A Comprehensive Analysis of Explainable AI for Malware Hunting

MOHD SAQIB*, School of Information Studies, McGill University, Montreal, Canada
SAMANEH MAHDAVIFAR†, School of Information Studies, McGill University, Montreal, Canada
BENJAMIN C. M. FUNG‡, School of Information Studies, McGill University, Montreal, Canada
PHILIPPE CHARLAND§, Mission Critical Cyber Security Section, Defence Research and Development Canada Valcartier, Quebec City, Canada

In the past decade, the number of malware variants has increased rapidly. Many researchers have proposed to detect malware using intelligent techniques, such as Machine Learning (ML) and Deep Learning (DL), which have high accuracy and precision. These methods, however, suffer from being opaque in the decision-making process. Therefore, we need Artificial Intelligence (AI)-based models to be explainable, interpretable, and transparent to be reliable and trustworthy. In this survey, we reviewed articles related to Explainable AI (XAI) and their application to the significant scope of malware detection. The article encompasses a comprehensive examination of various XAI algorithms employed in malware analysis. Moreover, we have addressed the characteristics, challenges, and requirements in malware analysis that cannot be accommodated by standard XAI methods. We discussed that even though Explainable Malware Detection (EMD) models provide explainability, they make an AI-based model more vulnerable to adversarial attacks. We also propose a framework that assigns a level of explainability to each XAI malware analysis model, based on the security features involved in each method. In summary, the proposed project focuses on combining XAI and malware analysis to apply XAI models for scrutinizing the opaque nature of AI systems and their applications to malware analysis.

CCS Concepts: • **Security and privacy → Malware and its mitigation**; • **Computing methodologies → Artificial intelligence**.

Additional Key Words and Phrases: Explainable AI , Malware Detection, Malware Analysis

1 Introduction

Today, many critical infrastructures in our society are connected to the Internet to reduce operating costs or simplify control. These cyber-physical systems are susceptible to various vulnerabilities that could be exploited by adversaries by injecting malware and conducting malicious activities, such as information theft and ransom collection [3].

*Mohd Saqib is responsible for survey screening, designed and completed the initial writing of this article.
†Samaneh Mahdavifar reviewed, revised and extended the study.
‡Benjamin C. M. Fung provided guidance throughout the manuscript preparation process and provided feedback at every stage of the project.
§Philippe Charland provided guidance throughout the manuscript preparation process and provided feedback at every stage of the project.

Authors' Contact Information: Mohd Saqib, School of Information Studies, McGill University, Montreal, Quebec, Canada; e-mail: mohd.saqib@mail.mcgill.ca; Samaneh Mahdavifar, School of Information Studies, McGill University, Montreal, Quebec, Canada; e-mail: samaneh.mahdavifar@mcgill.ca; Benjamin C. M. Fung, School of Information Studies, McGill University, Montreal, Quebec, Canada; e-mail: ben.fung@mcgill.ca; Philippe Charland, Mission Critical Cyber Security Section, Defence Research and Development Canada Valcartier, Quebec City, Quebec, Canada; e-mail: philippe.charland@drdc-rddc.gc.ca.

ACM 1557-7341/2024/7-ART
https://doi.org/10.1145/3677374

Importance of the topic: Cybersecurity has been enhanced using AI in many remarkable ways, including malware detection [53, 74], code similarity [24], intrusion detection [92], attack prediction [130], and digital forensics [48]. Detecting malicious or suspicious activity in time may prevent significant loss. As these tasks are crucial, the analytical model should be accurate and transparent. When a Portable Executable (PE) file is identified as malware, it is essential to specify which characteristics, referred to as low-level explanations, or which types of malicious activities, referred to as high-level explanations, contributed to this determination.

To construct a malware detection model, we need to reverse engineer PE files and analyze the resulting assembly code. The different paradigms that have been proposed for malware analysis fall into three main categories: Static, Dynamic, and Hybrid approaches that can be signature-based [83, 149], behavior-based [38, 66], data flow graphs [134, 135, 151], control flow graphs (CFG) [4, 15, 25, 69], OpCode oriented [37], Grayscale image analysis [64, 125], and Executable and Linkable Format (ELF) header analysis [14, 112], etc. However, all the studies mentioned have their shortcomings, e.g., signature-based models require frequent updates and behavior-oriented methods rely on predefined malicious activities. Furthermore, except for a few, previous studies failed to be robust and against code obfuscation and the rapidly growing malware variants [141].

The extension of the computational power of computers and the development of various DL approaches have made the analysis of security data easier. Malware detection using DL has already been noticeably explored. For instance, D'Angelo et al.[28] presented an integration of transfer learning and federated learning approaches to address regression issues. The paper demonstrates a significant advancement in malware detection for IoT devices by combining privacy preservation with high accuracy and efficiency, overcoming the limitations of existing federated learning methods. Similarly,[131] conducted a review of malicious traffic analysis. Moreover, intrusion detection using a DL-based model has been presented in [63], [120], and [75]. Convolutional Neural Networks (CNNs) have also been used in intrusion detection [147]. The authors have developed a CNN architecture to detect intrusion and malicious activity on the Web. Similarly, the study employed an extended version of CNN named Deep CNN or DCNN to classify the malware family [137]. Another advanced version of Artificial Neural Networks (ANNs) known as Autoencoder has been used in feature selection and other malware analysis-related tasks [145].

Different DL algorithms have also been used in other types of analysis, such as NLP-based malware detection, including, but not limited to, Recurrent Neural Networks (RNNs), e.g., Gated recurrent units (GRUs) [140], Long Short-Term Memory (LSTM) [47], and Bi-LSTM [17]. Although the above-mentioned state-of-the-art DL algorithms have shown their significance in malware detection in terms of precision and accuracy, they are still opaque in explaining the reason why a decision was made.

In 2016, the Defense Advanced Research Projects Agency (DARPA)[1] came up with a new AI concept named Explainable AI (XAI), to reduce the black-boxing of Deep Learning (DL) models and strengthen transparency, interpretability and explainability. With the conception of XAI, many researchers started to work on the new dimension of explainable models in different fields, including malware detection because it helps stakeholders understand and trust the decisions made by AI systems. Transparency in XAI ensures that these systems can be audited and validated, addressing potential biases and errors in AI-driven decisions. This is crucial in security contexts where the reasons behind labeling software or activities as malicious must be clear and justifiable.

We conducted a comprehensive literature survey on XAI methods for malware analysis (refer to Table 1) and identified several areas where further research could be beneficial. Specifically, areas such as XAI model evaluation, comparative analysis of classification methods, used datasets, and adversarial attacks as limitations, are crucial for explaining and exposing the inadequacies of traditional XAI methods in effectively addressing EMD. Given the unique and complex nature of malware threats, traditional methods, while useful, fall short in capturing the multi-dimensional and evasive characteristics of malware. This limitation highlights the need for the

[1]https://www.darpa.mil/program/explainable-artificial-intelligence

development of new XAI approaches or the adaptation of existing ones to better handle the intricacies of malware code and behavior. Moreover, establishing robust, objective metrics for evaluating the interpretability of malware detection models is critical to advancing the field. Enhancing the resilience of these models against adversarial attacks is also crucial for maintaining the integrity and reliability of malware analyses. Addressing these core challenges will not only improve model interpretability but also ensure that such interpretations are trustworthy and practical in real-world scenarios. Moreover, the method of communication should change according to the underlying security data, so it can convey relevant information as a true means of representation to the end user. Various models proposed by researchers are already inspired by Natural Language Processing (NLP) or image recognition, and although they are efficient for textual/image data, they need to be highly customized for this type of security data.

Malware analysts need to comprehend the features and malicious behaviors behind a prediction for a particular malware family. This way, they can later adjust the malware detection systems to be able to identify similar patterns and anomalies as malware [13].

This article bridges the gap between XAI and malware detection by reviewing and enriching recent studies published by top publishers. This study aims to address the following research questions: 1) How can XAI improve transparency and trust in malware detection systems? (Section 4) 2) What are the current state-of-the-art XAI methods used in malware analysis, and how effective are they? (Section 4.3) 3) What are the challenges and limitations of these methods, and how can they be overcome? (Sections 5.2, 5.4, 5.5) 4) What are the potential ways of communication for malware analysts and cybersecurity stakeholders? (Section 5.3)

This work also demonstrates the model construction of malware detection and classification using XAI. XAI ensures to overcome the black-box nature of DL algorithms and provide transparency, enhanced interpretability, and explainability to the deep models, along with probability calculation for each prediction. In this survey, we addressed XAI for malware analysis and explored all the existing endeavors in this domain, with their challenges and limitations. The contributions of this review paper are as follows:

- We conducted a review of XAI methods for malware analysis published in top journals between 2016 and 2023, evaluating their metrics, discriminating power, and interpretability (Section 4.3, 4.4).
- We presented several possible solutions that XAI offers for the problems related to malware detection and discuss adversarial attacks, as a limitation of some XAI models (Section 5.2, 5.4, 5.5).
- We identified a lack of generalization in evaluating XAI models for malware analysis and proposed a metric for evaluating the explainability of XAI models at different levels of explanations (Section 5.1.2).
- We proposed a framework for evaluating the explainability of XAI models for malware analysis based on different levels of explanations and used a taxonomy to assess how understandable the explanations are. We also applied this taxonomy to previous studies and discuss their communication level (Section 5.3).

The structure of the article is organized as follows: Section 2 presents an all-encompassing review of the relevant literature and contextual background. Section 3 covers malware analysis, including both traditional and automated techniques employing traditional ML or DL approaches, along with a thorough discussion of datasets. In Section 4, the XAI methods for malware analysis are discussed in detail. Section 5 discusses the challenges. Finally, Section 6 concludes the article, highlighting future research avenues.

2 Related survey articles

There has been a considerable number of publications in the domain of malware analysis using DL. For instance, in [53], the author used transfer learning to classify malware images. Similarly, [131] covered a review of malicious traffic analysis using ML. Moreover, Sohi et al. [120] used a DL-based model for intrusion detection. Subsequently, these authors recognized the sensitivity of the models and began to focus on their interpretability.

'Right to Explanation' was the primary reason behind XAI-models [127]. Afterwards, many researchers started working on interpretability in distinctive subdomains, including malware detection. For example, in [106], the authors interpreted the ANN model, and Lacave et al. [57] explained the Bayesian Neural Network (BNN). Not only DL algorithms but traditional ML models also sometimes need to enhance the level of explainability, such as Support Vector Machine (SVM) and logistic regression. Martens et al. [80] presented a comprehensive rule extraction-based study using SVM. Numerous researchers also worked on a graphical drawing of the black-box DL models using heatmap [22]. The authors in [146] provide a survey study on visual analytics techniques for ML.

To prove the significance of our research, we studied and analyzed papers on XAI for malware detection by following the PRISMA model [113]. We selected renowned publishers and conferences (see details in Appendix, Section 7) and Table 6 to ensure the comprehensiveness and relevance of our analysis. Table 1 presents a summary of the topics that have been covered in the review papers. Aslan et al. [13] provide deep insights into the various approaches for malware detection, including heuristic-based algorithms and IoT-based malware, concluding that no algorithm can detect sophisticated and new malware. Signature and heuristic-based methods, however, outperform the others. Similarly, Namanya et al. [87] presented a detailed review of various published works regarding obfuscation techniques. They also reviewed multiple methods to detect malware through recently published works, including heuristics. However, they did not include any future direction for malware detection.

Few studies are partially introducing XAI in the field of malware hunting. Gibert et al. [34] reviewed a comprehensive analysis of malware detection, including challenges and future scope. The study provides quality content on malware taxonomy and background on malware analysis, including static, dynamic, and hybrid methods. Although the authors included and discussed interpretability and the adversarial attack problem in future trends, they did not review the explainable malware model classifiers. In the same way, Majid et al. [78] mainly focused on DL techniques and elaborated on them. However, the study neither discussed the future challenges nor explainable DL models in detail. Limited published studies have assessed the area of cybersecurity using XAI, such as [124], and [44] presented an analysis of cybersecurity methods using XAI.

