper	Date	Link Data Types to Analysis Approaches	Link Data Types to Detection Approaches	Detection Approaches in Deep Categories	Techniques-Based Extraction Methods	Representation Methods	Show the Border between Collection Data and Feature Extraction Phases
[19]	2018	✗	✗	✗	✗	✗	✗
[20]	2020	✓	✗	✗	✓	✓	✓
[21]	2021	✗	✗	✗	✗	✗	✗
[22]	2014	✓	✗	✗	✓	✗	✗
[23]	2019	✗	✗	✗	✓	✗	✗
[16]	2020	✓	✓	✗	✓	✗	✗
[17]	2021	✓	✗	✗	✗	✗	✗
[24]	2017	✗	✗	✗	✗	✗	✗
[1]	2021	✗	✓	✗	✗	✗	✗
[9]	2020	✗	✗	✗	✗	✗	✗
[25]	2018	✗	✗	✗	✗	✗	✗
[26]	2019	✗	✗	✗	✗	✗	✗
[27]	2019	✗	✗	✗	✗	✗	✗
[28]	2018	✗	✓	✗	✗	✗	✗
[29]	2014	✗	✗	✗	✗	✗	✗
This survey		✓	✓	✓	✓	✓	✓

4. The Taxonomy of Malware Analysis and Detection Approaches

This section describes the taxonomy of malware analysis and detection approaches. While malware analysis is taxonomy and linked to the data types that are used with each analysis approach, malware detection is introduced with a deep taxonomy where each known detection approach is presented in subcategories and the relationship between each introduced detection subcategory and the data types that are utilized is determined. Figure 3 shows the malware analysis and detection taxonomy, where the analysis approaches are presented as static, dynamic, and hybrid, as well as showing the frequently

used data types with each analysis approach. Regarding malware detection approaches, sub-detection approaches which go deeper than the well-known approaches, signature-based, behavioral-based, and heuristic-based have been presented. Static and dynamic signatures, continuous, sequential, and common behavioral, and automated and manual rules are displayed as deep categories of the major detection approaches along with associating each sub-detection approach with the most used data types.

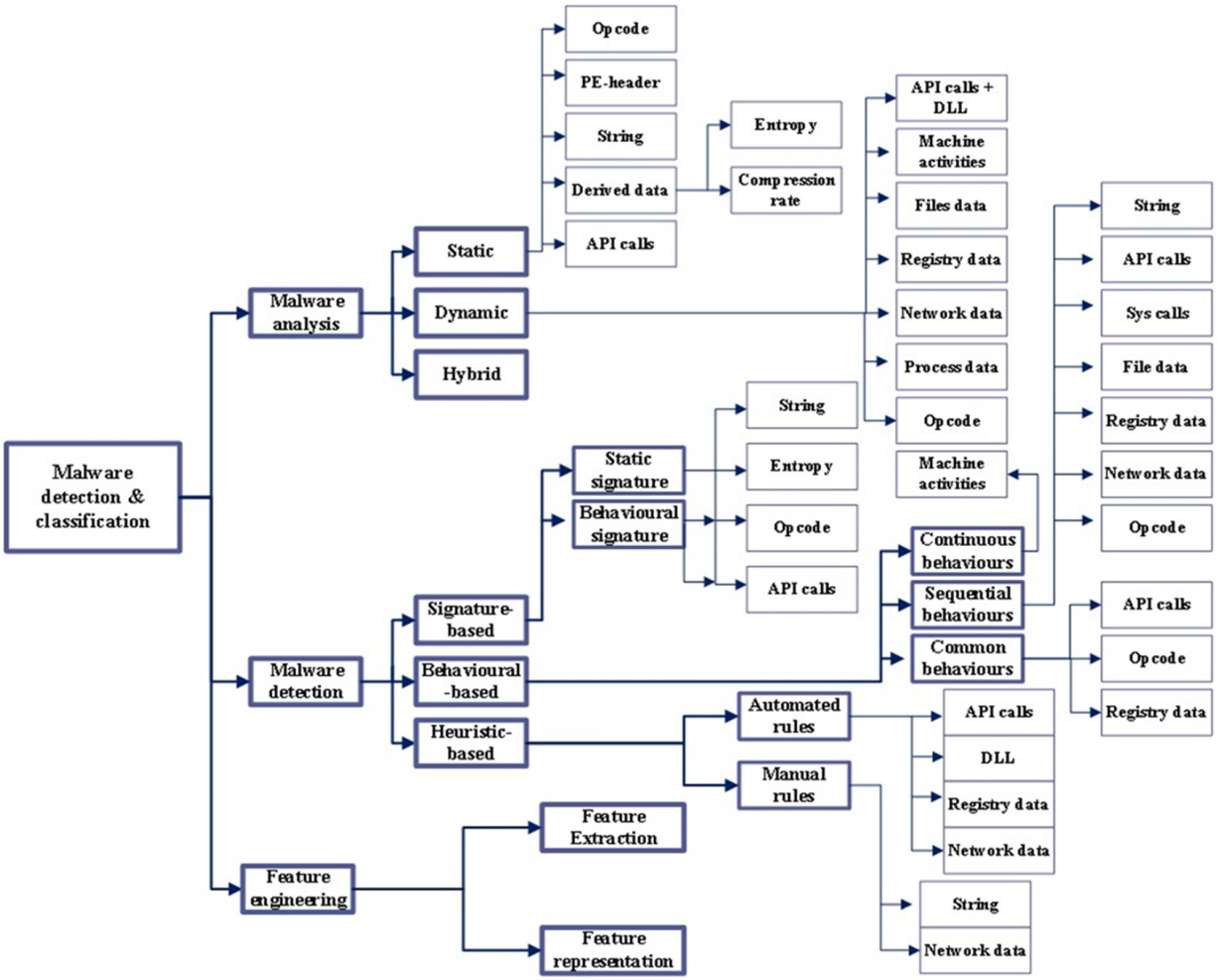

Figure 3. Malware Analysis and Detection Taxonomy.

4.1. Malware Analysis Approaches and Data Types

Malware analysis and the type of data have a considerable and growing impact on the detection process that determines the classification of the investigated file and therefore influences the overall accuracy of the detection models. Several types of data have been extracted using static, dynamic, and hybrid analysis, such as Byte code, Opcode, API calls, file data, registry data, and so on, to understand and acknowledge the major purpose and function of the files examined and therefore classify them as malware or benign files. Therefore, we discuss below the analysis approaches that have been employed in recent studies alongside the types of data that have been extracted in those studies to identify the impact and the trend of the data types. Table 2 contains the characteristics of each reviewed paper in terms of the analysis approach and the extracted data.

4.1.1. Static Analysis

The static analysis approach has been widely utilized by exploring the source code without running the executable files to extract a unique signature which is used to rep-

resent the file under investigation. Several types of static data can be collected via static analysis, including PE-header data [30–33], derived data such as string-based entropy and compression ratio [34–37]. Additionally, static analysis tools, such as IDA pro disassembler and Python-developed modules, are also used to collect static opcode and API calls [38–42]. Even though static analysis can track all the possible execution paths, it is influenced by packing and encryption techniques.

4.1.2. Dynamic Analysis

Several researchers performed a dynamic analysis approach to collect various data types to differentiate between malware and benign files by running the executable files in isolated environments, virtual machines (VM) or emulators to monitor the executable file behavior during the run-time and then collect the desired dynamic data [43]. Various kinds of data have been collected utilizing a dynamic analysis approach. Malicious activities can be dynamically represented using both executable file behavior and by retaining memory images during run-time. The executable files' behaviors are identified through collecting the invoked API calls [44–48], machine activities [47,49,50], file-related data [51–53], and registry and network data [45,54]. Opcode-based memory image can be taken to represent the malicious activities dynamically [15]. Even though obfuscated malware cannot hide how it behaves when dynamically analyzed, dynamic analysis is unable to satisfy all malicious conditions in order to explore all execution paths.

4.1.3. Hybrid Analysis

Some previous studies combined data extracted through static and dynamic analysis together to reduce the drawbacks of both analysis approaches and achieve a higher detection rate. Different tools, including Cuckoo sandbox, IDA pro disassembler, and OlleyDbg, are employed to collect dynamic and static data, and then hybrid feature sets are created based on several types of data, such as string, opcode, API calls, and others [55–59]. Even though the hybrid analysis approach benefits from the advantages of both static and dynamic analysis, it also suffers from their disadvantages.

4.1.4. Malware Analysis and Data Types Discussion

The most used data types with each analysis approach have been shown in Figure 4. On the *x*-axis, the data types are depicted as string (St), PE-header (P-h), Opcode (Op), API calls (API), Dynamic link library (DLL), machine activities (MA), process data (PD), file data (FD), registry data (RD), network data (ND), and derived data (DD). Similarly, static, dynamic, and hybrid are mapped on the y-axis as analysis approaches. The horizontal and vertical dotted lines illustrate the relationship between each data type and each analysis approach by showing how frequently each single data type has been used with each particular analysis approach in the literature review.

By focusing on static analysis, opcode and PE-header represent the first and second most repetitively utilized static data types, respectively. This survey found that using the static data type that is employed in the minimum number of studies, the API calls that are collected statically were not preferable in the literature compared to the usage of byte-code data and the derived data. In contrast to static analysis, the API calls data demonstrates the most significant data type that has been extended using dynamic analysis. While machine activities data is the second trend data type, using data related to registry value, file, and network delivers the average ratio among studies that extracted their required data dynamically. Furthermore, bytecode, PE-header, and Opcode data are rarely extracted using dynamic analysis. In the studies that have utilized both static and dynamic analysis as hybrid analysis, it is clear to note that the same ranges are repeated for the data that are associated with static and dynamic analysis, such as Opcode, bytecode, and PE-header as static data and the API calls as dynamic data.