Although some studies discuss XAI for malware analysis, they do not specifically focus on malware detection, but rather cover other cybersecurity domains related to XAI, such as intrusion detection, spam detection, and malicious traffic detection. For example, [123] is one of the rare review papers in which the authors discuss the interpretability, explainability, and accountability of AI-based malware and intrusion detection models. Srivastava et al. [123] presented a review of cybersecurity and its inherent subdomains, but not specifically on malware analysis. The authors discussed various other topics, e.g., health care, Industrial 4.0, supply chain, e-governance, etc. They give a superficial overview of the application of XAI for malware analysis. Similarly, in [81], the author reviewed articles on XAI for three domains: NLP, bioinformatics, and malware classification. In [48], the authors described and reviewed the application of XAI to build reliability in DL models for digital forensics, but not for malware analysis. For these reasons, we did not include these studies in Table 1.

To the best of our knowledge, no review paper has discussed the challenges and limitations of XAI models in the area of cybersecurity or malware detection. In [61], the authors demonstrated a comprehensive literature survey on cyberattacks and cybersecurity in recent developments and trends, but did not examine the scope of XAI in particular. In [55] and [16], the authors discussed the vulnerability of XAI models to adversarial attacks. These are among the few review papers that cover three major interrelated areas: cybersecurity, XAI, and adversarial attack. However, these studies also did not focus on malware analysis.

Singh et al. [116] partially discussed the problem of adversarial attacks and mainly reviewed articles published using various analysis methods, but they did not discuss the explainable model.

Although many articles have attempted to organize XAI studies, only a few have been successful in doing so. For example, studies by authors such as [35] and [2] covered a wide range of XAI topics, but they did not specifically focus on malware analysis. While these studies provided a general overview of XAI, our work is

significant in expanding the scope of research for both paradigms. Our article highlights the shortcomings of previous works, such as the lack of attention to adversarial attacks, and proposes new evaluation metrics. We also suggest a common evaluation method and communication level for XAI in malware analysis, contributing to the advancement of the field.

Table 1. Summary of published review papers on related topics. 'c' indicates complete explanation, 'p' signifies partial discussion, and 'x' means not discussed at all in the mentioned article. 'MAP' stands for Malware Analysis Process.

Study	Malware Taxonomy	MAP	Feature Discussion	Background on MAP	Analysis Type					Explainable Model	Adversarial Attack	Future Trends /Challenges
					Static	Dynamic	Hybrid	Heuristic	ML/DL			
Arfeen et al. [10]	c	x	c	c	c	c	c	c	c	x	x	p
Aslan et al. [13]	x	p	x	c	c	c	c	c	c	x	x	x
Bhusal et al. [16]	x	x	c	p	x	x	x	x	x	p	c	x
Feizollah et al. [31]	c	c	c	c	c	c	p	x	c	x	x	x
Gibert et al. [34]	c	x	c	c	c	c	c	x	c	x	x	c
Iadarola et al. [44]	p	x	p	p	p	p	p	x	c	c	x	c
Kuppa et al. [55]	x	x	x	p	x	x	x	x	x	c	c	p
Li et al. [61]	x	x	x	c	x	x	x	x	p	c	x	p
Majid et al. [78]	x	x	x	p	c	c	x	x	c	x	x	x
Mathews et al. [81]	p	x	x	p	x	x	x	x	c	c	x	p
Namanya et al. [87]	x	p	p	c	c	c	p	c	p	x	x	x
Razgallah et al. [100]	x	x	x	c	c	c	p	p	c	x	x	x
Saeed et al. [105]	c	x	x	c	p	p	x	c	c	x	x	x
Singh et al. [116]	x	c	x	x	c	c	x	x	c	x	x	p
Souri et al. [122]	p	x	p	c	c	c	p	x	c	x	x	x
Srivastava et al. [123]	c	p	p	p	x	x	x	x	c	c	x	c
Stevens et al. [124]	x	x	x	x	x	x	x	x	p	c	x	x
Wang et al. [132]	x	x	x	p	c	c	c	x	c	x	x	x
Our Survey	c	c	c	c	c	c	c	c	c	c	c	c

3 Malware Analysis and Detection

In this section, we discuss various types of approaches used prior to the introduction of XAI in this domain. These approaches encompass all types of malware analysis, ranging from basic static analysis to ML/DL-based algorithms. We refer to only a few studies in Table 2, not all studies within these categories, as our aim is to emphasize and explore XAI for EMD.

3.1 Traditional approaches for malware analysis

File analysis using a vetting service such as VirusTotal [10] is the first step of malware analysis. If VirusTotal does not recognize the file, it does not necessarily mean the file is benign. Such files are subjected to further

analysis using traditional approaches, which scrutinize deeper aspects that automated tools may miss. If a file is subsequently identified as suspicious or malicious through deeper analysis, it is then added to an internal or specialized malware database. This includes comprehensive steps such as code analysis, where findings at each stage are thoroughly validated. The final step in the malware analysis procedure is to authenticate and catalog a malicious file in this specialized database, ensuring its characteristics can be quickly identified in future scans.

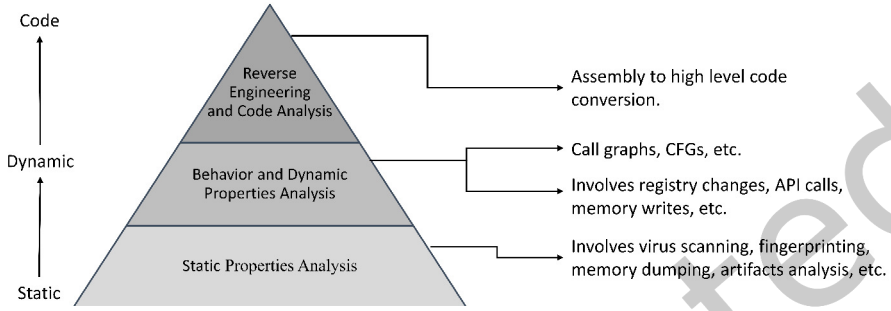


Fig. 1. Stages of Malware Analysis Process. This diagram details the sequential layers of malware analysis, highlighting the transition from static properties analysis through dynamic behavior and dynamic properties analysis, to reverse engineering and code analysis. It illustrates how each layer builds upon the previous to provide a thorough evaluation of potential malware.

The three main stages of malware analysis—static, dynamic, and hybrid—are distinguished by the quality, features (mentioned in Table 2), and complexity of the features. 'Quality' refers to the effectiveness of the features in accurately identifying malware, and 'complexity' involves the computational resources and expertise required to analyze these features. The diagram 1 details the typical flow of malware analysis, starting from initial static analysis through to dynamic and behavioral analysis, integrating both high-level overview and deeper, specific processes. These stages are detailed further below.

3.1.1 Static analysis. Static analysis is one of the fundamental ways to dissect a malware sample. Malware analysts employ a myriad of static features to analyze a known malware sample, as shown in Table 2. Since static features are simple to manipulate and are not robust, static-based malware detection systems can easily be circumvented and exploited by packed and obfuscated malware. Static analysis works well solely to gain an initial indication about a file. If analysts discover unusual indications about a file, they can perform a more thorough investigation.

3.1.2 Dynamic analysis. Dynamic analysis is more reliable than static analysis, although carrying out a thorough dynamic analysis is challenging [93]. Analysts are not only limited to classifying the files, but they can also watch their behavior. They can extract a file's dynamic properties (Table 2) by running it in a sandbox environment and observing their behavior, registry changes, memory alterations, network, and Internet-related activities. Any clue discovered during static analysis might be scrutinized throughout this procedure, because attackers eventually need to modify a dynamic feature, which is not simple to achieve, and evading a dynamic analysis is rather difficult. On the other hand, some files cannot be executed in a safe environment such as in a virtual machine or sandbox, and the detection mechanism may flag files as benign if disguised in specific environments.

The disadvantage of dynamic analysis, other than being computationally expensive, is that malware might hide its malicious behaviour while being analyzed.

Table 2. Studies and their method for malware analysis.

Type	Features	Article(s)	Features used
Static analysis	PE headers features, Import/Export libraries or services, Entropy, Printable-readable strings, Bytes of images, N-grams (from static contexts)	[49, 50, 60]	Op-code/Bytes n-gram analysis
Dynamic analysis	CFGs, API Calls, Call graphs, Memory modifications, Registry changes, Hardware related information, Network traffic, N-grams (from dynamic contexts)	[91, 95, 108, 110, 128]	API calls, Function in DLL import/export, system calls based detection
Hybrid analysis	A combined analysis utilizing both static and dynamic features mentioned above	[114]	Op-code n-grams and API calls based analysis
Heuristics-based analysis	Features based on behavioral characteristics and heuristic rules, e.g., API or system calls	[45, 90]	Diagram of system-call graph, printable string, etc.
ML/DL-based analysis	Utilize either static features, dynamic features, or a combination of both	[65, 86]	Gray scale images classification using CNN
		[29]	API calls analysis using Bayesian network
		[39, 136, 144]	Autoencoder based detection

3.1.3 Hybrid and code analysis. Hybrid analysis combines both static and dynamic analysis of malware. Engineers attempt to comprehend the relationship between the behavior of the files and their features. Additionally, analysts thoroughly examine the assembly code and function to determine how the file will affect the system or organization.

Adopting the hybrid approach is complicated, because attackers constantly create new malware variants and it is difficult to check such a large number of files manually [34]. As a result, the following new paradigms are used to detect malicious activity:

- **Heuristics-based analysis:** This type of malware analysis uses automated processes to extract rules from training files. Although heuristics-based analysis is good at finding zero-day malware, it is prone to false-positives. Identifying a malware sample generally involves dynamic features, such as API calls and CFGs, or static features like strings. Heuristic-based systems are vulnerable against polymorphic, packed, and obfuscated malwares.
- **ML/DL-based analysis:** This type of malware analysis is a more efficient, quick, and accurate technique to evaluate malware than traditional methods. However, ML/DL-based approaches suffer from a lack of interpretability and need a sufficient amount of labeled data. These shortcomings were the main motivation behind introducing XAI for malware analysis, which we will cover in detail in the next sections.

3.2 ML/DL algorithms for malware analysis

3.2.1 Traditional ML. There are many different types of traditional ML models with their strengths and weaknesses. Each ML model has a specific level of interpretability. Linear and logistic regression are among the basic

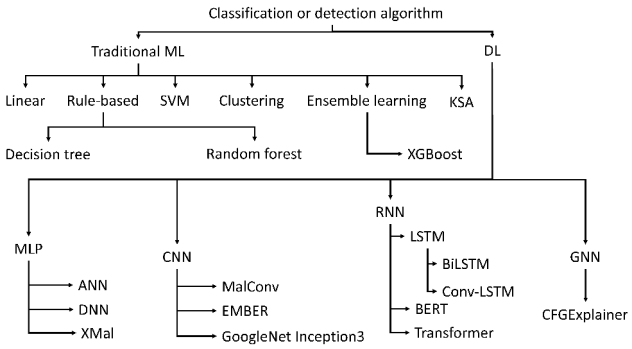


Fig. 2. Algorithms used for malware classification or detection

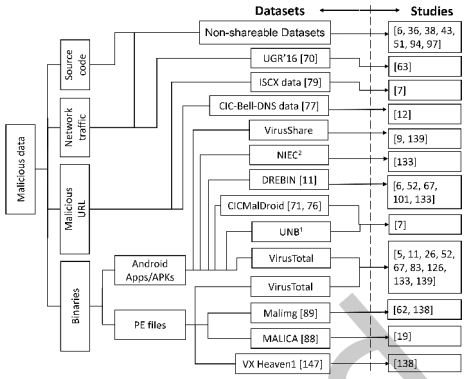


Fig. 3. Malware dataset classification used in the studies taken for the survey

classification algorithms that are straightforward to explain, but they perform poorly when the relationships between dependent and independent variables are non-linear.

In the article by Alzubaidi et al. [8], the authors use a rule-based approach described in the study [2] for intrusion detection. The approach involves generalizing a linear method and uses a rule-based ensemble to construct an explainable model. Similar to this approach, Decision Tree (DT) and Random Forest (RF) algorithms also split data based on feature rules. For example, the ID3 algorithm[2] selects features based on entropy and then classifies the data.

DT has high interpretability, but if we ensemble multiple trees for RF, it becomes more difficult for analysts to study. Studies [5, 7, 83, 84] use RF for classification purposes. In addition to RF, XGBoost has also been used for malicious detection [7, 12].

SVM has frequently been used for malware classification. Studies [51, 67, 83] used SVM in different ways and also provided explainability by using it as a surrogate model. The study [83] employed two versions, i.e., Linear and RBF SVM on the DREBIN dataset [11] for malicious Android app classification. [56] are among the rare studies that use unsupervised learning for classification, where classification is based on Indicators of Compromise (IOC), such as registry keys, file path, command lines, domain names, and IP addresses.

3.2.2 DL. DL-based models have been applied to a wide range of applications, due to their high classification power. A DL model can process different types of data, such as images, time series, and graphs. Therefore, the algorithm we use for a classification problem primarily depends on the kind of data we are processing for detection or classification. Notably, four types of DL algorithms are predominantly employed for malware classification: Multilayer Perceptrons (MLP), which are used for feature-based data; CNN and their customized versions, suitable for image data; RNN, ideal for time series or assembly code analysis; and Graph Neural Networks (GNN), used when processing graph-based data.

Initially, feature-based classification applied on structured data had been frequently used, as it was easy to surrogate for explainable models. Early studies used different versions of MLP [36, 96] that used ANN, and the features were system calls, libraries, and kernels. The studies [38, 139] extracted features such as API systems calls from binaries and used DNN for the classification of malicious files. Study [133] proposed a novel method for interpretable malware detection that used MLP with attention layers to detect the most influencing features. The authors compared their method with other state-of-the-art models at that time.

[2]https://pypi.org/project/decision-tree-id3/

Because most of the studies represented binaries in terms of images, either colored or grey-scale, the authors employed different versions of CNN for image-based classification. [19, 32, 99, 138] used initial versions of CNN and the study [19] used a customized version of CNN called EMBERMalConv. Similar to [133], Bose et al. [19] added an attention mechanism in CNN for extracting the details of weights and gradients of the layers and determining the influencing pixels of the image. The model proposed in the study claimed to be better than the original MalConv [138]. Similarly, Lin et al. [62] used an advanced version of CNN, namely GoogLeNet Inception 3, a CNN architecture with 22 hidden layers. In addition, the study by Mitchell et al. [85] implemented a CNN using opcode data.

There are two types of data that researchers use in RNNs: 1) time series data and 2) assembly code extracted from binaries using a dissembler. The studies [36, 51, 94, 97] used various versions of RNN. For instance, studies [51, 97] detected malicious activities using network traffic data. Prasse et al. [97] implemented their malicious behavior detection method by LSTM and transformers. Khan et al [51] implemented an LSTM-based autoencoder for a similar task. Article [36] presented a state-of-the-art model named LEMNA (Local Explanation Method using Nonlinear Approximation) for explaining cyber threat data. They processed the hex sequence of assembly code using RNN and provide explainability using LEMNA. Furthermore, [94] employed an attention-based RNN that investigates the utilization of registers in each cycle and depicts the gradient of the layers as an explainability of a malicious event. Different transformers used to process assembly code are also popular among researchers. For example, in [59], the authors used a galaxy-based transformer to process the assembly code, creating the embedding of the function, and providing influencing functions as interpretability. Moreover, BERT (Bidirectional Encoder Representations from Transformers) was used in the study [67].

Up to now, Herath et al. [40] is the only study that used CFG for the detection and processing through GNN. In the same model, they also proposed a method to provide explainability through a graph (network of blocks that are responsible for a malicious activity). Additionally, Saqib et al. [107] introduced a new graph model, the Canonical Executable Graph (CEG), which they utilized for malware family detection. Their results, when compared with those obtained using CFG, demonstrated superior performance.

3.3 Datasets of the studies

Data analysts primarily use four types of data for malicious activity detection and classification. Figure 3 depicts the classification of the data types used in the studies and connects them with their specific data. The detail of each data type is as follows:

3.3.1 Binaries. Security analysts rely on binaries as essential data for their investigations, since they encompass all the necessary information for detecting malicious activities. Binary files, identifiable by extensions such as .exe or .bin, consist of a sequence of eight-bit bytes. Disassemblers are required to interpret this type of file and transform it into other required data formats, such as CFGs, code, images, or features. Binary files can contain various file types, such as executables, libraries, images, databases, and archives, among others. In the research studies examined in our proposed review paper, the majority of the binaries used were Android (APK) files or PE files.

Android APPs or APK files An APK file is an Android package that is used to install an Android application on a mobile device. Recent investigations have shown that many hackers use APK files to carry out malicious behavior on a user's mobile phone, such as draining its battery, stealing passwords, or sharing confidential material. Considering that smartphones contain a great deal of a user's confidential information, it is essential to analyze harmful code that spreads through APK files. In this subsection, we review the numerous APK datasets that are publicly available and were employed in the articles we evaluated.

Two very popular public repositories of malicious APK files are VirusShare and VirusTotal. VirusTotal can be used for checking whether an APK is either malicious or benign by using its hashes. For instance, study [11,

121, 133, 139] downloaded malicious APK files from VirusShare. However, for the benign samples, they used the Google Play Store and tested all the samples over VirusTotal. Furthermore, [94] and [83] used datasets from VirusTotal.

Drebin [11], which was created for static analysis of malware running on Android, is another benchmark dataset. The authors [11] gathered binaries from a variety of sources, including the Google Play Store and the Russian and Chinese markets. They also obtained samples from the Android Malware Genome Project [150]. Later, they employed VirusTotal to evaluate each sample and distinguish malware from benign samples by taking the majority votes of the results from ten antivirus scans (AntiVir, AVG, BitDefender, ClamAV, ESET, F-Secure, Kaspersky, McAfee, Panda, and Sophos). Drebin is a huge repository for malware analysis, which is why it has been used by many studies, e.g., [5, 11, 52, 67, 83, 133].

The study [133] explores a new repository of malware, the National Internet Emergency Center (NIEC)[2]. This directory has the latest malware samples and contains various malware categories including Trojans, spyware, and phishing. Similarly, study [7] used a dataset (CICMalDroid 2020 [73, 76]) published by the University of New Brunswick (UNB)[3]. CICMalDroid 2020 is a recently released Android malware dataset consisting of more than 17,341 APK files spanning four categories of adware, banking malware, riskware, and SMS malware. They also have a separate category for benign binaries [73, 76]. Another study by Ambekar et al.[9] utilized two different repositories: Borah et al.[18] and Mathur et al. [82].

PE files The MALICIA dataset is a collection of binaries that have so far exploited 502 servers [88]. The authors of the dataset collected samples of malicious binaries from different servers and provided their metadata [121]. The study [19] used this data for classification purposes. Another dataset is MALIMG [89], which is a collection of malware images from 25 different families. The authors proposed a method to convert binaries into grey-scale images before classifying them. This dataset was used by [62] and [138] to propose an interpretable malware detection model. Other studies that have employed PE binaries have not publicized their datasets, due to non-disclosure agreements. Their dataset consists of a mixture of Android and Windows binaries, or the samples were collected from various sources and verified on VirusTotal.

3.3.2 Source code. Source code is also used by some studies for malware detection or classification. For example, in [94], researchers executed 367 programs on the Xilinx Zynq7000 SoC ZC702 evaluation board. Similarly, in [43], Smali code extracted from source code was converted into an image that was used for further analysis. This demonstrates the potential of source code data in enhancing the accuracy and effectiveness for malware detection and classification.

3.3.3 Network traffic. Four studies in our review used network traffic data for interpretable malware detection. The study [97] collected network traffic data from various companies and different users. The dataset consists of 9,776,911 training samples and 9,970,560 test samples, where each sample is a combination of an organization, a user, and a five-minute interval in which at least one network event was observed. In total, 216 distinct network events occurred at least once in the training and evaluation data—most of these events occurred frequently. On average, 2.69 network events were observed in each five-minute interval in the training data and 2.73 events in the test data. Similarly, Sharma et al. [115] collected network traffic data from various sources, such as MalShare and VirusShare. Another study [51] used a real-world gas pipeline system data source in their conducted experiments, created by Mississippi State University (MSU) [105]. This data source contains time series data with real and synthetic labeled anomaly points. The entire dataset consists of 2,74,628 samples, out of which 2,14,580 are normal data samples, and 6,0048 are anomalous. Through this process, normal and abnormal fragments are created. Unlike the previous two studies, article [63] used publicly available network traffic data, known as UGR'16 [70].

[2]https://share.anva.org.cn/web/publicity/listMalware
[3]https://www.unb.ca/cic/datasets/index.html

This dataset is a collection of about four months of real network traffic from a tier-3 Internet Service Provider (ISP), containing background and attack traffic. It is a well-labeled dataset with enough ground truth attacks. For their specific analysis, the researchers selected a portion of this dataset (i.e., 115 GB) that encompasses the network flows captured within a designated time window.

3.3.4 Malicious URLs. Apart from CICMalDroid 2020 [73, 76], UNB also published other URL-based datasets, such as ISCX-URL2016 [79]. This is a dataset containing benign, spam, phishing, defacement, and malicious URLs. The study [7] utilized this dataset to performed interpretable malicious URL detection. Similarly, in article [12], authors used another dataset (CIC-Bell-DNS 2021) [77] for malicious domain classification. This dataset includes discriminative DNS-based features (e.g., subdomain length, numeric percentage, character distribution, entropy, N-grams, obfuscation method, etc.) that are more robust than the previous studies.

4 XAI-based Malware Analysis

4.1 General approach

In this section, we present a general approach to implement XAI for malware analysis. Fig. 4 depicts the overall process of malware detection/classification using XAI. The pipeline of the process consists of four different components. **1) Classification/Detection:** In this pipeline the data is first split into two parts for the training and testing phases. In the training phase, the data is further divided into training and validation sets. The validation set is employed to fine-tune the classifier parameters and save the best model for the testing phase. These processes are iterated until the specified hyper-parameter epoch is exceeded or the required evaluation criteria are met. **2) Model evaluation:** In the second phase of the analysis, it is crucial to evaluate the performance of the model using various measures commonly used to assess a model's classification or detection performance: precision, recall, accuracy, misclassification score, and F-score. Precision refers to the model's ability to accurately identify true positive samples, while recall measures the model's ability to identify all positive samples. Accuracy denotes the percentage of correct predictions made by the model, while the misclassification score measures the number of incorrectly classified samples. The F-score combines precision and recall to provide a more comprehensive evaluation of the model's performance. These measures are essential for understanding the effectiveness of the model and identifying areas for improvement. **3) Explainability/Interpretability:** Regarding the explainability and interpretability of the classification, if a model's performance is deemed satisfactory during the evaluation phase, the subsequent step involves explaining or interpreting the classification results. While various models have been developed to describe real-world data, the interpretability of malicious data varies depending on the input. The explainability generator interprets the predictions in terms of features, images, and graphs. Two possible ways to generate these explanations include local explanations during the testing phase (Fig. 4) or global explanations that tune the explanation generator's parameters during the training phase. The subsequent section elaborates more on these distinct categories of explanations. **4) Explainability assessment:** After generating explanations, it is crucial to assess their quality. This process is typically qualitative and involves ground truth matching, case studies, and other similar methods. Some researchers have also proposed assessment metrics to quantify the quality of the explanations.