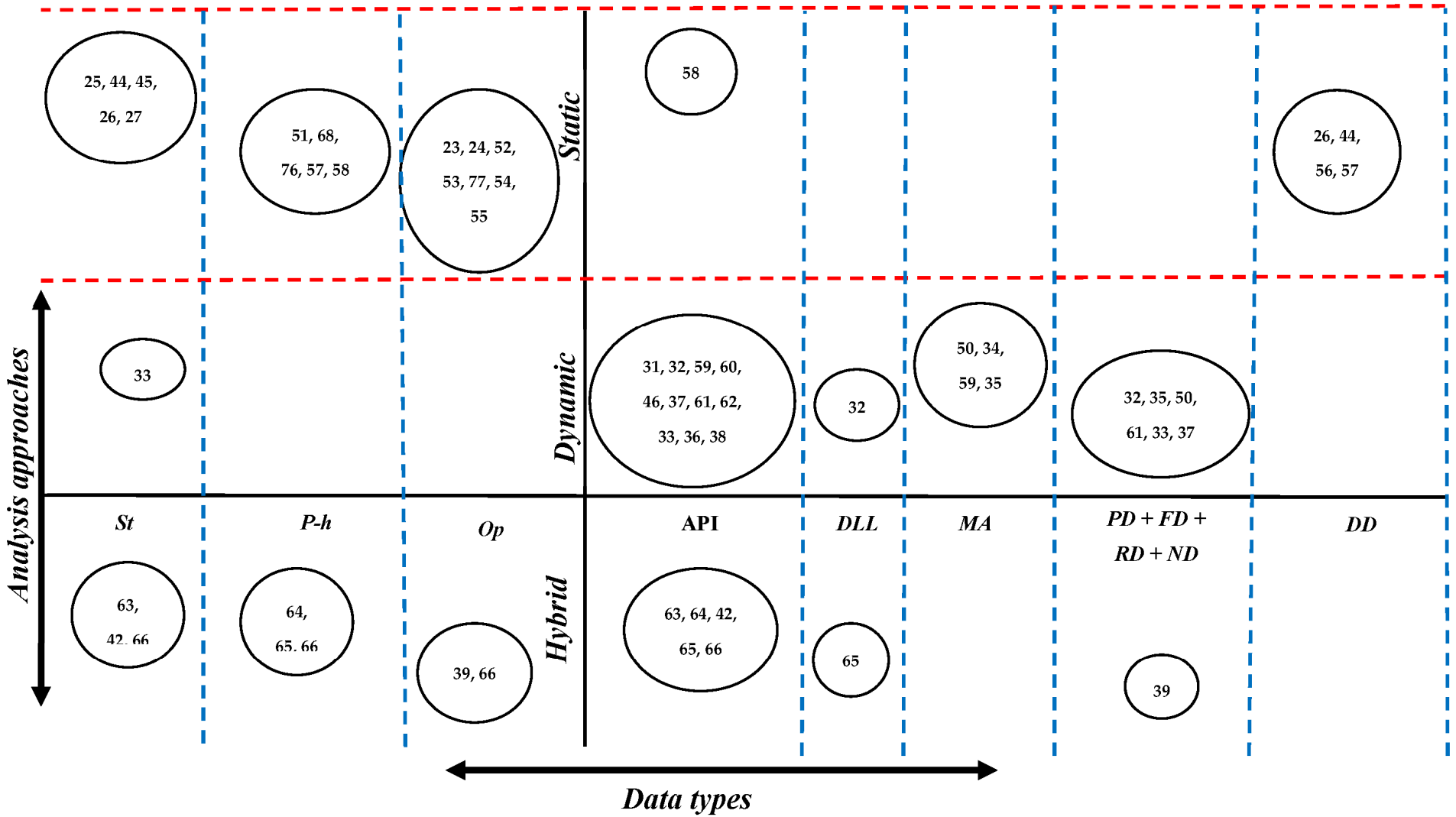

Figure 4. Analysis approaches vs. data types.

On the other hand, the usage ratio of some of the dynamic data, such as data related to files, registry, and machine activities, is decreased in the studies that choose hybrid analysis to extract their features.

Even though the static analysis is safe due to there being no need to run the files, malicious software frequently employs encryptors like UPX and ASP Pack Shell to prevent analysis. As a result, unpacking and decompression processes are required before analysis, which is accomplished with disassemblers such as IDA Pro and OlleyDby. Contrary to the static analysis approach, the dynamic analysis approach is more effective because there is no need to unpack the investigated file before the analysis process. Dynamic analysis can support the detection models to be able to detect novel malware behaviors in addition to known malware. Further, the general behavior of the malware is not affected by obfuscation techniques, so it is difficult for obfuscated malware to evade the dynamic analysis approach because the obfuscation techniques change the malware structure but could not change the malware behavior. However, dynamic malware analysis is more sensitive to evasive malware and can detect whether it is being executed in a real or controlled environment.

4.2. Malware Detection Approaches

The malware detection process is the mechanization that must be implemented to discover and identify the malicious activities of the files under investigation. As a result, several approaches to detecting malware have been improved year after year, with no single approach providing 100% success with all malware types and families in every situation. Therefore, malicious software has been detected based on two main characteristics, which are signatures and behaviors using three malware detection approaches that are signature-based, behavioral-based, and heuristic-based. Therefore, the following sections discuss the malware detection approaches. Table 3 contains the characteristics of each reviewed paper in terms of detection approach with the extraction and representation methods.

4.2.1. Signature-Based

Several studies have been undertaken to improve malware detection and classification models by relying on a unique signature that has been previously extracted statically or dynamically and stored to compare it with the collected investigated file signature. Those signatures include but are not limited to a set of API calls, opcodes or byte-code series, and entropy quantity. Static string-based signatures have been generated by [35,60] to detect VBasic malicious software by representing the obtained strings using frequency vectors, while [61] generated static signatures based on n-grams and binary vectors. Additionally, [38] formed malicious static signatures using the statistical values of opcodes. On the other hand, behavioral signatures have been constructed based on the dynamically collected data. The authors of [48,62,63] created behavioral signatures using API calls invoked by malware during their running time. Specific sets of API calls are identified as reflecting malicious activities, and thus the behavioral malicious signatures are constructed utilizing those API calls. Static and behavioral signature-based malware detection models suffer from low detection rates when classifying unknown signatures that may be linked to unknown malware or different variants of known malware.

4.2.2. Behavioral-Based

After monitoring the executable files in an isolated environment and collecting the exhibited behaviors, features extraction techniques have been developed to extract the sensitive features by which the developed model can classify the known malicious behaviors as well as any behavior that seems to be similar to them with respect to false positive behaviors. The ability to identify novel malware behaviors in addition to the known ones based on collecting behaviors during run-time has made this approach more valuable than the signature-based approach. As a result, the majority of the studies in the literature review focused on using behavioral-based approaches in the form of continuous, sequential, and common behaviors to increase malware detection ratios.

Table 2. The relation between analysis approaches and data types.

Ref.	Date	String	PE-Header	Opcode	API Calls	DLL	Machine Activities	Process Data	File Data	Registry Data	Network Data	Derived Data		Accuracy
												Entropy	Compression	
Static Analysis														
[38]	2015			✓										98.60%
[31]	2015		✓											99.97%
[64]	2016	✓												99.00%
[35]	2018	✓										✓		NA
[39]	2017			✓										96.09%
[65]	2017			✓										98.90%
[61]	2017	✓												81.07%
[41]	2018			✓										NA
[32]	2019		✓											99.80%
[33]	2019		✓											87.50%
[42]	2019			✓										97.87%
[66]	2020			✓										NA
[67]	2019			✓										91.43%
[34]	2016	✓											✓	89.44%
[36]	2019											✓		NA
[37]	2019		✓									✓		NA
[40]	2020			✓	✓							✓		NA
[68]	2016		✓		✓									98.17%
Dynamic analysis														
[43]	2015							✓						97.8%
[44]	2016				✓									97.19%
[15]	2016			✓										98.92%

Table 2. *Cont.*

Ref.	Date	String	PE-Header	Opcode	API Calls	DLL	Machine Activities	Process Data	File Data	Registry Data	Network Data	Derived Data		Accuracy
												Entropy	Compression	
[45]	2016				✓	✓				✓				96.00%
[52]	2016						✓		✓					NA
[69]	2017						✓							98.54%
[47]	2018				✓		✓							NA
[49]	2019						✓							92.00%
[54]	2019				✓						✓			97.22%
[48]	2019				✓									94.89%
[46]	2020				✓									97.28%
[53]	2019				✓				✓					75.01%
[70]	2020				✓									98.43%
[51]	2020	✓			✓				✓	✓	✓			99.54%
[71]	2017				✓									NA
Hybrid analysis														
[72]	2014	✓			✓									98.71%
[55]	2016			✓					✓					99.99%
[73]	2019		✓		✓									99.70%
[58]	2021	✓			✓									94.70%
[74]	2020		✓		✓	✓								93.92%
[75]	2020	✓	✓	✓	✓									96.30%

4.2.3. Behavioral-Based

After monitoring the executable files in an isolated environment and collecting the exhibited behaviors, features extraction techniques have been developed to extract the sensitive features by which the developed model can classify the known malicious behaviors as well as any behavior that seems to be similar to them with respect to false positive behaviors. The ability to identify novel malware behaviors in addition to the known ones based on collecting behaviors during run-time has made this approach more valuable than the signature-based approach. As a result, the majority of the studies in the literature review focused on using behavioral-based approaches in the form of continuous, sequential, and common behaviors to increase malware detection ratios.

Some studies have been conducted based on the extracted continuous behaviors which, are represented by machine activities. The authors of [41] was interested in the Windows platform and used the Cuckoo sandbox to extract machine activity data (CPU, memory, received, and sent packets). After that, the observations were transformed into vectors, which were used to train and assess classification algorithms. Most of the previous studies were concerned with extracting API calls, system calls, opcodes, and others to form them sequentially (sequential behaviors) or into ordered patterns to understand the malicious functionalities. The sequential or ordered patterns can be API calls, registry data, and network data [13,40,45] or opcode sequences [76]. Moreover, the common behaviors that are performed by malware and benign samples can be used as an indicator to classify the investigated file between malware or benign classes in the binary classification models. In addition, those common behaviors can be observed in each malware family in the case of multi-classification models. The time of the matching process has been reduced from where the developed models classify the test files based on only the common behaviors. Common behaviors graph-based malware detection and classification models have been proposed by [10,77] in their work through observing the most frequent behavior graphs in each malware family. Additionally, ref. [78] presented binary and multi-classification models using (LSTM) long-short term models based on the common API call sequences offered by each malware family.

4.2.4. Heuristic-Based

A heuristic-based approach has been used in various research by generating generic rules that investigate the extracted data, which are given through dynamic or static analysis to support the proposed model of detecting malicious intent. The generated rules can be developed automatically using machine learning techniques, the YARA tool, and other tools or manually based on the experience and knowledge of expert analysts.