The rest of the article elaborates on the models proposed in the literature for each of these four pipeline components, providing a detailed discussion of their strengths, weaknesses, and applicability in the context of malware analysis.

4.2 Categories of XAI

Based on interpretability, we can divide the XAI models into two groups:

4.2.1 Global Interpretation. Global interpretation analyzes the model as a whole, focusing on the overall structure, parameters, and the representations it has captured [72]. This type of interpretation provides insights into how the entire model behaves across all data points. It examines the contributions of model parameters (weights and biases) to the predictions. Essentially, it depicts the distribution of the predicted outputs concerning the features. For instance, in a neural network trained to classify malware, global interpretation might involve analyzing the importance of different layers and neurons in making predictions. Techniques such as feature importance scores, where the impact of each feature on the overall prediction is quantified, are commonly used for global interpretation. However, achieving global interpretability becomes challenging as the number of parameters increases, especially when dealing with feature spaces that exceed three dimensions, making them difficult to visualize and comprehend.

As an example, suppose we have a deep learning model trained to detect malware based on various static and dynamic features. A global interpretation technique might analyze the overall importance of features such as PE headers, API calls, and network traffic patterns. It could use feature importance scores to show that API calls contribute more significantly to the model's predictions than PE headers. This information helps understand the model's general behavior and the features it relies on most across all instances.

4.2.2 Local Interpretation. Local interpretation, in contrast to global interpretation, focuses on understanding the model's prediction for a single instance. This approach seeks to explain why the model made a specific prediction for a particular input. For example, in the context of malware detection, local interpretation might highlight the specific features of a malware sample (such as certain API calls or byte sequences) that were most influential in the model's decision to classify it as malicious. Techniques such as Local Interpretable Model-agnostic Explanations (LIME) [101] and Shapley Additive Explanations (SHAP) [68] are commonly used for local interpretation. These methods generate explanations by approximating the model locally around the instance of interest, providing insights into which features contributed most to the prediction. For instance, LIME might perturb the input features and observe the changes in the model's output to identify important features, while SHAP values quantify the contribution of each feature based on cooperative game theory.

For local interpretation, consider a specific malware sample that the model has classified as malicious. Using SHAP, we can generate a local explanation that shows which features of this particular sample influenced the model's decision. For instance, the explanation might reveal that certain unusual API calls and specific byte sequences in the file were critical in identifying it as malware. This localized insight helps security analysts understand why the model flagged this specific sample, aiding in further investigation and validation.

4.3 XAI Methods

In this section, we discuss various studies and their explainability methods. Table 3 collectively presents these methods, noting whether they include ground truth and whether their methods have been evaluated.

4.3.1 Rule-based. Rule-based explainability has been inspired by DT. Since DT is a self-interpretable traditional ML algorithm, it does not require further exploration. In a rule-based model, we need to formalize thresholds for features or define some rules that are constructed in the form of trees that make them understandable for people (Figure 6a). However, applying DT to malware analysis is not very convincing and researchers have shifted to black-box DL and further extract certain principles to add explainability.

Researchers typically create DT from a trained neural network and record the output of hidden layers [139] by extracting them as features. Mathematically, the features are defined as follows:

$$c_i j = \frac{1}{n_i} \sum_{n=0}^{N_j - 1} O^i_{h_j} \tag{1}$$

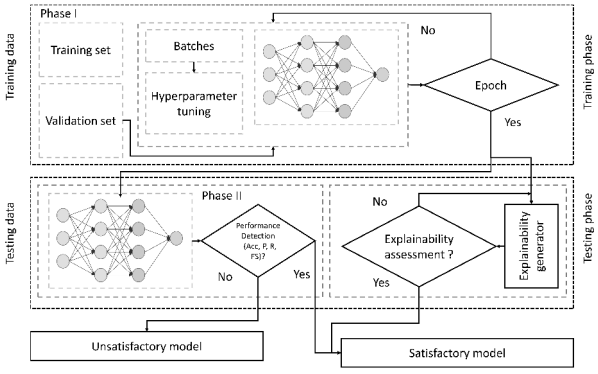


Fig. 4. Illustration of an XAI model for malware analysis, showing key performance metrics: Accuracy (Acc), Precision (P), Recall (R), and F1-Score (FC).

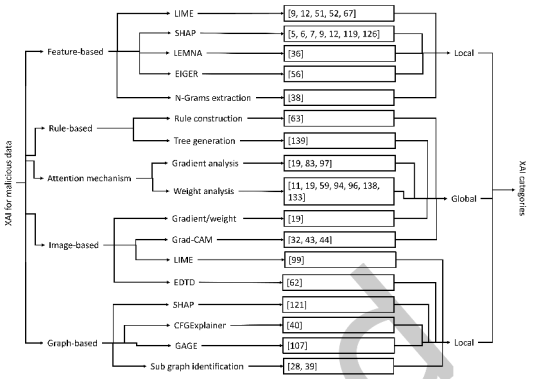


Fig. 5. Explanation methods for malware classification

where c_{ij} is the i^{th} instance and column j of the dataset D', used to construct the explanation DT. In this equation, $O^i_{h_j}$ denotes the output of the j^{th} hidden layer for the i^{th} instance, n_i is the total number of outputs for the i^{th} instance, and N_j represents the total number of neurons in the j^{th} hidden layer. The final result (e.g., benign or malware) of the black-box model is used as a label column in D'. After constructing the data D', a DT model is trained based on the entropy of D' for column c,

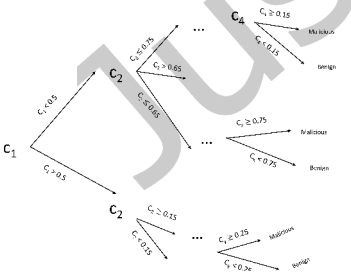
$$Ent(D') = -\sum_{c \epsilon C} p_c log_2(p_c) \tag{2}$$

where p_c is the proportion of class c in the dataset. By splitting D', information gain (IG) can be calculated

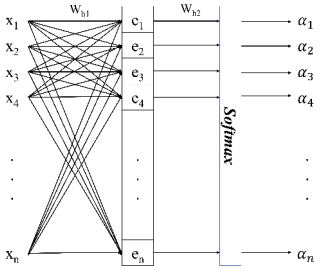
$$IG(D', c = \lambda) = H(D') - H(D'|c = \lambda) \tag{3}$$

where $H(D'|c = \lambda)$ is the entropy for a specific value/class (λ) of any column c in a given sample D'.

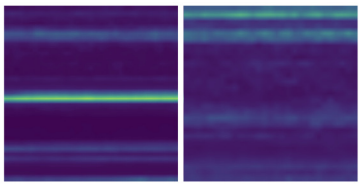
In the last step, a column with maximum IG will be chosen. The explainability of the model looks as Fig 6a.



(a) Rule-based method used for malware classification/detection



(b) Attention mechanism for explainability



(c) Image-based explainability for two different malware families presented in [44]. A sample of the Mecor and Airpush malware families are presented on the left and right.

Fig. 6. XAI methods for EMD

Yan et al. [139] proposed a rule-based explainable model. First, they constructed a rule-based tree from the output layer of the trained neural network and then extracted rules from the input and input layers of the same classifier. Finally, they used values of the output layer as a bridge to join the input-rule-tree and output-rule-tree. The study [63] used another rule-based model that was originally represented by Dash et al. [34]. In [34], authors used Boolean rules either in the form of disjunctive (OR-of-AND) or conjunctive (AND-of-OR) normal forms. Similarly, Sharma et al. [115] used network traffic data to extract network features and employed DT to extract rules for explaining the attack.

4.3.2 Attention mechanism. Most of the XAI malware detection studies used attention-based mechanisms to provide explainability in the model. Attention mechanism can be applied to any type of data and provides explainability according to the input and the model used for classification or detection. For instance, if the input data is an image, this mechanism generates some patterns in the image that may represent a malware family or malicious event. If the input data is a feature, it can be applied to various hidden layers of MLP to help us understand the feed-forwarding mechanism for any set of features.

As shown in Figure 6b, α_i indicates the importance of the corresponding feature, which can be calculated using the Softmax function as,

$$\alpha_i = \frac{exp(e_i^h)}{\sum_{j=0}^{n} exp(e_{ij}^h)}, \tag{4}$$

where n is the total number of the features involved in the classification, and e_i^h is the output of the hidden layer h, which is

$$e_i^h = \sum_{k=0}^{n-1} x_i^k w_{hk}, \tag{5}$$

where n is the total number of the features involved in the classification, and e_i^h is the output of the hidden layer, w_{hk} is a learnable weight of $h^t h$ hidden layer and $k^t h$ feature. The Softmax function is crucial here as it normalizes the output of the hidden layer, converting them into probabilities that sum to one, thereby highlighting the most influential features for the classification process. This normalized scoring by the Softmax function emphasizes the significant features by amplifying the highest scores and suppressing others, facilitating a clear and interpretable visualization of feature importance in the model.

The attention mechanism is the most generic solution for the explainability of any black-box model and any form of data type. For example, in [19], the authors conducted experiments on weights and gradients associated with different layers of MalConv, while predicting the class of PE. This way, they extracted raw bytes, which are more influential for the prediction, and deciphered them as part of the PE. They found that header bytes contribute more than other parts. However, other sections of the binary also show responsibility for the class prediction.

Similar to [19], other studies used ensemble attention mechanisms in CNN, e.g., [138]. Furthermore, article [94] proposed a model in which they integrated CNN with RNN. Some studies also embedded attention layers in MLP and detected the key features of the classification, e.g., [133] and [59]. In [59], researchers inserted an attention layer in FFNN (Feed Forward Neural Network) and detected the main features in the input layer. Also, they utilized this mechanism to detect the most influencing assembly functions. Arp et al. [11] used SVM for the classification and tune some weights to the feature.

Studies [83, 96, 97] employed gradient-based explainability. They analyzed the relevance of each feature in each layer by calculating the gradients and providing explainability. In [97], the authors used integrated gradients and determined the sequence of the most important network events. Similarly, in [83], Melis et al. proposed a

Table 3. Explainable methods types and explaining strategies for studies. 'x' presets the study not evaluated their XAI method.

Method	Ref	XAI Model	Explained by	Ground Truth	XAI Metrics
Attention mechanism-based	[19]	Analysing gradient and weights of layers	Extracting influencer bytes	No	x
	[133]	Attention-based mechanism	Key features	Yes	Interpretability result
	[97]	Integrated gradients method	Determine the sequence of the most important network events	No	x
	[83]	Gradient-based approach	Most influential local features	No	x
	[96]	Layer wise relevance and aggregate	Most influencing system call to the tag classification		Manual evaluation
	[94]	Attention in RNN	Highlight register uses in cycles	No	x
	[138]	Attention-based mechanism	Most influencing instructions	No	x
	[11]	Attention-based weight extraction	Static key features		Manual evaluation
	[59]	Attention-based mechanism	Static features and code embedding		Manual evaluation
rule-based	[139]	Rule-based tree generation	Important features	No	x
	[63]	Boolean rule in disjunctive normal form or conjunctive normal form	Feature based explanation	Yes	x
	[115]	DT	Boolean rules for attack traffics	No	x
Feature based	[36]	LEMNA	Most influencing bytes	Yes	Fidelity test
	[38]	N-grams extraction	Most influencing system calls	No	x
	[51]	LIME	Key features	No	x
	[52]	LIME	Most contributing OpCode sequence	No	x
	[56]	EIGER	IOC detection		Manual evaluation
	[7, 126]	SHAP	Key features	No	x
	[67]	SHAP, LIME	Key features		Fidelity,robustness
	[5, 6, 119]	SHAP	Key features	No	x
	[9, 12]	SHAP, LIME	Key features	No	x
	[85]	H-LIME	Key Opcode features extraction	No	Completeness, sparsity, consistency, efficiency
Image based	[19]	Analysing gradient and weights of layers	Extracting influencing bytes	No	x
	[99]	LIME	Heatmap	No	x
	[44]	Grad-CAM	Heatmap generation, cumulative heatmap, learning evaluation	Yes	x
	[32, 43]	Grad-CAM	Most influencing pixels, heatmap	Yes	Manual evaluation
	[62]	Ensemble Deep Taylor Decomposition (EDTD)	Pixel-level explanation		Fidelity, robustness
Graph-based	[30]	Relate a sub graph to the tactics, techniques, and procedures (TTP)	Subgraph identification	Yes	Manual evaluation
	[121]	SHAP	Subgraph of API call graph	No	x
	[40]	CFGExplainer	Subgraph identification		Sparsity, fidelity
	[107]	GAGE	Subgraph extracted from CEG	Yes	Robustness

model to identify the most influential local features. Study [96] analyzed layer-wise relevance and aggregated the gradient to detect the most influencing system call to the tag classification.

4.3.3 Feature-based. Similar to rule-based explainability, feature-based explainability also detects influencing features for the prediction. However, in this type of explainability, we detect those features by formalizing their importance in the prediction. In this section, we provide a detailed description of these models, including LIME, SHAP, and LEMNA.

LIME is a local surrogate model used for explaining each individual prediction. LIME is trained on the training dataset and for each epoch, it tries to understand how much the output may change on which input. Mathematically, LIME can be explained as follows:

$$\mathcal{E}(f_x) = argmin_{g \epsilon G}\, L(f, g, \pi_x) + \Omega(g), \tag{6}$$

where $\mathcal{E}(f_x)$ is an explanation of model f for instance x, and g, a linear model is fitted by LIME, and x is the instance for making the interpretability. G is the set of all possible explanations and we strive to keep the minimum loss function L for the instance x. The $explanation(f)$ is the function that checks the prediction's closeness with g and the actual model or trained classifier f for instance x. $\Omega(g)$ is the complexity of the model, which we aim to keep as small as possible by controlling the number of parameters used for explainability. The proximity π_x represents the strength of the observation, which is closed to x and used for explainability.

The studies [9, 12, 51, 52, 67] used LIME to identify the main features of the classification. Moreover, Mitchell et al. [85] proposed a novel hierarchical LIME (H-LIME) approach, applied at the levels of classes and methods, resulting in a sparser explanation.

SHAP Similar to LIME, SHAP is a local interpretable model. In this approach, we calculate the contribution of the individual features in the prediction using various possible combinations for all other features. Furthermore, it can be formulated as follows:

$$\phi_j(val) = \Sigma_{S \subseteq (1,\ldots,p)\{j\}} \frac{|S|!(p-|S|-1)!}{p!}(val(S \cup j) - val(S)), \tag{7}$$

where x is the input that we are deciphering, S is the subset of the attributes used in the model, and p is the total number of the models. Furthermore, $val_x(S)$ is the prediction for the instance x for the S subset of the features.

$$val_x(S) = \int \hat{f}(x_1, x_2, ..., x_p) d\mathbb{P}_{x \notin S} - E_X(\hat{f}(X)) \tag{8}$$

SHAP used the Shapley value to detect the contribution of each feature in the prediction for an instance x. It is inspired by Game Theory, where each feature behaves like an individual player of the game and each player is independent to make a decision. Below is the SHAP function:

$$g(z') = \varnothing_0 + \sum_{M}^{j=1} \varnothing_j z_j^{'} \ , \tag{9}$$

where g is the function for interpretability, $\varnothing_j$ is each feature's contribution, which sums up p times, and the total number of features is computed as follows:

$$\sum_{j=1}^{p} \varnothing_j = \hat{f}(x) - E_X(\hat{f}(X)) \ . \tag{10}$$

$\hat{f}(x)$ is the prediction for x. In articles [5–7, 9, 12, 67, 119, 126], researchers employed SHAP for the interpretability of the main features.

Other Some studies proposed an XAI model dedicated to security data. For instance, Guo et al. [36] proposed a model known as LEMNA, customized for security applications. They claimed that it generated high-fidelity results. Although it is a locally interpretable model like LIME, it can handle feature dependency and nonlinear local boundaries to increase explanation fidelity for cybersecurity data. In [56], the authors proposed a model called EIGER that automatically generates IOC. N-Grams extraction of the input features is also employed to explain the underlying model. In [38], the authors extracted N-grams and system calls to explain malware classification.

4.3.4 Image-based. In image-based explainability, researchers either highlight some part of the image (check Figure 6c) or create representing images for each class by employing ensemble, aggregation, or calculating gradients from images of training data. Selvaraju et al. [117] proposed an explainable model, Gradient-weighted Class Activation Mapping (Grad-CAM), for various versions of CNN. It finds a value for each pixel of the image, which is called a class discriminative localization map, $L_c^{Grad-CAM} \epsilon \mathbb{R}^{u \times v}$, with dimensions height(u) and width(v) formulated as follows:

$$L_c^{Grad-CAM} = ReLU(\sum_k w_k^c A_k) \tag{11}$$

where w_k^c is the learnable weight for convolution k and class c, and A_k is the activation of convolution k. w_k^c can be calculated as follows:

$$w_k^c = \overbrace{\frac{1}{Z} \sum_i \sum_j}^{\frown} \frac{\partial y^c}{\partial A_{ij}^k}, \tag{12}$$

where $\frac{\partial y^c}{\partial A_{ij}^k}$ is the gradient of class c score for feature y, with respect to activation A^k, and the first part ($\overbrace{\quad}^{\frown}$) of the Eq. 12 depicts the global average pooling.

The Grad-CAM model was later used by researchers for security applications, e.g., [32, 43, 44]. The studies [43, 44] used CNN for malware classification and explored the portion of images that contributed the most in the classification using Grad-CAM. Moreover, studies [19, 62] fond the most influencing bytes (image pixels) by analyzing the gradient of CNN. Lin et al. [62] proposed an Ensemble Deep Taylor Decomposition (ETDTD) as a method to provide pixel-level explanations for the outputs of a Selective Deep Ensemble Learning-based (SDEL) for image-based malware detection. SDEL is a detector proposed in the same study for image-based malware detection, which combines multiple deep learning models to improve the accuracy. In [138], the model was similar to [19], but the classification was explained by highlighting instructions instead of bytes. Apart from this, Rahman et al. [99] only generated a heatmap.

4.3.5 Graph-based. Recently, graph-based explainability has been employed by security researchers to detect a network of blocks or functions that are malicious, instead of finding just the functions. It represents the interpretability of the malicious events or code in the form of a subgraph. In the studies [30, 40], the authors extracted CFGs from an executable and used them for classification. Later, they employed any subgraph extraction technique and surrogate in their classification method to explain the prediction. In [40], Herath et al. proposed a model namely, CFGExplainer, to extract the blocks of a CFG and used it to explain the behavior of the malware. Moreover, in [30], the authors extracted the most influential API calls from a CFG of a malicious file. Additionally, Soi et al.[121] utilized an API call graph as input to extract APIs for explainability purposes. Building upon these approaches, Saqib et al.[107] developed the CEG and employed a novel method, the Genetic Algorithm based Graph Explainer (GAGE), to identify malicious functions and their caller-callee relationships within CEG. The approach by Saqib et al. [107] demonstrated enhanced robustness and discriminative power compared to CFGExplainer.

4.4 XAI malware analysis model evaluation

Evaluating the performance of an explainable malware detection model requires assessing two key components: discrimination power and interpretability [20]. Discrimination power refers to the model's ability to accurately classify benign and malicious files or identify the specific malware family to which a file belongs. Interpretability, on the other hand, refers to the quality of the explanations provided by the model, including factors such as correctness and robustness.

4.4.1 Discriminating power. The discriminating power of an explainable model is vital and should not be compromised while explaining the prediction. The metrics that are normally used to quantify the classification power of a model include precision, recall, and accuracy, as defined below:

Precision measures the number of right predictions out of the total number of observations of a class. Mathematically, the precision for a model m and class c is:

$$P(m, c) = \frac{\text{No. of correct predictions of c}}{\text{Total observations in c}} \tag{13}$$

Recall is the proportion of correct predictions out of the total number of predictions made for a class c by model m.

$$R(m, c) = \frac{\text{No. of correct predictions of c}}{\text{Total no. of c}} \tag{14}$$

Accuracy is the percentage of correct predictions out of a total number of predictions made by any model m.

$$A(m) = \frac{\text{No. of correct predictions}}{\text{Total no. of observations}} \times 100 \tag{15}$$

F1-Score is the harmonic mean of precision and recall and can be mathematically written for a class c and model m.

$$F1(m) = \frac{2 \times P(m, c) \times R(m, c)}{P(m, c) + R(m, c} \tag{16}$$

The higher the score for the discriminating power, the better the classification model will be.

In this, Section 4.4.1, the discriminating power of various models is compared based on different datasets, as presented in Table 8. Although these comparisons are drawn from diverse datasets, they are instrumental in demonstrating the robustness and adaptability of the models across varying contexts and data characteristics. This approach allows us to identify which models maintain high performance regardless of dataset variability, offering insights into their generalization capabilities. Furthermore, this comparison helps in highlighting specific strengths and weaknesses of each model, facilitating a more informed choice of model based on the dataset's nature and the requirements of the classification task. By evaluating models across different datasets, we can better understand the potential impacts of dataset-specific factors on model performance and thus refine model selection and tuning strategies for optimal results in real-world applications.

4.4.2 Interpretability. Interpretability is an essential evaluation criterion for malware detection models, because it allows for understanding how and why a model is making its decisions. Interpretability is necessary in the context of malware detection, because it identifies specific features or characteristics that the model uses to make its decisions. This information can be used to improve the model by focusing on the most relevant features and to gain insight into the behaviour of the malware itself. Additionally, interpretability can increase trust in the model and its decisions, as users can understand how it arrived at its predictions. This is particularly important in scenarios where the consequences of misclassification can be severe. Overall, interpretability is an essential aspect of designing explainable malware detection models and researchers are using the following metrics to evaluate it:

Interpretability result (IR) is proposed by Wu et al. [133] to evaluate a model generated explanations regarding any malicious file. The model proposed by Wu et al. [133] generates textual information regarding the file. In addition, their ground truth data about being malicious is also an unstructured text. The key word or set of words extracted from both textual explanations are known as 'concept'. This way, we have two sets of 'concepts' (e.g., C_{GT} and C_{MG}):

$$C_{GT} = \{w \mid w \in \text{Word(s) in ground truth explainability}\}$$

$$C_{MG} = \{w \mid w \in \text{Word(s) in model generated explainability}\}$$

So, the numbers of elements in the following subsets are

$$N_d = \left|C_{MG} \bigcap C_{GT}\right| \ , N_s = |C_{MG} - C_{GT}| \ , N_t = \left|C_{MG} \bigcup C_{GT}\right|,$$

where N_d denotes the number of concepts correctly verified with ground truth from model-generated explainability. N_s represents the number of those concepts identified by the model, but not in ground truth. Last, N_t is the number of total concepts in both sets. The above numbers help to determine the following metrics:

$$precision = \frac{N_d}{N_d + N_s}, recall = \frac{N_d}{N_t}, ir = \frac{2 * precision * recall}{precision + recall}$$

where $0 \leq ir \leq 1$. If N_d increases, ir tends to 1. However, for higher N_s values, the value of ir decreases. As a corollary, for high explainability, ir should have a value close to 1.

Fidelity is a more generic and global metric to evaluate the explanations of any XAI model. Fidelity is not only for unstructured textual data, but also for image-, graph-, and feature-based explainability, as studies [36, 40, 62, 67] have used these metrics to evaluate the performance of their models.