Several studies have been done to develop malware detection models by which decisions have been taken based on the automated behavioral rules that are created using machine learning techniques and the YARA tool [45,47,54]. On the other hand, based on statically extracted string data, ref. [34] was concerned with generating manually general rules to recognize the existence of malicious activities that might be achieved by malware using HTML elements and JavaScript functions. Moreover, (DNS) domain name system-based rules were developed by [76] to build a botnet attack detection model. The proposed model took the final decision based on the manually developed general rules which can detect abnormalities in DNS queries and responses.

4.2.5. Malware Detection Discussion

According to the literature, the researchers applied string, opcode, and derived features to construct static signatures, while only API calls and opcodes were used to create dynamic signatures. In general, the ability to detect malware utilizing previously derived signatures using both static and dynamic signatures is insufficient to combat malicious software enhancement due to obfuscation techniques popularly used by malware writers to create different malware variants as well as new malware, because each malware variant and new

malware seems to have a different fingerprint, which must be obtained before the detection process can begin.

Furthermore, malware detection models based on static signature patterns have been defeated during the tackling of encrypted or compromised malware. Regarding the behavioral-based approach and the preferable data types, machine activities data is the most used feature in the literature to depict malware functionality using continuous behaviors, whereas API calls data is the most frequently utilized feature to build malware detection models using sequential behaviors. Furthermore, the most used data types when authors try to get the common behaviors among malware groups are API calls and opcodes. The behavioral-based approach is a promising solution to overcome the weaknesses of the signature-based approach but relying on behaviors leads the suggested models to misclassify malware that performs functions that are similar to benign functions or mimic legitimate behaviors, and then those models suffer from a high false-positive rate.

Moreover, malware is capable of recognizing the nature of its execution surroundings using evasion techniques and then changing its behaviors to be like benign behaviors or terminating its execution, resulting in representing them through unrepresentative behaviors. In addition, extracting a sufficient feature set is a tough process that has a massive effect on the malware detection and classification models. Furthermore, representing malware behaviors based on the names, sequence, or frequency of extracted characteristics results in malware detection and classification models that are more vulnerable to obfuscation techniques, which are employed to update the names, sequences, and frequencies of the extracted characteristics. Furthermore, several researchers trained their models by employing malicious behaviors that were extracted from the most recent malware to provide the classifiers the capacity to recognize trends in malicious activity. On the other hand, developed malware detection models have become vulnerable to older malicious behaviors.

By focusing on the heuristic-based approach, there is no single type of data that is commonly used with this approach in the literature, but the researchers have used practically almost all the types of data at the same rate, including API calls, network data, registry data, import DLL, and others. However, the creation of general rules that play a significant role in the final decision is required when building a malware detection and classification model based on the heuristic approach. Therefore, generating the investigation rules manually consumes time and effort and needs malware behavior experts with enough experience. Even though the required rules can be generated automatically, the suggested rule-based model is limited to detecting only the malicious activities that are represented in the critical general rules.

5. The Taxonomy of Feature Extraction and Representation Methods

This section covered a taxonomy of feature extraction and representation methods. In contrast to previous feature extraction taxonomies that were introduced based on the analysis approach, this paper presented a feature extraction taxonomy according to the techniques employed to extract the features. Furthermore, a novel feature representation taxonomy is identified to clear up the borders between data collection, extraction, and representation phases. The feature extraction and representation taxonomy are shown in Figure 5.

5.1. Feature Extraction Methods

Generally, feature engineering refers to the extraction, selection, and representation of features. This is an important phase in the malware detection and classification process because it has a significant impact on the classification model's performance [70]. After all, the feature engineering procedure arranges the features towards a more machine-understandable subset of data [22]. Additionally, the reduced computational overhead is achieved by decreasing the dataset for processing [79]. The feature extraction methods based on extraction techniques are presented in the following subsections. Table 3 shows the feature extraction method in each reviewed paper.

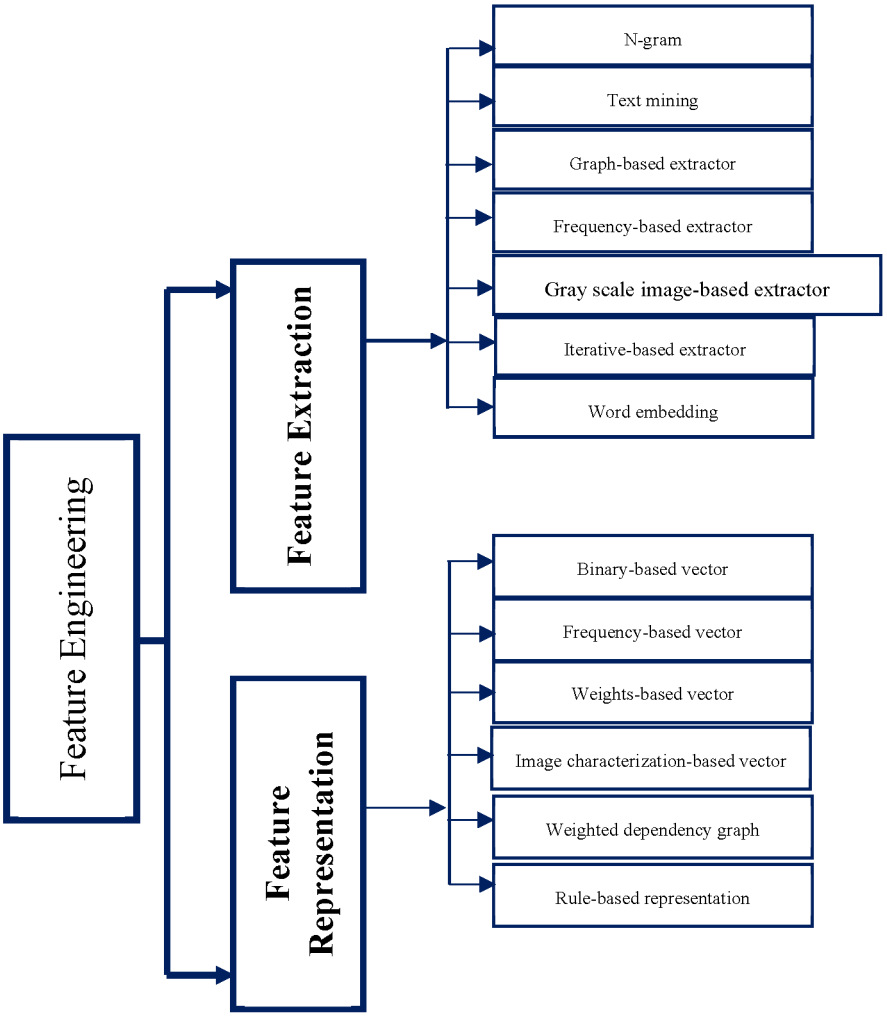

Figure 5. Taxonomy of feature extraction and representation methods.

5.1.1. N-Gram

Using a single feature, such as an API call or an opcode, separately causes a low detection rate in some text classification models because doing so ignores the importance that can be obtained when combining multiple features together as one feature during the extraction phase. Malicious behaviours can be done using a group of API calls or opcodes. To the best of our knowledge, malware often does not rely on a single feature to perform its malicious activities. Therefore, the existing solutions for malware detection and classification models have widely used feature extraction methods based on n-gram techniques in order to combine more than one feature over the extraction phase. By using the n-gram technique, each single extracted feature is constructed from N features that are offered by the malware during the analysis phase.

Several studies have used the n-gram technique to extract features [50,65,72,80,81]. Constructing subsets from an original set of text with a length of n is called n-gram. Depending on the application, this string may contain a variety of kinds. It can include letters or words. N-grams are made by dividing a text string into fixed-length substrings. The accuracy of the measure of similarity across terms has been enhanced because of the use of the N-gram [82]. However, high dimensionality feature space is caused by using the n-gram technique, due to the large number of generated features using the n-gram technique which leads to more time being consumed [10].

Table 3. Summary of detection approaches, extraction, and representation methods.

Reference	Feature Engineering		Model Type		Technique	Model Mode		Detection Approach		
	Feature Extraction	Feature Representation	Binary-Class	Multi-Class		General	Specific	Signature-Based	Behavioral-Based	Heuristic-Based
[43]	Text mining	Weight-based vector	✓			✓		✓		
[72]	Frequency-based extractor, N-gram	Binary-based vector	✓		SVM		✓		✓	
[45]	Frequency-based extractor	Binary-based vector	✓		SVM	✓				✓
[34]	Frequency-based extractor	Weight-based vector	✓		AMA		✓			✓
[39]	Graph-based extractor	Weight-based vector			K-NN, SVM					
[35]	Frequency-based extractor	Frequency-based vector	✓		RF		✓	✓		
[83]	N-gram	Binary-based vector		✓	K-NN	✓			✓	
[80]	Graph-based extractor	Weighted dependency graph	✓			✓			✓	
[84]	Iterative-based extractor	Weight-based vector		✓	J48, K-NN		✓		✓	
[54]	Text mining	Rule-based representation	✓			✓				✓
[63]	Frequency-based extractor	Binary-based vector	✓		SVM	✓		✓		
[75]	N-gram, Frequency-based extractor, Gray-scale image-based extractor	Image characterization-based vector	✓	✓	XGBoost	✓			✓	
[74]	N-gram	Weight-based vector	✓		RF	✓			✓	
[85]	Word embedding	Weight-based vector	✓		MC	✓			✓	
[50]	N-gram	Frequency-based vector	✓		SVM	✓			✓	
[10]	Graph-based extractor	Binary-based vector			HLES-MMI		✓		✓	
[81]	N-gram	Weight-based vector	✓			✓		✓		
[70]	N-gram	Binary-based vector	✓		LR	✓			✓	
[86]	Gray-scale image-based extractor	Image characterization-based vector	✓		SVM, CNN	✓			✓	
[40]	N-gram, Frequency-based extractor	Frequency-based vector, Weight vector	✓		XGBoost	✓		✓		
[87]	N-gram, Frequency-based extractor	Frequency-based vector	✓		CNN, BPNN	✓			✓	
[88]	Word embedding	Weight-based vector	✓		RNN	✓			✓	
[89]	Frequency-based extractor	Weight-based vector	✓		CS	✓			✓	
[90]	Frequency-based extractor	Binary-based vector	✓		SVM	✓			✓	
[91]	Graph-based extractor	Weighted dependency graph	✓	✓		✓			✓	
[76]	Frequency-based extractor	Rule-based representation	✓			✓	✓			✓
[92]	Frequency-based extractor	Weight-based vector	✓		JSD	✓			✓	
[93]	Gray-scale image-based extractor	Image characterization-based vector		✓	CNN	✓		✓		
[94]	Gray-scale image-based extractor	Image characterization-based vector		✓	SVM	✓		✓		

5.1.2. Text Mining

Text mining techniques are taken from the information retrieval field. Term indexing and term weighting represent the two main categories of text mining techniques. Term indexing, where a unique index is assigned to each term, while term weighting means that a specific weight is calculated and assigned to each term. In the malware detection and classification fields, term weighting methods have been widely used to extract significant features. Identifying specific importance for each word or term arranged in the analysis reports is critical to improving the proposed model's performance. There are various techniques, such as (TF-IDF) Term Frequency-Inverse Document Frequency, (IG) Information Gain, and others, to mine the texts and define their importance by assigning weights by which the analysts can know the most useful words for classification purposes.