Assume we have two sets of explanations, namely model-generated and ground truth:

$$S_i^{GT} = \left\{f_{ij} \mid f_{ij} \in \text{value of } j^{th} \text{ feature in ground truth explainability for } i^{th} \text{ instance}\right\}$$

$$S_i^{MG} = \left\{f_{ij} \mid f_{ij} \in \text{value of } j^{th} \text{ feature in model generated explainability for } i^{th} \text{ instance}\right\}$$

where S_i^{GT} and S_i^{MG} are sets of features from ground-truth and model-generated data, respectively. Moreover, f can represent any kind of feature that is used to perform the classification or detection of malware, e.g., pixel of malicious file's image, static or dynamic features of PE, etc. A viable approach to determine the fidelity of black-box models is by utilizing Mean Absolute Percentage Error (MAPE). Using MAPE, the fidelity of the model m is

$$F(m) = 1 - \left[\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{M}\frac{\left|S_i^{GT}(f_{ij}) - S_i^{MG}(f_{ij})\right|}{\left|S_i^{GT}(f_{ij})\right|}\right], \tag{17}$$

where N and M are the total number of instances and features in the testing data, respectively.

A model with higher fidelity is good and demonstrates how well the model is able to mimic the ground truth explanation.

Robustness shows how much the XAI model can generate diverse explanations for different predictions and data. Therefore, the robustness of the model does not depend on ground truth.

First, we need to define the similarity between two different explanations generated for two different data points by model m,

$$sim(e_i^{c_x}(m), e_j^{c_x}(m)) = \begin{cases} \sum_{k=0}^{M}\left|e_i^{c_x}(f_k) - e_j^{c_x}(f_k)\right| & if f_k \in e_i^{c_x}(m) \cap e_j^{c_x}(m) \\ \\ \sum_{k=0}^{M}\left|e_i^{c_x}(f_k)\right| & if f_{ik} \notin e_i^{c_x}(m) \cap e_j^{c_x}(m) \end{cases} \tag{18}$$

where $e_i^{c_x}(m)$ denotes the explanation generated by model m for an instance i from class c_x and $c_x \in C$. C is a set of all the classes. f_k is the k^{th} feature of the explanation and the total number of features is denoted by M.

For two different classes c_x and c_y from which samples to calculate similarity are selected, if we repeat the experiment $\eta(c_x)$ and $\eta(c_y)$ times, respectively, then the robustness of model m is calculated as:

$$R(m) = \frac{\sum_{\eta(c_x)} sim(e_i^{c_x}(m), e_j^{c_x}(m))/\eta(c_x)}{\sum_{\eta(c_y)} sim(e_i^{c_y}(m), e_j^{c_y}(m))/\eta(c_y)} \tag{19}$$

Robustness represents a ratio between the similarities of different classes. Thus, a robust model m should generate a high value for two different malware families or for benign and malicious files.

Expressiveness describes the expressive power of any explanation generated by model m from a human perspective [62]. High class distinctiveness, i.e., robustness and correctness of an explanation, i.e., fidelity, lead us to understand an explanation better. Thus, the expressiveness of model m is formulated as a factor of both robustness and fidelity [62]:

$$E(m) = F(m)/R(m) \tag{20}$$

Sparsity refers to the ratio between the size of the feature set used to provide an explanation by a model and the size of the ground truth explanation set. It can be computed for a model m as:

$$S(m) = \frac{1}{N}\sum_{i=0}^{N}\left(1 - \frac{\left|S_i^{MG}\right|}{\left|S_i^{GT}\right|}\right), \tag{21}$$

where N is the size of testing data and $|.|$ presents the cardinality of any set. Sparsity has its significance when we evaluate a model with fidelity. A good model should have high sparsity without compromising its fidelity.

Completeness quantifies how much of the decision-making process the explanation covers. For model m, it can be measured as the proportion of decision factors explained:

$$C(m) = \frac{\text{Number of factors explained}}{\text{Total decision factors}}, \tag{22}$$

where a higher $C(m)$ indicates more comprehensive explanations. A high level of completeness ensures that users gain a comprehensive understanding of how and why decisions are made, fostering greater trust and reliability in the system.

Consistency assesses whether similar inputs lead to similar explanations in model m. It can be defined as:

$$K(m) = 1 - \frac{\text{Var}(e(x))}{\text{Var}(x)}, \tag{23}$$

where $e(x)$ is the explanation for input x, and Var represents variability. Higher $K(m)$ implies greater consistency and high consistency ensures that the model's logic is stable and predictable, which is particularly important in high-stakes environments like malware detection.

Efficiency evaluates the computational cost of generating explanations. For model m, it is given by:

$$E(m) = \frac{1}{\text{Time to generate explanations} + \text{Resource usage}}, \tag{24}$$

aiming for higher $E(m)$ to ensure practical usability in real-time systems. A highly efficient model minimizes the overhead of generating explanations, ensuring that the system remains practical even in resource-constrained environments.

5 Discussion and Analysis

5.1 Performance analysis

Table 4. Combined Performance and Interpretability Assessment. Abbreviations used: Acc - Accuracy, P - Precision, R - Recall, FS - F1-Score. A mark 'x' denotes studies not using any metrics to evaluate their XAI methods.

Study	Malicious	Benign	Algorithm	Acc	P	R	FS	Interpretability Assessment
[139]	31,805	10,000	MLP	98.55	97.93	98.27	98.04	x
[19]	700		MalConv	87.10			87.30	x
[19]	700		EMBER Mal	92.20				x
[133]	15,570	20,120	Drebin	95.24	95.94	94.9	95.42	Interpretability result
[133]	15,570	20,120	MLP	96.50	96.38	97.13	96.75	x
[133]	15,570	20,120	XMal	98.35	98.48	98.28	98.37	x
[97]			CNN	92.10				x
[97]			LSTM	92.30				x
[97]			Transformer	94.70				x
[97]			Transformer	48.60				x
[97]			RF	32.20				x
[44]	7386	1060	CNN	97.00				x
[43]			CNN	94.40	94.70	94.30	94.50	Empirical testing
[30]	3250	133743	SIR-GN	89.60			92.70	Empirical testing
[51]	60048	214580	Conv-LSTM-	89.21	93.87	51.43	67.91	x
[52]	5560		CNN	98.00	98.00	98.00	97.00	x
[62]			GoogleNet In	99.87	99.80	99.14	99.50	Fidelity, Robustness, Expressiveness
[83]	5615	121329	SVM	99.00				x
[96]			ANN	94.00	85.00	96.00		Empirical testing
[94]			RNN	98.90				Empirical testing
[7]			XGBoost	95.00				x
[7]			XGBoost	98.00				x
[40]			GNN	77.55				Fidelity, Sparsity
[67]			BERT	99.40				Fidelity, Robustness
[5]			RF				98.60	x
[12]			XGBoost	98.18	97.79	98.57	98.18	x
[11]			Drebin	95.90				x
[59]			IFFNN	97.70	97.50	97.90		Empirical testing
[85]	17,240		CNN	0.9290	0.9371	0.9494	0.9432	Completeness, Sparsity, Stability/Consistency, Efficiency
[126]	50,000	50,000	LR	0.7566	0.8965	0.5802	0.7045	x
[126]	50,000	50,000	DT	0.9720	0.9670	0.9772	0.9721	x
[126]	50,000	50,000	KNN	0.7671	0.7132	0.8933	0.793	x
[99]	9,339		CNN	0.9944	0.9944	0.9944	0.9944	x
[121]	48,372		CNN	0.8700	0.8600	0.8800	0.8700	x
[6]	29,298		XGB	0.9985	0.9985	0.9985	0.9985	x
[119]			Hybrid CNN-BiGRU	0.9798	0.9775	0.9776	0.9775	x
[32]			GradCam	0.9600	0.9500	0.9540	0.9704	x
[9]			TabLSTMNet	0.9763	0.9789	0.9776	0.9800	x
[107]			GCNN, GAGE	0.9000	0.8500	0.8700	0.8700	Robustness

5.1.1 Discriminating power analysis. The performance of the models varies significantly across different studies (Table 4). For example, the model in study [139] had an accuracy of 0.9855, a precision of 0.9793, a recall of 0.9827, and an F1-score of 0.9804. In contrast, the model in study [19] (using the MalConv algorithm) had an accuracy of 0.871 and an F1-score of 0.873. Similarly, the study [133] demonstrated the improvement in performance with increasing complexity in the algorithms used (Drebin, MLP, and XMal), with an accuracy of 0.9524, 0.965, and 0.9835, respectively. Furthermore, study x4 showed the impact of using different architectures such as CNN, LSTM, Transformer (pre-trained), Transformer, and RF on the performance, where the Transformer architecture had the lowest performance with an accuracy of 0.486. It is important to note that the results presented in this table should be interpreted cautiously, as they are highly dependent on the specific dataset and experimental setup used in each study.

Upon reviewing the interpretability assessment alongside the discriminative power, we find that there is no consistent correlation between explainability and discriminative power. For example, studies [43, 133] exhibit high discriminative power coupled with substantial explainability, whereas study[40] demonstrates very low discriminative power.

5.1.2 Interpretability assessment. The interpretability of the models is an essential aspect of evaluating explainable malware detection models. In the literature review, most of the studies have used manual evaluation methods to assess the interpretability of the models. However, there needs to be more standardization and consistency in evaluating interpretability. For example, some studies may focus on the transparency of the model's decision-making process. In contrast, others may concentrate on the interpretability of the features or representations learned by the model.

Despite the lack of standardization in the interpretability evaluation, it is clear that the interpretability of the models is important for ensure that the discrimination results are meaningful and actionable. For example, suppose a model can correctly identify malware, but cannot provide insight into why it made that decision. In that case, it may be difficult for security analysts to use that information to take action. This highlights the importance of developing interpretability evaluation methods that are both standardized and meaningful.

Interpretability is an essential aspect of explainable malware detection. It should be evaluated using numeric values to ensure consistency and standardization. However, most studies reviewed in this article use manual evaluation, which may lead to subjectivity. Balancing performance and interpretability, while designing and evaluating malware detection models, is important (Table 4).

5.2 Specificity to malware analysis

Existing XAI methods, such as LIME and SHAP, are only partially suitable for interpretable malware analysis, due to their limitations in handling high-dimensional and complex data, such as the binary code of a malware sample [36]. These methods are designed to explain the predictions of a model on individual instances. They may need help to comprehensively understand the entire malware detection process. Additionally, these methods are not explicitly designed to handle the unique characteristics of malware, such as evasive tactics and polymorphism, which can make them less effective in explaining the behaviour of a malware sample [111]. Furthermore, the use of such methods for malware analysis has been limited in the literature, and their effectiveness in this domain needs to be thoroughly evaluated. For example, in a study by Guo et al. [36], the authors applied LIME to explain the predictions of a malware detection model and found that the explanations were only sometimes relevant or sufficient to understand the model's decision.

In conclusion, while existing XAI methods may provide some level of interpretability, they may not be appropriate for interpretable malware analysis, due to their following limitations:

- Lack of proper evaluation metrics: The field of interpretable malware analysis is still in its infancy and appropriate evaluation metrics must be used to measure the interpretability of the models. The lack of such

metrics leads to subjectivity in assessing interpretability and makes it difficult to compare different models. For instance, in our survey, many studies claim to provide explainability, yet they have not evaluated their models against XAI metrics. In Table 4, the majority of the studies, denoted by 'x,' do not utilize any evaluation metrics.

- The complexity of malware: Malware, by its very nature, is designed to evade detection and can use various techniques to achieve this. This makes it difficult to understand the underlying behaviour of malware and thus challenging to generate interpretable explanations of the models' predictions. For example, studies [30, 40] exemplify the challenges in generating interpretable model explanations. Study [30] uses CFG generation to analyze how malware's dynamic alteration of execution paths complicates detection and interpretation. Similarly, study [40] examines CFG with node features, demonstrating how malware uses obfuscation techniques to evade detection, further hindering clear interpretation. Both studies illustrate the difficulty in maintaining accuracy in models' explanations due to the sophisticated evasion strategies employed by malware, highlighting a significant gap that necessitates further research in robust, adaptive malware analysis techniques.
- Adversarial attacks: XAI models are vulnerable to adversarial attacks, which can manipulate the models' predictions and the generated explanations. This makes it difficult to trust the interpretability of the models, especially in the context of malware detection, where adversaries may have a vested interest in evading detection. For instance, the studies listed in Table 3 that utilize image-based explainability methods demonstrate increased vulnerability to such attacks [148]. These adversarial strategies can subtly alter image inputs in ways that are imperceptible to human observers but lead to incorrect model outputs, thus misleading the explanation process and undermining trust in the system's decisions.

5.3 Method of communication

In Section 4.2, the method of communication is typically divided into two main categories: local and global explanations. To further refine these categories, we can also categorize them based on the level of abstraction, which determines the comprehensibility level of the explanation to people [71]. However, there is a trade-off between the level of abstraction and the fidelity and faithfulness of the method. Methods with higher abstraction levels are more understandable to humans, but may not accurately reflect the model's behavior.

For local explanations, feature explanation is the lowest level of abstraction, while natural language explanations are the highest level. Feature explanation involves highlighting the input features that have led to a particular output, while natural language explanations use sentences to describe predictions using more abstract concepts. For global explanations, vocabulary explanation is the lowest level of abstraction, while rule explanations are the highest level. Vocabulary explanation explains the entire model in terms of each word in the vocabulary, while rule explanation extracts general rules to explain the model's behavior, although this can be challenging due to the complexity of the rule extraction process.

Choosing the appropriate method of communication for XAI explanations is an ongoing challenge and depends on the specific application and target audience. The most popular methods of communication include, but are not limited to, input features, adversarial examples, influential examples, counterfactuals, natural language, vocabulary, ensemble, linguistic information, and rules [71].

DL-based models for malware analysis should be explainable in terms of features used in manual or traditional methods (see Table 2) so that security administrators and reverse engineers may better comprehend the model's behavior and thought process. Malware identification has previously been made and explained using various techniques, such as feature extraction, API call analysis, subgraph extraction from CFGs, etc. However, some of the explanations do not directly help the stakeholders, who need to understand the proposed method of explanation generation from scratch, which again leads to a blackboxing.

Table 5. Level of studies in the literature

Com. Level	Study	Method Used	Justification
Level 0	[19, 99]	Heat maps	Analysing gradient, weights of layers, and/or pixels which are not directly explainable to stakeholders
	[32, 36, 44]	Grayscale image, influencial bytes	Image generation, heatmap generation, cumulative heatmap, learning evaluation but not applicable to code
	[43]	Smali code to images	Most influencing pixels heatmap
	[62]	NA	Pixel-level explanation
	[63, 115]	Features of attack traffic, boolean rule in disjunctive normal form or conjunctive normal form	Feature-based explanation
Level 1	[139]	Static features (N-grams)	Rule-based tree generation
	[85, 133]	Opcode N-grams	Static key features
	[52]	Opcode sequences to image	Most contributing opcode sequence
	[84]	Static features analysis	Threshold-based rules constructed from features
	[5–7, 9, 11, 12, 51, 67, 83, 119, 126]	Static features	Feature-based explanation
Level 2	[97]	Network traffic	Determine the sequence of the most important network events IOC
	[38, 121]	Most influencing system calls, APIs	Dynamic features analysis
	[30]	CFG generation	Subgraph identification, relate a subgraph to the TTP
	[96]	System calls, system libraries, and kernel	Most influencing system call to the tag classification
	[56]	IOC	Extracts features based on their behavior, IOC detection
	[94]	Register utilization in each cycle	Highlight register uses in cycles
	[59]	Static and dynamic features	Uses opcode frequency and features analysis
Level 3	[138]	Bytes to image	Most influencing instructions
	[40]	CFG with node features	Subgraph identification
	[107]	CEG	Subgraph of malicious functions and their caller-callee

In this article, we propose different levels of explanations categories based on features used in explanation and their relation with manual analysis. They are as follows:

Level 0: This level involves visualizations, some tree construction, or rule generations, which are neither based on static nor dynamic features mentioned in Table 2. This level does not directly explain something to the stakeholders. However, after learning some background about the proposed method, they may learn the pattern and use it in analyses. For example, heatmaps may not directly mean anything to reverse engineers or security administrators, but can provide an overview of the model's behaviour. Suitable for initial assessments by data analysts and entry-level security personnel, these visualizations can help identify patterns or anomalies that merit further investigation.

Level 1: At this level, static indicators provide security administrators with informationabout whether a file is suspicious or not. Key feature extraction based on LIME or SHAP is in this category. Although this level of explanation provides a clue solely at a superficial level, it could be helpful for further dynamic analysis. Additionally, these explanations can guide the configuration of security tools to better detect similar threats in the

future and assist in the initial stages of incident response by outlining the primary characteristics of the potential malware. This level of explanation is instrumental for security administrators and malware analysts who need to rapidly assess the potential threat of a file and preparing the groundwork for more detailed forensic analysis.

Level 2: At this stage, dynamic features determine whether a file is malicious. Network connections, file system activity, API calls, and CFGs are some features that can be retrieved from file behaviour. An explanation based on these could provide substantial understanding to the stakeholders. Therefore, it is on a higher level of communication. This level is crucial for network administrators and cybersecurity incident responders who require a deeper understanding of an active or potential threat's behavior within a network environment. The explanations provided here support proactive threat hunting and incident response strategies.

Level 3: Reverse engineering-related features such as subgraph extraction and code analysis are included at this level. These aspects-based interpretations may help explain the malware's operation and means of evading detection more in depth. This is the highest level of explanation because following this, reverse engineers immediately draw their conclusions and investigate the degree and nature of the danger that might arise from the user only making minor efforts. This level is tailored for expert stakeholders like forensic analysts and advanced security researchers. These professionals benefit from a granular understanding of malware operations and evasion tactics, facilitating a comprehensive threat analysis.

The proposed study suggests a categorization based on four levels of explanation for evaluating models for malware analysis. From Level 0 to Level 3, these levels are arranged in ascending order of interpretation quality. According to their degree of interpretability, we used this framework to group the research we included in our survey (see Table 5) into different categories.

5.4 Time Efficiency

Generally, explainability adds additional computational cost on top of the underlying deep neural network training. Although most systems today are equipped with GPU, we need to improve the time efficiency of the XAI model. Overlapping the training and the explanation part is a way to enhance the system's efficiency and performance. In other words, explanations are extracted at the same time that the model is being trained.

Intrinsic interpretability is a way to integrate interpretability into the models to increase efficiency. One approach is to add a new layer with interpretable constraints to improve the comprehensibility of the classification models globally [72]. Another way is to use an attention weight matrix to specify which parts of the input are attended by the model.

For method of communication discussed in the previous section, generally, higher levels of abstraction in explanations (Levels 2 and 3) involve more complex computations and hence are less time-efficient compared to Levels 0 and 1. However, these higher levels provide richer insights that can be critical for advanced forensic analysis and detailed system audits. Therefore, while they require more computational resources, their potential for providing deep insights justifies the additional time cost.

Time efficiency is a crucial assessment criterion often overlooked in many studies. In our survey, only four studies considered this aspect. Alani et al. [5] reported a testing phase time of 0.7631 microseconds (μs) to extract features for explainability using SHAP. Aslam et al. [12] achieved a time of 0.0424 seconds to extract URL features using XGBoost. However, they achieved a significantly lower time of 0.0078 seconds using DT. Despite the longer extraction time, XGBoost demonstrated better discriminative power than DT. Alani et al. [6] used SHAP and reported a training time of 0.518116 seconds and a testing time of 0.569026 microseconds (μs). Li et al. [59] claimed their method could analyze 15,239 samples per second. These results underscore the importance of including time efficiency in the evaluation of explainable models, as it directly impacts their practicality and scalability in real-world applications. Furthermore, we analyzed their time complexity in Table 6, providing a comprehensive comparison of the computational costs associated with different algorithms.

Table 6. Time complexity analysis of different algorithms for training and testing phases

Algorithm	Time Complexity	Brief Analysis	Phase	Studies
Grad-CAM	$O(n^2 \cdot d)$	Grad-CAM involves a forward pass, gradient calculation, and matrix multiplications, primarily affecting the computational cost with respect to the sequence length n and feature dimension d. The operations scale quadratically with the sequence length and linearly with the feature dimension.	Testing	[32, 43, 44]
SHAP	$O(n \cdot 2^n)$	SHAP calculates the Shapley values for feature importance, with an exponential complexity for exact calculations due to the combinatorial nature of subsets.	Testing	[5–7, 9, 12, 67, 119, 121, 126]
LIME	$O(N \cdot p + n^2 \cdot N)$	LIME generates local explanations by perturbing the input and fitting a simple model to these perturbations. Here, N is the number of perturbed samples, p is the prediction time, and n is the number of features in the local model. The complexity is linear in the number of samples and quadratic in the number of features.	Testing	[9, 51, 52, 67, 99]
Attention-ANN	$O(n^2 \cdot d)$	Attention mechanisms, as used in transformers, compute relevance scores and apply them to input sequences. The dominant cost is in computing the dot-product attention for sequences of length n and feature dimension d, scaling quadratically with the sequence length.	Training and Testing	[11, 59, 94, 133, 138]
DT	$O(n \cdot m \cdot \log(m))$	Constructing a DT involves splitting data based on feature values to minimize impurity. Here, n is the number of features, and m is the number of samples. The complexity reflects the effort to evaluate splits at each node, with a logarithmic factor for tree depth in balanced cases.	Training	[115]
DT	$O(d)$	Once the tree is constructed, making predictions and generating explanations involves traversing the tree from the root to a leaf, where d is the depth of the tree. This is generally fast and efficient.	Testing	[115]
GAGE	$O(G \times P \times E)$	GAGE iteratively refines subgraphs using a genetic algorithm. G represents the number of generations, P is the population size per generation, and E is the fitness evaluation time for subgraphs, influenced by the number of nodes and edges. The process is computationally intensive due to the iterative nature and the complexity of graph operations.	Training	[107]

5.5 Adversarial attack as limitation

There is no doubt that DL speeds up the malware analysis process. In addition, XAI provides a way to verify the detection or classification performed by black box DL models. However, both DL and XAI have limitations and sometimes even interpretability and explainability make it easy for attackers to evade security. Many studies [27, 118, 148] explained how XAI can be manipulated and have proposed a model to uncover the vulnerabilities in XAI models. For example, Zhang et. al [148] demonstrated that interpretable DL systems (IDLSes) are vulnerable to adversarial manipulations, allowing adversaries to arbitrarily designate an input's prediction and interpretation, and suggested potential countermeasures. The paper [27] demonstrated how explanations can be manipulated by applying visually imperceptible perturbations to inputs and proposed mechanisms to enhance the robustness of explanations. Similarly, Slack et. al [118] showed how post hoc explanation techniques, such as LIME and SHAP, can be manipulated by adversarial entities using a novel scaffolding technique, allowing biased classifiers to remain biased, while generating innocuous explanations. Therefore, when dealing with DL and XAI-based models designed for malware analysis, it becomes additionally critical to address the security of AI.

The adversarial attack is still an open issue for the DL-based model, which could make a malware detection system fragile. An adversary could also use explainability to exploit the malware detection model. Thus, in this section, we review two types of papers: 1) studies that discussed the fragility of a DL model for malware detection and 2) studies that used explainability to evade the detection mechanism.

Mathematically, the minimum perturbation added to the x' feature used for classification can affect the classification's direction and result in a misclassification.

$$\begin{aligned} &min \, \|\delta_x\| \\ &s.t. \quad x^{'} = x + \delta_x, \quad f(x^{'}) \neq f(x) \ , \end{aligned} \tag{25}$$

where x is any instance of the dataset, $f(.)$ is the classification model, and δ_x is the perturbation.