The authors of [43] assigned weights for each text in the analysis report using the information gain technique. Those weights were assigned to the texts that represent the performed operations and their locations based on the occurrences of those texts in malware and benign classes. In addition, TF-IDF (Term Frequency-Inverse Document Frequency) has been used by [54] as a weighting method to extract the API calls and network traffic by evaluating the appearance of each feature in both malware and benign classes to identify the uncommon features that are more significant than the common features.

5.1.3. Graph-Based Extractor

To extract only the most important features, raw data such as API calls and opcodes are formed into graphs that represent the appearance of each feature in that sample as well as its relationships with others. Therefore, the frequent sub-graphs among all the samples of a specific class or family are considered significant features that have to be extracted. Those sub-graphs consist of nodes (API calls, opcodes) and their relationships with other features (dependency or control flow). Some studies extract the features using algorithms that construct graphs. Those graphs consist of groups of nodes as basic blocks in the program. These basic blocks are connected by edges to represent the path of control flow between the nodes as well as the blocks [95]. Part of the previous studies [10,80,96], have used graph-based extraction methods to extract the optimal feature set. CFGs were created based on opcodes or API calls. Some studies used matching algorithms immediately to distinguish between malicious and benign constructed graphs, while others represented the constructed graphs as binary vectors, weight vectors, or weighted dependency graphs. However, it is difficult to build a unique graph for each individual malware and, considering the common graph-based behaviors, might lead to an increase in the detection time.

5.1.4. Frequency-Based Extractor

Each feature that appears in the raw collected data can be a discriminative or redundant feature. The importance of the features depends on how they occur in each class. As long as the feature appears frequently in one of the classes and does not occur in the other classes, the possibility for that feature to be a significant feature increases. According to the aforementioned concept, some studies rely on the frequency when they develop feature extraction techniques. The occurrence of the features is measured and analyzed to be used as an indicator for the prominent features to prevent creating an extremely high-dimensional feature space that leads to an ineffective model. The frequency of API calls and opcode were used by [75,87] to extract features and construct the dataset that was used to train and assess their suggested models, while [45] considered the frequency of each unique API, DLL, and registry key in each document as a feature. However, obfuscation techniques such as instruction replacement and dead code insertion, which update the frequency of the derived features, have had an impact on frequency-based models [10,62].

5.1.5. Word Embedding

The word embedding method can predict the distribution of each word or feature. The well-known embedding word Word2Vec can be developed in two different ways: Bag-

of-Words (CBOW) and Skip-Gram. (CBOW) used the (context) words before and after the target word as input to predict the output, which is the target word, while Skip-Gram used the target word as input to predict the output, which is the (context) words before and after the target word. Word2Vec transforms the corpus of text into vector space in which each text feature in the corpus is represented by a vector in the space. Textual similarities are considered in the vector space. In the case of two words that are similar in context, they will be placed close to each other in the vector space. Therefore, based on the fact that the order of API calls or opcodes in malware sequences must contain textual meaning and it does not randomly create it, parts of the previous studies have utilized Word2Vec techniques to extract the feature set from which, the textual relationships are represented [85].

The authors of [85,97,98] used API sequence, while [88] used opcode as inputs to produce their extracted features as vectors representing each word using the assigned weights that show the similar words in the context close to each other. Even though the Word2Vec technique captures the contextual relationship between the features (words), several characteristics of the words might lead to an increase in the computational complexity that reduces the overall performance of the proposed models.

5.1.6. Iterative-Based Extractor

Since the final objective of each developed malware detection and classification model is to achieve satisfactory accuracy when classifying the test data, this method focuses on the features by which the detection accuracy is extremely improved. The challenge when an iterative-based extractor method is used is how to identify the initial feature set. The obvious solution to this challenge is that the whole features that appear in almost all the samples of one class have to be included. Therefore, performing extensive experiments and analyzing the results, along with including and excluding parts of the initial feature set, in order to identify the optimal feature set by which it achieves the highest performance. This method entails conducting a large number of experiments, followed by analyzing the obtained results to highlight the features that are associated with the highest detection accuracy. The authors of [84] extracted the typical feature set from the analysis reports based on the series of experiments that were conducted to identify the effective features relying on the improvements that have been achieved by each additional feature in every experiment. Even though this method ensures the efficiency of the extracted feature set because those extracted features are associated with the highest achieved detection accuracy, it is time- and effort-consuming.

5.1.7. Gray Scale Image-Based Extractor

The malware binaries are divided into 8-bit units, and then each 8-bit unit can be utilized as a unique pixel in the image. This is because any 8-bit unit of malware binary provides a numeric value between 0 and 255, where 0 represents the black color and 255 represents the white color. The values between 0 and 255 are gradually moved between the black and white colors. Moreover, each 8-bit unit or a single pixel in the image of malware binary can be painted based on the degree of the color that is expressed in that pixel. Therefore, gray images can be generated for each malware binary, and then there are several features that are related to the generated images that can be extracted, such as texture, intensity, and wavelet. Malware variants that are remembered in the same family produce similar visualizations as a result of reused codes [99]. Therefore, some research is concerned about distinguishing between malware and benign files based on the extracted visualization features.

The authors of [86] read the file as an 8-bit unsigned integer between 0 (black) and 255 (white), and then use a machine learning technique to transfer those files and save them as images. However, relying on visualization-based features requires an adequate number of malware families and types during the training phase to build an effective model. The authors of [100] in their work, utilized the transfer learning TL-based CNN models to obtain the main characteristics of malware based on their converted images. The gray-scale

images were created by converting the malware files into 1D, 8-bit vectors and then 2D images. Moreover, an image resizing process was achieved to introduce appropriate inputs for TL-based CNN models. The created dataset was divided into training data (80%) and test data (20%). The VGG16 model performed the best among all the utilized TL-based CNN models.

5.1.8. Discussion of Feature Extraction Methods

Even though using the n-gram method to extract features appears to be quite popular among authors, this technique produces a large number of features, making N-gram-based models suffer from high dimensionality. The frequency-based feature extraction method is employed to decrease the number of extracted features by extracting only the most often occurring ones, which helps to overcome the problem of high dimensionality space. However, obfuscation techniques that can adjust the frequencies of certain features in each variant invalidate the frequency-based method.

Though it is impossible to develop a unique graph for each type of malware, some studies have employed a graph-based feature extraction method to construct generic graphs using common characteristics. The matching procedure, on the other hand, caused the graph-based models to have a significant level of complexity over time. The problem of time matching has been examined, and a solution has been proposed in the representation phase when some studies represented constructed graphs as vectors.

On the other hand, the text features have been given weights to indicate which features should be extracted. Text mining-based models are particularly vulnerable to obfuscation techniques because those weights are derived using frequency-based techniques like TF-IDF. Another direction was taken by using a Gray-based method as a feature extractor when the produced data can be visualized. The limitation with the visualization extractor is that the extracted features would be stored as images, which needs more storage space. Regardless of how the word embedding method captures the contextual relationship between the characteristics (words), various aspects of the words may raise the computational cost and lowering the overall performance of the model.

5.2. Feature Representation Methods

Next to the feature extraction step, a significant step must be performed, which is how to represent the characteristics of malicious and legitimate activities. In other words, how to transfer the extracted characteristics to be understandable for algorithms and machines using several vector types from which the proposed models can learn the behaviors of different classes. Therefore, the proposed models have become capable of distinguishing between malware and benign files based on the representation methods that have been used by those models to recognize the characteristics of both classes [79]. Features representation methods are identified and discussed in the next subsections. Table 3 shows the feature representation in each reviewed paper.

5.2.1. Binary-Based Vector

This method of representation examines the extracted features from the perspective, which has only two aspects: true and false. True if the examined feature exists in the document, and false if the examined feature does not exist in the document. The binary vector representation is a widely used method where the extracted features are represented by 1 or 0 according to their existence in each sample to create a binary vector that represents the characteristics of that sample. The authors of [72] integrated the binary vectors that are generated based on static printable strings and dynamic API-n-gram features. As a result, hybrid binary vectors are constructed to represent the characteristics of malware and benign samples. The authors of [63] created the local binary vector for each file by converting each API call into 1 or 0 depending on its presence in the global list.

However, using binary vectors to represent the features is vulnerable to obfuscation techniques that produce irrelevant features such as irrelevant API calls or opcodes without

affecting the overall functionality [101]. Such a representation method seems to be sensitive to the employed feature extraction method. The legitimate features, which are often injected by malware authors in the produced malware to overcome the analysis efforts, can be represented in the malware binary vectors as malware characteristics in the case of the developed extraction method that includes them as malware features.

5.2.2. Frequency-Based Vector

The production of malware variants nowadays is an easy task for the reason that there are various obtainable malicious software code libraries and online tools to reuse and alter the existing malicious codes and then introduce new variants. As a result, the frequency (number of occurrence times) of each extracted feature in each malware sample, such as API calls and opcodes, can be used to identify the similarity between malicious behaviors that belong to multiple variants of the same malware family [44].

To construct the frequency-based vectors, the occurrence times of the extracted features must be considered. Based on the assumption that there are differences between malware and benign programs in the occurrence numbers of their performed functions, frequency-based vectors have been used to represent the differences between malicious and legitimate activities. Therefore, the most frequent features have to be determined for both malware and benign classes [102]. In their study, ref. [70] was concerned about VBasic-based malware, so the VBscript samples were looked up to identify specific functions, methods, and keywords along with their number of occurrences to construct the feature vector. The authors of [44] extracted API calls using n-gram techniques, and then the frequency-based vectors which consist, of the occurrences of each n-gram, were constructed to represent the file characteristics.

Despite the fact that using a frequency-based approach to represent malware on graphs yields discriminating patterns [103], the performance of frequency-based malware detection models is hampered by obfuscation techniques such as dead code insertion, which is capable of updating the distribution of the feature [104].

5.2.3. Weight-Based Vector

The distribution of the extracted features is one of the biggest differences between legitimate and malware classes, and even between malware families. The features that are related to performing malicious behavior have to be heavily distributed in malware samples and are not or rarely distributed in legitimate samples. Furthermore, malware families are determined based on assumptions about how the malware acts and what techniques are used to carry out their own malicious actions [83]. Therefore, among the whole set of extracted features, there are specific features that can be related to particular families and are rarely distributed in others. Based on the above concept, some of the previous studies have represented the extracted text features as numerical weights using a weight-based vector with the help of statistical methods such as Term Frequency-Inverse Document Frequency (TF-IDF), Information Gain (IG), and entropy value.

A weight-based vector consists of the extracted features and their weights. Specific weights are assigned to each feature as their data values using statistical methods such as information gain and TF-IDF. Therefore, the proposed models can distinguish between malware and benign files based on the assigned weights. The authors of [43] calculated the frequency of each extracted word, and then the frequency values were normalized. Further, the Information Gain method has been employed to assign the calculated weights to each feature. Those weights were used to determine where the testing files should be placed, whether in malware or benign files. Another study [81], computed weights for each extracted feature using TF-IDF to select only the most significant features and then represented their samples using the selected features along with their weights as an Attribute Relation File Format (ARFF), with each row in this file representing a weight vector. However, the statistical methods which aim to convert text data into numerical

data to be fed into ML techniques do not capture any contextual relationship between features [85].

5.2.4. Image Characterization-Based Vector

In the gray-scale-based malware detection and classification models, the gray images of malware and benign samples are generated by dividing the PE malware files into 8-bit vectors consisting of hexadecimal values. Those values have been represented in the form of pixels, which range from 0 to 255 to simulate the gradient from the white to the black colors for each pixel in order to represent the malware files as images. The width of the images is predetermined, while the length can be hung based on its size [105]. Therefore, those created images have been utilized in many studies during the representation phase when the generated vectors comprise the numerical data that characterize the created image.

To generate image characteristics-based vectors and represent the malware files as vectors that describe the created malware images from several aspects, there are various algorithms such as Color Layout Descriptor (CLD), Homogeneous Texture Descriptor (HTD), and GIST for obtaining the vector from the generated images. The vectors which are generated to describe the constructed images of malware and benign files have to contain distinguishable values since those values represent several aspects of the images, such as texture and intensity, which are expressed differently in malware and benign files.

In their studies [75], they converted each byte in the binary malware file into one pixel of the image that represents the malware file. The authors of [58] integrated the static and dynamic features after visualizing them as two images. Both images of the original binary file were encoded into a single image. Therefore, the image characterization vector method is applied to generate the vector that consists of numerical data that represents the texture and color of the hybrid image. However, it takes more time to generate the images. Furthermore, representing the files as images requires a large amount of storage space. Additionally, visualization-based models depend on the matching process to classify the testing files. Therefore, samples with few variants in the training data lead to no matching and then misclassification for them.

5.2.5. Weighted Dependency Graph

Because malware detection and classification models that have been developed based on individual features or sequences of features are vulnerable to unreadable insertion and reordering instruction techniques, more complex information such as features dependencies (the probability degree of the dependency between each feature and the others) has to be included and learned by the developed models [80]. Therefore, the features' dependencies have been calculated and introduced as weights that describe the dependency between each two features in the generated graph to represent the malicious behaviors. Several similarity measures, such as the Maximum Weight Subgraph Algorithm (MWSA), NP-similarity, and Same-similarity, have been used to tell the difference between the weighted dependency graph-based behaviors. The results of the similarity measures are then compared to thresholds to see if the examined behavior is similar or different to the learned behaviors.

Calculating weights as indicators to express the probability degree of the dependency between every two characteristics is an alternate representation method for making the extracted feature graphs more representational of the corresponding behaviors. The authors of [80] assigned weights that denote the probability that a dependency relation appears in a malware family to represent the behaviors that belong to that family. A weight-based threshold is used to determine which dependency relationships in the graph of each family have to be used as common behaviors for that family.

5.2.6. Rule-Based Representation

Some of the studies in the literature focused on the rule-based representation method for representing malicious software behavior. There are two main mechanisms for generating the rule-based behaviors, which are: manually and automatically. Manual rules

necessitate significant effort on the part of malware analysts to observe the features by which malicious behaviors are represented, whereas tools such as the YARA tool have been used to generate automatic rules. The rule-based constructed behaviors can be utilized in the detection phase by matching them with the generated rule-based behavior of the test file.

The majority voting decision-making mechanism has been widely used in the previous studies to decide if the test file's rule-based behaviour is more in line with benign or malware rule-based behaviour. However, rule-based malware detection and classification models are limited to recognizing only the malicious activities that are represented in their generated rules. The authors of [54] represented malware and benign behaviors using dynamic API calls, API sequences, and network traffic features by building the rules that are stored in a database to describe malware and benign behaviors, and then leveraging a majority voting method when their model matched the extracted testing files' behaviors with those rules.

5.2.7. Discussion of Feature Representation Methods

According to the existence of each feature in the investigated file, a value of 0 or 1 can be assigned to that feature to represent the examined file using a binary vector. Similarly, instead of the values 0 or 1 that are utilized in the binary vector representation method, the times of occurrences for each feature are computed to describe the sample using a frequency-based vector. Unfortunately, the existence of the features can be changed from one variant to another one that achieves the same malicious activities along with the presence of irrelevant features. Moreover, the frequency-based vector representation method is defeated when the malware developers contribute their samples using obfuscation techniques.

To mitigate the weaknesses of binary/frequency-based representation methods, some researchers attempted to rank the weight value for each feature using statistical methods to represent the investigated file using a weight vector. However, the contextual relationship among the features is not captured by converting the text features into numerical features.

Moreover, the binary files were represented using vectors which consists of numerical information extracted from the malware and benign images that are generated by converting the bytes into pixels that ranged gradually from white to black color via 0 to 250 degrees, but generating the images took a significant period of time and storing the behaviors as images required a large storage capacity.

Some researchers exploited the rules generated by particular tools like the YARA tool to characterize the features that are produced during the extraction phase. Only the harmful behaviors specified in the created rules are identified using the rule-based representation method. Therefore, this representation method eliminates the inefficiency of the matching process.

6. Open Issues

Based on the survey of the recently proposed malware detection and classification models along with reviewing the approaches and techniques that have been used, the shortcomings and usefulness of those approaches and techniques are specified. Therefore, our review could identify the challenges and open issues. The following are the most open issues and suggestions for research directions.

6.1. Obfuscation Techniques

Those techniques were first used to protect the intellectual property of software contactors, but lastly, the writers of malicious programs leveraged those techniques to transfer their malware to different forms that are harder to analyze and detect [106,107]. Such techniques as dead code insertion, registry reassignment, instruction reordering, and instruction substitution have been used to update the characteristics of malware along with achieving the same functions [108,109]. Based on [87], almost 50% of the unknown malicious programs are different variants of the old ones, while [75] reported that only

20% of the new malware represents unseen malware, and 80% of them are the same malware but in different variants. Therefore, producing the malware in several variants that have their own characteristics makes the malware detection and classification models, which have been built based on the signatures, sequences, or frequencies, suffer from poor detection accuracy.

6.2. Evasion Techniques

Through executing particular operations, the evasive malware is capable of recognizing whether they are running in a controlled or real environment. When malware discovers the characteristics that indicate a controlled environment, they immediately alter their behaviors to show different behaviors that are similar to benign behaviors or stop their executions [25]. In addition, the evasive malware can identify the nature of the execution environment by utilizing specific information that is related to sandboxes, debuggers, virtual machines, or monitors as well as waiting for the action of the user, such as mouse move, mouse click, or others, as a condition to their execution starting [110]. Furthermore, according to [111] due to the usage of anti-analysis techniques, around 1% of the scanned malware is not detectable for 64% of anti-virus scanners after one year. Therefore, the malware detection and classification models that obtained the features by running the samples in controlled environments experienced difficulties detecting the evasive malware.

6.3. Zero-Day Malware

Previously unseen malware usually causes harm to the systems through its malicious activities to achieve new attacks. To the best of our knowledge, after each new attack that has occurred by the zero-day malware, there might be zero-day until this malware is discovered [112]. Unknown malware has new characteristics that fulfill their purposes, so the malware detection models that have been designed based on past information are not efficient when attempting to detect zero-day malware [113]. Furthermore, the ratio of malware with new strategies to fulfill their goals as zero-day malware has increased [32,114]. According to [85], around 350,000 zero-day of malware have been produced daily. Therefore, detecting the zero-day malware that presents new characteristics during their attacks is harder using malware detection models that recognize the malware behaviors based on the obtained characteristics from the training data.

6.4. Redundancy and Irrelevant Behaviours

Since machine learning techniques have trouble coping with data that contains redundant and irrelevant features [115], the presence of redundant and/or unnecessary features in datasets is one of the issues of the malware detection community [50,116]. Indeed, the redundant/irrelevant behaviors that are irrelevant for a certain class can significantly raise the operational cost and reduce the accuracy of most learners. Therefore, the task of generating datasets, including feature extraction and selection phases, is challenging and must be continuously improved in order to improve the overall performance of the developed models.

6.5. False Positive/Negative Rate

Malware authors have attempted to make the produced malware accomplish their functions in a way that is consistent with legitimate behavior [117]. Therefore, some characteristics and fingerprints in malicious files and benign samples might be quite similar. Several malware detection approaches are vulnerable to false positive and false negative rates. Though an increase in false positive or false negative rates reduces the model detection accuracy, false positives are far more significant than false negatives in the effective malware detection models. If a legitimate file is mistakenly identified as malicious on a user's computer, the operating system may become unbootable and other applications may become non-working [118].

6.6. Incremental Learning

Malware analysis and detection developers have refined their classifiers to include the most modern malicious activity techniques since malware writers develop malicious software daily. During the collection phase, the preferred training data must represent the behaviors of modern malware by considering the most recent malware files. As a result, older malware behaviors are not captured in the models developed, rendering them undetected. It is a sensitive issue to incorporate adequate historical trends of harmful activity during the sample collection phase so that the models can recognize both recent and older malicious behavior [119]. For example, when the testing data contained older malware, the proposed model in [58] provided a low detection rate. This is because the suggested model was trained considering the behaviors of the most recent malware samples. As a result, the characteristics of older malware behaviors were not represented in the developed model.

7. Future Directions

Even though the existing solutions in the literature review have established the road to developing trustworthy malware detection and classification models, evasive malware detection is still challenging. To detect evasive malware, several approaches have been taken such as: generating API-based evasive malware signatures, discovering evasion behaviors using multiple execution environments, and using the known evasion techniques to detect evasive malware. To the best of our knowledge, each evasive malware detection solution has its own weaknesses. For example, the distinction between evasion techniques that have been used in legitimate behavior and malicious-related evasion techniques is still a challenge. Additionally, it's quite difficult for the developed models, which are learned based on the known evasion techniques, to detect and recognize the unknown ones. Moreover, using several execution environments without high complexity in terms of time and resources is another challenge.

Despite several studies having been done to enhance the evasive malware detection rate, there is no available dataset from which the evasive behaviors are represented. Therefore, creating an evasive behavior dataset would contribute to the efforts of researchers to produce robust solutions. For evasive malware detection purposes, efficient feature extraction and representation techniques are required to extract and represent a feature set that represents evasion techniques related to only malicious behaviors

On the other hand, zero-day malware and unknown malware variants' daily production has been greatly increased since the availability of online tools by which to create new malware or reformat the existing ones using obfuscation techniques to introduce new variants. Therefore, efficient updating learning mechanisms are required to render the developed models to adaptively learn the coming new behaviors. For this end, deep learning techniques in conjunction with unsupervised machine learning techniques can be designed and implemented for updating learning and developing models which are adaptively learning new malicious behaviors.

8. Conclusions

In this survey, we have introduced a comprehensive review on the evolution as well as the trends of malware analysis and detection approaches. Particularly, this survey concerned with the perspectives often ignored or partially studied by previous surveys. For example, exploring the usefulness of each data type according to the utilized analysis approaches, offering a deep taxonomy for malware detection approaches where the detection approaches are presented in more detail than signature-based, behavioral-based, and heuristic-based. This is to provide the research community an opportunity to improve the existing malware detection solutions. Additionally, this survey has associated the feature extraction methods with the employed extraction techniques instead of the analysis approaches to highlight the boundary between the data collection and data extraction phases. A novel taxonomy for methods of feature representation has been presented in this

survey. The root cause problems by which each approach or method for analysis, detection, extraction, and representation suffers from its own drawbacks have been investigated to produce the open issues and suggestions for future research directions.

Author Contributions: Conceptualization, F.A.A. and A.Z.; methodology, F.A.G. and B.A.S.A.-r.; validation, F.A.G. and B.A.S.A.-r.; analysis, F.A.A.; investigation, F.A.A. and A.Z.; resources, T.A.E.E. and A.A.H.E.; writing—original draft preparation, F.A.A.; writing—review and editing, A.Z., F.A.G. and B.A.S.A.-r.; visualization, F.A.A.; supervision, A.Z., F.A.G. and B.A.S.A.-r.; project administration, T.A.E.E. and A.A.H.E.; funding acquisition, F.A.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been supported by the funding from the Deanship of Scientific Research at King Khalid University, Large Groups. (Project under grant number (RGP.2/49/43)).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through Large Groups. (Project under grant number (RGP.2/49/43)).

Conflicts of Interest: There is no conflict of interest.

References

1. Caviglione, L.; Choras, M.; Corona, I.; Janicki, A.; Mazurczyk, W.; Pawlicki, M.; Wasielewska, K. Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection. *IEEE Access* **2021**, *9*, 5371–5396. [CrossRef]
2. Morgan, S. Cybercrime Damages $6 Trillion by 2021. 2017. Available online: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/ (accessed on 15 July 2021).
3. Cannarile, A.; Dentamaro, V.; Galantucci, S.; Iannacone, A.; Impedovo, D.; Pirlo, G. Comparing Deep Learning and Shallow Learning Techniques for API Calls Malware Prediction: A Study. *Appl. Sci.* **2022**, *12*, 1645. [CrossRef]
4. Villalba, L.J.G.; Orozco, A.L.S.; Vivar, A.L.; Vega, E.A.A.; Kim, T.-H. Ransomware Automatic Data Acquisition Tool. *IEEE Access* **2018**, *6*, 55043–55051. [CrossRef]
5. Urooj, U.; Al-Rimy, B.A.S.; Zainal, A.; Ghaleb, F.A.; Rassam, M.A. Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions. *Appl. Sci.* **2022**, *12*, 172. [CrossRef]
6. Hansen, S.S.; Larsen, T.M.T.; Stevanovic, M.; Pedersen, J.M. An approach for detection and family classification of malware based on behavioral analysis. In Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC), Kauai, HI, USA, 15–18 February 2016; pp. 1–5. [CrossRef]
7. Vignau, B.; Khoury, R.; Halle, S. 10 Years of IoT Malware: A Feature-Based Taxonomy. In Proceedings of the 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Sofia, Bulgaria, 22–26 July 2019; pp. 458–465. [CrossRef]
8. Asam, M.; Hussain, S.J.; Mohatram, M.; Khan, S.H.; Jamal, T.; Zafar, A.; Khan, A.; Ali, M.U.; Zahoora, U. Detection of exceptional malware variants using deep boosted feature spaces and machine learning. *Appl. Sci.* **2021**, *11*, 10464. [CrossRef]
9. Sahay, S.K.; Sharma, A.; Rathore, H. Evolution of Malware and Its Detection Techniques. In *Advances in Intelligent Systems and Computing*; Springer: Singapore, 2020; Volume 933, pp. 139–150.
10. Kakisim, A.G.; Nar, M.; Sogukpinar, I. Metamorphic malware identification using engine-specific patterns based on co-opcode graphs. *Comput. Stand. Interfaces* **2019**, *71*, 103443. [CrossRef]
11. Sihwail, R.; Omar, K.; Ariffin, K.A.Z. A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis. *Int. J. Adv. Sci. Eng. Inf. Technol.* **2018**, *8*, 1662. [CrossRef]
12. Vidal, J.M.; Orozco, A.L.S.; Villalba, L.J.G. Alert correlation framework for malware detection by anomaly-based packet payload analysis. *J. Netw. Comput. Appl.* **2017**, *97*, 11–22. [CrossRef]
13. Saxena, S.; Mancoridis, S. Malware Detection using Behavioral Whitelisting of Computer Systems. In Proceedings of the 2019 IEEE International Symposium on Technologies for Homeland Security (HST), Greater Boston, MA, USA, 5–6 November 2019; pp. 1–6. [CrossRef]
14. Gajrani, J.; Sarswat, J.; Tripathi, M.; Laxmi, V.; Gaur, M.S.; Conti, M. A robust dynamic analysis system preventing SandBox detection by android malware. In Proceedings of the ACM International Conference Proceeding Series, Sochi, Russian, 8–10 September 2015. [CrossRef]
15. Banin, S.; Shalaginov, A.; Franke, K. Memory access patterns for malware detection. *Nor. Inf.* **2016**, *96*, 107.
16. Aslan, O.; Samet, R. A Comprehensive Review on Malware Detection Approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]
17. AAbusitta, A.; Li, M.Q.; Fung, B.C. Malware classification and composition analysis: A survey of recent developments. *J. Inf. Secur. Appl.* **2021**, *59*, 102828. [CrossRef]

18. Deylami, H.M.; Muniyandi, R.C.; Ardekani, I.T.; Sarrafzadeh, A. Taxonomy of malware detection techniques: A systematic literature review. In Proceedings of the 2016 14th Annual Conference on Privacy, Security and Trust (PST), Auckland, New Zealand, 12–14 December 2016; pp. 629–636. [CrossRef]
19. Tahir, R. A Study on Malware and Malware Detection Techniques. *Int. J. Educ. Manag. Eng.* **2018**, *8*, 20–30. [CrossRef]
20. Gibert, D.; Mateu, C.; Planes, J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *J. Netw. Comput. Appl.* **2020**, *153*, 102526. [CrossRef]
21. Alsmadi, T.; Alqudah, N. A Survey on malware detection techniques. In Proceedings of the 2021 International Conference on Information Technology (ICIT), Amman, Jordan, 14–15 July 2021; IEEE: New York, NY, USA, 2021; pp. 371–376. [CrossRef]
22. Panchariya, H.; Bharkad, S. Comparative Analysis of Feature Extraction Methods of Malware Detection. *IOSR J. Comput. Eng.* **2014**, *16*, 49–54. [CrossRef]
23. El Merabet, H.; Hajraoui, A. A Survey of Malware Detection Techniques based on Machine Learning. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 366–373. [CrossRef]
24. Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R.P.; Tang, J.; Liu, H. Feature selection: A data perspective. *ACM Comput. Surv.* **2017**, *50*, 1–45. [CrossRef]
25. Oyama, Y. Trends of anti-analysis operations of malwares observed in API call logs. *J. Comput. Virol. Hacking Tech.* **2018**, *14*, 69–85. [CrossRef]
26. Sicato, J.C.S.; Sharma, P.K.; Loia, V.; Park, J.H. Vpnfilter malware analysis on cyber threat in smart home network. *Appl. Sci.* **2019**, *9*, 2763. [CrossRef]
27. Chakkaravarthy, S.S.; Sangeetha, D.; Vaidehi, V. A Survey on malware analysis and mitigation techniques. *Comput. Sci. Rev.* **2019**, *32*, 1–23. [CrossRef]
28. Souri, A.; Hosseini, R. A state-of-the-art survey of malware detection approaches using data mining techniques. *Hum.-Cent. Comput. Inf. Sci.* **2018**, *8*, 3. [CrossRef]
29. Balkrishna, S.; Me, K.; Pratishthan, V.; Shital, M.; Kuber, B. A Survey on Data Mining Methods for Malware Detection. *Int. J. Eng. Res. Gen. Sci.* **2014**, *2*, 672–675.
30. Naz, S.; Singh, D.K. Review of Machine Learning Methods for Windows Malware Detection. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 6–8 July 2019; pp. 1–6. [CrossRef]
31. Zakeri, M.; Daneshgar, F.F.; Abbaspour, M. A static heuristic approach to detecting malware targets. *Secur. Commun. Netw.* **2015**, *8*, 30. [CrossRef]
32. Zelinka, I.; Amer, E. An Ensemble-Based Malware Detection Model Using Minimum Feature Set. *Mendel* **2019**, *25*, 1–10. [CrossRef]
33. Denzer, T.; Shalaginov, A.; Dyrkolbotn, G.O. Intelligent Windows Malware Type Detection based on Multiple Sources of Dynamic Characteristics. *Nis. J.* **2019**, *12*, 20.
34. PSeshagiri, P.; Vazhayil, A.; Sriram, P. AMA: Static Code Analysis of Web Page for the Detection of Malicious Scripts. *Procedia Comput. Sci.* **2016**, *93*, 768–773. [CrossRef]
35. Wael, D.; Sayed, S.G.; AbdelBaki, N. Enhanced Approach to Detect Malicious VBScript Files Based on Data Mining Techniques. *Procedia Comput. Sci.* **2018**, *141*, 552–558. [CrossRef]
36. Ling, Y.T.; Sani, N.F.M.; Abdullah, M.T.; Hamid, N.A.W.A. Nonnegative matrix factorization and metamorphic malware detection. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 195–208. [CrossRef]
37. Kumar, A.; Kuppusamy, K.; Aghila, G. A learning model to detect maliciousness of portable executable using integrated feature set. *J. King Saud Univ.-Comput. Inf. Sci.* **2019**, *31*, 252–265. [CrossRef]
38. Khodamoradi, P.; Fazlali, M.; Mardukhi, F.; Nosrati, M. Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms. In Proceedings of the 2015 18th CSI International Symposium on Computer Architecture and Digital Systems (CADS), Tehran, Iran, 7–8 October 2015; pp. 1–6. [CrossRef]
39. Hashemi, H.; Azmoodeh, A.; Hamzeh, A.; Hashemi, S. Graph embedding as a new approach for unknown malware detection. *J. Comput. Virol. Hacking Tech.* **2017**, *13*, 153–166. [CrossRef]
40. Euh, S.; Lee, H.; Kim, D.; Hwang, D. Comparative Analysis of Low-Dimensional Features and Tree-Based Ensembles for Malware Detection Systems. *IEEE Access* **2020**, *8*, 76796–76808. [CrossRef]
41. Khalilian, A.; Nourazar, A.; Vahidi-Asl, M.; Haghighi, H. G3MD: Mining frequent opcode sub-graphs for metamorphic malware detection of existing families. *Expert Syst. Appl.* **2018**, *112*, 15–33. [CrossRef]
42. Lu, R. Malware Detection with LSTM using Opcode Language. *arXiv* **2019**, arXiv:1906.04593.
43. Choudhary, S.; Vidyarthi, M.D. A Simple Method for Detection of Metamorphic Malware using Dynamic Analysis and Text Mining. *Procedia Comput. Sci.* **2015**, *54*, 265–270. [CrossRef]
44. Galal, H.S.; Mahdy, Y.B.; Atiea, M.A. Behavior-based features model for malware detection. *J. Comput. Virol. Hacking Tech.* **2016**, *12*, 59–67. [CrossRef]
45. Mosli, R.; Li, R.; Yuan, B.; Pan, Y. Automated malware detection using artifacts in forensic memory images. In Proceedings of the 2016 IEEE Symposium on Technologies for Homeland Security (HST), Waltham, MA, USA, 10–12 May 2016; pp. 1–6. [CrossRef]
46. Hwang, J.; Kim, J.; Lee, S.; Kim, K. Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques. *Wirel. Pers. Commun.* **2020**, *112*, 2597–2609. [CrossRef]

47. Jerlin, M.A.; Marimuthu, K. A New Malware Detection System Using Machine Learning Techniques for API Call Sequences. *J. Appl. Secur. Res.* **2018**, *13*, 45–62. [CrossRef]
48. Kim, H.; Kim, J.; Kim, Y.; Kim, I.; Kim, K.J.; Kim, H. Improvement of malware detection and classification using API call sequence alignment and visualization. *Cluster Comput.* **2019**, *22*, 921–929. [CrossRef]
49. Fasano, F.; Martinelli, F.; Mercaldo, F.; Santone, A. Energy Consumption Metrics for Mobile Device Dynamic Malware Detection. *Procedia Comput. Sci.* **2019**, *159*, 1045–1052. [CrossRef]
50. Ahmed, Y.A.; Koçer, B.; Huda, S.; Al-Rimy, B.A.S.; Hassan, M.M. A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. *J. Netw. Comput. Appl.* **2020**, *167*, 102753. [CrossRef]
51. Singh, J.; Singh, J. Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms. *Inf. Softw. Technol.* **2020**, *121*, 106273. [CrossRef]
52. Norouzi, M.; Souri, A.; Zamini, M.S. A Data Mining Classification Approach for Behavioral Malware Detection. *J. Comput. Netw. Commun.* **2016**, *2016*, 1–9. [CrossRef]
53. Arabo, A.; Dijoux, R.; Poulain, T.; Chevalier, G. Detecting Ransomware Using Process Behavior Analysis. *Procedia Comput. Sci.* **2020**, *168*, 289–296. [CrossRef]
54. Belaoued, M.; Boukellal, A.; Koalal, M.A.; Derhab, A.; Mazouzi, S.; Khan, F.A. Combined dynamic multi-feature and rule-based behavior for accurate malware detection. *Int. J. Distrib. Sens. Netw.* **2019**, *15*, 155014771988990. [CrossRef]
55. Fraley, J.B.; Figueroa, M. Polymorphic malware detection using topological feature extraction with data mining. In Proceedings of the SoutheastCon 2016, Norfolk, VA, USA, 30 March–3 April 2016; pp. 1–7. [CrossRef]
56. Ndibanje, B.; Kim, K.H.; Kang, Y.J.; Kim, H.H.; Kim, T.Y.; Lee, H.J. Cross-Method-Based Analysis and Classification of Malicious Behavior by API Calls Extraction. *Appl. Sci.* **2019**, *9*, 239. [CrossRef]
57. Zhong, W.; Gu, F. A multi-level deep learning system for malware detection. *Expert Syst. Appl.* **2019**, *133*, 151–162. [CrossRef]
58. Huang, X.; Ma, L.; Yang, W.; Zhong, Y. A Method for Windows Malware Detection Based on Deep Learning. *J. Signal Process. Syst.* **2021**, *93*, 265–273. [CrossRef]
59. Damodaran, A.; Di Troia, F.; Visaggio, C.A.; Austin, T.; Stamp, M. A comparison of static, dynamic, and hybrid analysis for malware detection. *J. Comput. Virol. Hacking Tech.* **2017**, *13*, 1–12. [CrossRef]
60. Wael, D.; Shosha, A.; Sayed, S.G. Malicious VBScript detection algorithm based on data-mining techniques. In Proceedings of the 2017 International Conference on Advanced Control. Circuits Systems (ACCS) Systems & 2017 International Conference on New Paradigms in Electronics & Information Technology (PEIT), Alexandria, Egypt, 5–8 November 2017; pp. 112–116. [CrossRef]
61. Fuyong, Z.; Tiezhu, Z. Malware Detection and Classification Based on N-Grams Attribute Similarity. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017; pp. 793–796. [CrossRef]
62. Ki, Y.; Kim, E.; Kim, H.K. A Novel Approach to Detect Malware Based on API Call Sequence Analysis. *Int. J. Distrib. Sens. Netw.* **2015**, *11*, 659101. [CrossRef]
63. Sihwail, R.; Omar, K.; Ariffin, K.A.Z.; Al Afghani, S. Malware Detection Approach Based on Artifacts in Memory Image and Dynamic Analysis. *Appl. Sci.* **2019**, *9*, 3680. [CrossRef]
64. Kumar, R.; Vaishakh, A.R.E. Detection of Obfuscation in Java Malware. *Procedia Comput. Sci.* **2015**, *78*, 521–529. [CrossRef]
65. Liu, L.; Wang, B.; Yu, B.; Zhong, Q. Automatic malware classification and new malware detection using machine learning. *Front. Inf. Technol. Electron. Eng.* **2017**, *18*, 1336–1347. [CrossRef]
66. Li, X.; Qiu, K.; Qian, C.; Zhao, G. An Adversarial Machine Learning Method Based on OpCode N-grams Feature in Malware Detection. In Proceedings of the 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC), Hong Kong, China, 27–30 July 2020; pp. 380–387. [CrossRef]
67. Zhang, H.; Xiao, X.; Mercaldo, F.; Ni, S.; Martinelli, F.; Sangaiah, A.K. Classification of ransomware families with machine learning based on N-gram of opcodes. *Futur. Gener. Comput. Syst.* **2019**, *90*, 211–221. [CrossRef]
68. Belaoued, M.; Mazouzi, S. A chi-square-based decision for real-time malware detection using PE-file features. *J. Inf. Process. Syst.* **2016**, *12*, 644–660. [CrossRef]
69. Burnap, P.; French, R.; Turner, F.; Jones, K. Malware classification using self organising feature maps and machine activity data. *Comput. Secur.* **2018**, *73*, 399–410. [CrossRef]
70. Ali, M.; Shiaeles, S.; Bendiab, G.; Ghita, B. MALGRA: Machine Learning and N-Gram Malware Feature Extraction and Detection System. *Electronics* **2020**, *9*, 1777. [CrossRef]
71. JVidal, M.; Orozco, A.L.S.; Villalba, L.J.G. Malware Detection in Mobile Devices by Analyzing Sequences of System Calls. *Int. J. Comput. Electr. Autom. Control. Inf. Eng.* **2017**, *11*, 588–592.
72. Shijo, P.; Salim, A. Integrated Static and Dynamic Analysis for Malware Detection. *Procedia Comput. Sci.* **2015**, *46*, 804–811. [CrossRef]
73. Darshan, S.L.S.; Jaidhar, C.D. Windows malware detection system based on LSVC recommended hybrid features. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 127–146. [CrossRef]
74. Darshan, S.L.S.; Jaidhar, C.D. An empirical study to estimate the stability of random forest classifier on the hybrid features recommended by filter based feature selection technique. *Int. J. Mach. Learn. Cybern.* **2020**, *11*, 339–358. [CrossRef]
75. Kang, J.; Won, Y. A study on variant malware detection techniques using static and dynamic features. *J. Inf. Process. Syst.* **2020**, *16*, 882–895. [CrossRef]

76. Alieyan, K.; Almomani, A.; Anbar, M.; Alauthman, M.; Abdullah, R.; Gupta, B.B. DNS rule-based schema to botnet detection. *Enterp. Inf. Syst.* **2021**, *15*, 545–564. [CrossRef]
77. Du, D.; Sun, Y.; Ma, Y.; Xiao, F. A Novel Approach to Detect Malware Variants Based on Classified Behaviors. *IEEE Access* **2019**, *7*, 81770–81782. [CrossRef]
78. Catak, F.O.; Yazı, A.F.; Elezaj, O.; Ahmed, J. Deep learning based Sequential model for malware analysis using Windows exe API Calls. *PeerJ Comput. Sci.* **2020**, *6*, e285. [CrossRef] [PubMed]
79. Shabtai, A.; Moskovitch, R.; Elovici, Y.; Glezer, C. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. *Inf. Secur. Tech. Rep.* **2009**, *14*, 16–29. [CrossRef]
80. Ding, Y.; Xia, X.; Chen, S.; Li, Y. A malware detection method based on family behavior graph. *Comput. Secur.* **2018**, *73*, 73–86. [CrossRef]
81. More, S.S.; Gaikwad, P.P. Trust-based Voting Method for Efficient Malware Detection. *Procedia Comput. Sci.* **2016**, *79*, 657–667. [CrossRef]
82. Zhao, Y.; Bo, B.; Feng, Y.; Xu, C.; Yu, B. A Feature Extraction Method of Hybrid Gram for Malicious Behavior Based on Machine Learning. *Secur. Commun. Netw.* **2019**, *2019*, 1–8. [CrossRef]
83. Banin, S.; Dyrkolbotn, G.O. Multinomial malware classification via low-level features. *Digit. Investig.* **2018**, *26*, S107–S117. [CrossRef]
84. Daku, H.; Zavarsky, P.; Malik, Y. Behavioral-Based Classification and Identification of Ransomware Variants Using Machine Learning. In Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 1560–1564. [CrossRef]
85. Amer, E.; Zelinka, I. A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence. *Comput. Secur.* **2020**, *92*, 101760. [CrossRef]
86. Liu, X.; Lin, Y.; Li, H.; Zhang, J. A novel method for malware detection on ML-based visualization technique. *Comput. Secur.* **2020**, *89*, 101682. [CrossRef]
87. Zhang, J.; Qin, Z.; Yin, H.; Ou, L.; Zhang, K. A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding. *Comput. Secur.* **2019**, *84*, 376–392. [CrossRef]
88. Jha, S.; Prashar, D.; Long, H.V.; Taniar, D. Recurrent neural network for detecting malware. *Comput. Secur.* **2020**, *99*, 102037. [CrossRef]
89. Santos, I.; Brezo, F.; Nieves, J.; Penya, Y.K.; Sanz, B.; Laorden, C.; Bringas, P.G. Idea: Opcode-sequence-based Malware Detection. In *International Symposium on Engineering Secure Software and Systems*; Springer: Berlin/Heidelberg, Germany, 2010.
90. Garg, V.; Yadav, R.K. Malware Detection based on API Calls Frequency. In Proceedings of the 2019 4th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 21–22 November 2019; pp. 400–404. [CrossRef]
91. Nikolopoulos, S.D.; Polenakis, I. A graph-based model for malware detection and classification using system-call groups. *J. Comput. Virol. Hacking Tech.* **2017**, *13*, 29–46. [CrossRef]
92. Ghiasi, M.; Sami, A.; Salehi, Z. Dynamic VSA: A framework for malware detection based on register contents. *Eng. Appl. Artif. Intell.* **2015**, *44*, 111–122. [CrossRef]
93. SJang, S.; Li, S.; Sung, Y. Generative adversarial network for global image-based local image to improve Malware classification using convolutional neural network. *Appl. Sci.* **2020**, *10*, 7585. [CrossRef]
94. Nisa, M.; Shah, J.H.; Kanwal, S.; Raza, M.; Khan, M.A.; Damaševičius, R.; Blažauskas, T. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. *Appl. Sci.* **2020**, *10*, 4966. [CrossRef]
95. Allen, F.E. Control flow analysis. *ACM SIGPLAN Not.* **1970**, *5*, 1–19. [CrossRef]
96. Zhao, Z. A virus detection scheme based on features of Control Flow Graph. In Proceedings of the 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce, AIMSEC 2011—Proceedings, Zhengzhou, China, 8–10 August 2011; pp. 943–947. [CrossRef]
97. Alami, N.; Meknassi, M.; En-Nahnahi, N. Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning. *Expert Syst. Appl.* **2019**, *123*, 195–211. [CrossRef]
98. Amer, E.; El-Sappagh, S.; Hu, J. Contextual identification of windows malware through semantic interpretation of API call sequence. *Appl. Sci.* **2020**, *10*, 7673. [CrossRef]
99. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the ACM International Conference Proceeding Series, Rourkela Odisha, India, 12–14 February 2011. [CrossRef]
100. El-Shafai, W.; Almomani, I.; AlKhayer, A. Visualized malware multi-classification framework using fine-tuned cnn-based transfer learning models. *Appl. Sci.* **2021**, *11*, 6446. [CrossRef]
101. Canfora, G.; Di Sorbo, A.; Mercaldo, F.; Visaggio, C.A. Obfuscation techniques against signature-based detection: A case study. In Proceedings of the 2015 Mobile Systems Technologies Workshop: Architecture, Technology Trends, and Memory Solutions, MST 2015, Milano, Italy, 22 May 2015; pp. 21–26. [CrossRef]
102. Yewale, A.; Singh, M. Malware detection based on opcode frequency. In Proceedings of the 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), Tamil Nadu, India, 25–27 May 2016; pp. 646–649. [CrossRef]

103. Wuechner, T.; Cislak, A.; Ochoa, M.; Pretschner, A. Leveraging compression-based graph mining for behavior-based malware detection. *IEEE Trans. Dependable Secur. Comput.* **2019**, *16*, 99–112. [CrossRef]
104. Mirzazadeh, R.; Moattar, M.H.; Jahan, M.V. Metamorphic Malware Detection Using Linear Discriminant Analysis and Graph Similarity Reza. In Proceedings of the 2015 5th International Conference on Computer and Knowledge Engineering (ICCKE), Iran, Islamic, 29–30 October 2015.
105. Parmuval, P.; Hasan, M.; Patel, S. Malware Family Detection Approach using Image Processing Techniques: Visualization Technique. *Int. J. Comput. Appl. Technol. Res.* **2018**, *7*, 129–132. [CrossRef]
106. You, I.; Yim, K. Malware Obfuscation Techniques: A Brief Survey. In Proceedings of the 2010 International Conference on Broadband, Wireless Computing, Communication and Applications, Fukuoka, Japan, 4–6 November 2010; pp. 297–300. [CrossRef]
107. Vidal, J.M.; Castro, J.M.; Orozco, A.S.; Villalba, L.G. Evolutions of Evasion Techniques against network Intrusion Detection Systems. In Proceedings of the ICIT 2013 The 6th International conference on Information Technology, Amman, Jordan, 8 May 2013.
108. Wong, W.; Stamp, M. Hunting for metamorphic engines. *J. Comput. Virol.* **2006**, *2*, 211–229. [CrossRef]
109. Christodorescu, M.; Jha, S. Static analysis of executables to detect malicious patterns. In Proceedings of the 12th USENIX Security Symposium, Washington, DC, USA, 4–8 August 2003; pp. 169–186.
110. Veerappan, C.S.; Keong, P.L.K.; Tang, Z.; Tan, F. Taxonomy on malware evasion countermeasures techniques. In Proceedings of the 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, 5–8 February 2018; pp. 558–563. [CrossRef]
111. Kruegel, C. Evasive Malware Exposed and Deconstructed (Talk). In Proceedings of the RSA Conference, San Francisco, CA, USA, 20–24 April 2015; pp. 12–20.
112. Venkatraman, S.; Alazab, M. Use of Data Visualisation for Zero-Day Malware Detection. *Secur. Commun. Netw.* **2018**, *2018*, 1728303. [CrossRef]
113. Millar, S.; McLaughlin, N.; del Rincon, J.M.; Miller, P. Multi-view deep learning for zero-day Android malware detection. *J. Inf. Secur. Appl.* **2021**, *58*, 102718. [CrossRef]
114. Yang, S.; Li, S.; Chen, W.; Liu, Y. A Real-Time and Adaptive-Learning Malware Detection Method Based on API-Pair Graph. *IEEE Access* **2020**, *8*, 208120–208135. [CrossRef]
115. Letteri, I.; di Cecco, A.; della Penna, G. Dataset Optimization Strategies for MalwareTraffic Detection. *arXiv* **2020**, arXiv:2009.11347.
116. Parrales-Bravo, F.; Torres-Urresto, J.; Avila-Maldonado, D.; Barzola-Monteses, J. Relevant and Non-Redundant Feature Subset Selection Applied to the Detection of Malware in a Network. In Proceedings of the 2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM), Cuenca, Ecuador, 12–15 October 2021; pp. 1–6. [CrossRef]
117. Yoo, S.; Kim, S.; Kim, S.; Kang, B.B. AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification. *Inf. Sci.* **2021**, *546*, 420–435. [CrossRef]
118. Gavrilut, D.; Benchea, R.; Vatamanu, C. Optimized Zero False Positives Perceptron Training for Malware Detection. In Proceedings of the 2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 26–29 September 2012; pp. 247–253. [CrossRef]
119. Najari, S. Malware Detection Using Data Mining Techniques. *Int. J. Intell. Inf. Syst.* **2014**, *3*, 33. [CrossRef]