5.5.1 Adversarial attacks against DL. DL models can be exploited using various types of data. For instance, in [42, 58, 111], researchers proposed models to manipulate the static and dynamic features of the PE and evade detection. Laskov et al. [58] utilized PE features and automated the process to search the space where they can inject malicious features. In our survey, the majority of studies focus on feature-based explainability; for instance, studies [5, 7, 51, 52, 67] utilize this approach. These models are particularly vulnerable to the types of adversarial attacks discussed earlier, where malicious features are inserted to evade detection.

Other studies [42, 111] proposed models based on API calls and dynamic features analysis. Furthermore, some researchers also used dynamic features as sequential data, e.g., sequence of API calls. It is challenging to create adversarial samples by modifying such data, because one wrong API addition may crash the software. However, some the authors proposed a model to automatically find the space where they can insert benign API calls and mislead the classifier. In the articles [41, 103, 104], authors proposed a model to automatically insert API calls and other printable strings that do not affect the functionality of the executable. The model proposed by Han et al. [38] employs system calls as input, a form of sequential data, making it prone to specific adversarial tactics. Such tactics take advantage of the sequential arrangement by inserting seemingly innocuous API calls in a strategic manner. These inserted calls are designed to deceive classifiers effectively while maintaining the functionality of the software, thereby avoiding system crashes.

The CFG is also a major component to detect a malicious process because it is hard to be distort. However, Abusnaina et al. [1] performed adversarial analysis to generate a subgraph and modify the CFG to evade the system. In addition, some researchers have proposed binary-level modifications. For example, in [26, 98], the authors extract benign prototypes during the training of the neural network and add them to the malicious file. Studies employing CFG, such as those by researchers in [40] and [30], are vulnerable to the adversarial techniques discussed. These techniques involve the generation of subgraphs that subtly alter the CFG to evade detection systems.

5.5.2 Adversarial attacks against XAI. XAI is used to enhance the transparency of DL models and involves humans to verify their classification. However, some attacks have been devised to exploit these models, such as [54, 102]. Rosenberg et al. [102] discussed how XAI could be used to generate adversarial examples for malware classifiers. The paper presented a new approach for generating adversarial examples that focused on modifying specific features of the input, rather than adding new features. The authors first used XAI techniques to identify the most important features of a given malware sample and then conducted a specific modification, feature-by-feature. This approach allowed them to generate adversarial examples that were the most likely to evade detection, while still preserving the functionality of the malware. The paper [102] also introduced the concept of transferability of explainability, which means that the same XAI techniques can be applied to different classifiers and datasets and still result in a similar subset of important features. Overall, this method highlighted how XAI techniques can be used to generate more effective adversarial examples for malware classifiers and how adversaries can leverage these techniques to bypass multi-feature types of malware classifiers. It also raised important questions about the trade-offs between interpretability and robustness in traditional ML models.

In another work, Kuppa et al. [54] proposed a method for exploiting XAI-based models in a black box setting. The authors proposed a taxonomy for XAI methods, covering various security properties and threat models. They

then designed a novel black box attack to analyze the consistency, correctness, and confidence security properties of gradient-based XAI methods. The key idea behind this attack was to use the information provided by the XAI model's explanation report to craft adversarial examples that could fool the model without affecting its output. To conduct the attack, the authors used a gradient-based optimization method to find adversarial examples that maximized the difference between the explanation report and the actual classifier output. After, they evaluated the proposed approach on three security-relevant datasets and models, and demonstrated that the method could mislead both the classifier and explanation report. The results of the study showed that the proposed black box attack is effective in exploiting the XAI models and it can help in designing more secure and robust XAI methods.

Over-revealing malicious features in files for the sake of explainability can lead to the types of attacks discussed previously. For instance, the study by [56] exposes features based on their behavior in IOC detection, potentially informing attackers. Similarly, [40] openly reveals CFG with node features that have malicious intent. Moreover, [96] utilizes system calls and system libraries, which inadvertently disclose the most influential system calls for tag classification to potential attackers.

6 Conclusion

In this review paper, we conducted an in-depth analysis of state-of-the-art techniques in XAI for malware analysis. Our analysis revealed several challenges in creating effective explainable models, including the difficulty in balancing interpretability with accuracy, the absence of a standardized evaluation framework, and the complexity of explaining intricate models.

We observed a need for a generic metric for comparing the quality of explainability, which presents a challenge for malware analysts and reverse engineers. To address this issue, we proposed a framework for comparing the level of explainability that provides insight into how well the model explains malicious file predictions and the depth of knowledge contained in the explanation. Our proposed taxonomy categorized each study considered in this review, along with their justification for falling into that particular category.

We further noted that various articles used different metrics for evaluating their explainability, with some failing to provide sufficient justification for accuracy. We proposed a generic approach for quantifying explainability quality to address this issue. Additionally, we evaluated and compared each study based on its discriminative power.

There are several potential future directions for XAI in malware analysis research. One important area for future research is generating a reliable ground truth dataset that could be used for training and evaluating explainable models. This would help improve the models' reliability and increase their effectiveness. Another important direction is the development of more effective techniques for improving the explainability levels of these models. Our proposed taxonomy provides a road map for increasing the interpretability of models up to level 3. Nonetheless, there is potential to make explanations more understandable to malware analysts and other security stakeholders. Additionally, future work should focus on developing standard evaluation criteria for explainable models. In this article, we generalized some of the metrics used in the literature and proposed some generic metrics. Evaluating models on these metrics is necessary to set benchmark models for the field.

Overall, this review paper contributes to the field of XAI for malware analysis by identifying the challenges in creating effective explainable models, proposing a framework for comparing explainability levels, and offering a taxonomy for categorizing studies. Our proposed approach for quantifying explainability quality and evaluating each study based on its discriminative power can guide researchers and practitioners in developing effective XAI models for malware analysis.

Acknowledgments

This research is supported by BlackBerry Ltd. (ALLRP 561035), Defence Research and Development Canada (contract no. W7701-217332), NSERC Alliance Grants (ALLRP 561035-20), NSERC Discovery Grants (RGPIN-2024-04087), and Canada Research Chairs Program (CRC-2019-00041).

References

[1] Ahmed Abusnaina, Aminollah Khormali, Hisham Alasmary, Jeman Park, Afsah Anwar, and Aziz Mohaisen. 2019. Adversarial learning attacks on graph-based IoT malware detection systems. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1296–1305. https://doi.org/10.1109/ICDCS.2019.00130

[2] A. Adadi and M. Berrada. 2018. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052

[3] W. A. Al-Khater, S. Al-Maadeed, A. A. Ahmed, A. S. Sadiq, and M. K. Khan. 2020. Comprehensive Review of Cybercrime Detection Techniques. *IEEE Access* 8 (2020), 137293–137311. https://doi.org/10.1109/ACCESS.2020.3011259

[4] S. Alam, I. Traore, and I. Sogukpinar. 2015. Annotated Control Flow Graph for Metamorphic Malware Detection. *Comput. J.* 58, 10 (2015), 2608–2621. https://doi.org/10.1093/comjnl/bxu148

[5] Mohammed M Alani and Ali Ismail Awad. 2022. Paired: An explainable lightweight android malware detection system. *IEEE Access* 10 (2022), 73214–73228. https://doi.org/10.1109/ACCESS.2022.3189645

[6] Mohammed M Alani, Atefeh Mashatan, and Ali Miri. 2023. XMal: A lightweight memory-based explainable obfuscated-malware detector. *Computers & Security* 133 (2023), 103409. https://doi.org/10.1016/j.cose.2023.103409

[7] Rafa Alenezi and Simone A Ludwig. 2021. Explainability of Cybersecurity Threats Data Using SHAP. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 01–10. https://doi.org/10.1109/SSCI50451.2021.9659888

[8] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. 2021. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data* 8, 1 (2021), 1–74. https://doi.org/10.1186/s40537-021-00444-8

[9] Namrata Govind Ambekar, N Nandini Devi, Surmila Thokchom, and Yogita. 2024. TabLSTMNet: enhancing android malware classification through integrated attention and explainable AI. *Microsystem Technologies* (2024), 1–19. https://doi.org/10.1007/s00542-024-05615-0

[10] A. Arfeen, Z. A. Khan, R. Uddin, and U. Ahsan. 2022. Toward accurate and intelligent detection of malware. *Concurrency and Computation: Practice and Experience* 34, 4 (2022), e6652. https://doi.org/10.1002/cpe.6652

[11] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. 2014. Drebin: Effective and explainable detection of android malware in your pocket.. In *Ndss*, Vol. 14. 23–26. https://doi.org/10.14722/ndss.2014.23247

[12] Nida Aslam, Irfan Ullah Khan, Samiha Mirza, Alanoud AlOwayed, Fatima M Anis, Reef M Aljuaid, and Reham Baageel. 2022. Interpretable Machine Learning Models for Malicious Domains Detection Using Explainable Artificial Intelligence (XAI). *Sustainability* 14, 12 (2022), 7375. https://doi.org/10.3390/su14127375

[13] Ömer Aslan Aslan and Refik Samet. 2020. A comprehensive review on malware detection approaches. *IEEE Access* 8 (2020), 6249–6271. https://doi.org/10.1109/ACCESS.2019.2963724

[14] J. Bai, Y. Yang, S. Mu, and Y. Ma. 2013. Malware Detection Through Mining Symbol Table of Linux Executables. *Information Technology Journal* 12, 2 (2013), 380–384. https://doi.org/10.3923/itj.2013.380.384

[15] H. Berger, C. Hajaj, E. Mariconti, and A. Dvir. 2022. MaMaDroid2.0 – The Holes of Control Flow Graphs. 2 (2022). https://doi.org/10.48550/arXiv.2202.13922 arXiv:2202.13922

[16] D. Bhusal and N. Rastogi. 2022. *Adversarial Patterns: Building Robust Android Malware Classifiers.* Building Robust Android Malware Classifiers. arXiv, Adversarial Patterns. https://doi.org/10.48550/ARXIV.2203.02121

[17] P. Bhuvaneshwari, A. N. Rao, and Y. H. Robinson. 2021. Spam review detection using self attention based CNN and bi-directional LSTM. *Multimedia Tools and Applications* 80, 12 (2021), 18107–18124. https://doi.org/10.1007/s11042-021-10602-y

[18] Parthajit Borah, DK Bhattacharyya, and JK Kalita. 2020. Malware Dataset Generation and Evaluation. In *2020 IEEE 4th Conference on Information & Communication Technology (CICT)*. IEEE, 1–6. https://doi.org/10.1109/CICT51604.2020.9312053

[19] Shamik Bose, Timothy Barao, and Xiuwen Liu. 2020. Explaining ai for malware detection: Analysis of mechanisms of malconv. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207322

[20] Nicola Capuano, Giuseppe Fenza, Vincenzo Loia, and Claudio Stanzione. 2022. Explainable Artificial Intelligence in CyberSecurity: A Survey. *IEEE Access* 10 (2022), 93575–93600. https://doi.org/10.1109/ACCESS.2022.3204171

[21] Jaime G Carbonell, Ryszard S Michalski, and Tom M Mitchell. 1983. An overview of machine learning. *Machine learning* (1983), 3–23. https://doi.org/10.1016/B978-0-08-051054-5.50005-4

[22] J. Choo and S. Liu. 2018. Visual Analytics for Explainable Deep Learning. *IEEE Computer Graphics and Applications* 38, 4 (2018), 84–92. https://doi.org/10.1109/MCG.2018.042731661

[23] Giovanni Ciaramella, Fabio Martinelli, Francesco Mercaldo, and Antonella Santone. 2023. Exploring Quantum Machine Learning for Explainable Malware Detection. In *2023 International Joint Conference on Neural Networks (IJCNN)*. 1–6. https://doi.org/10.1109/IJCNN54540.2023.10191964

[24] S. H. H. Ding, B. C. M. Fung, and P. Charland. 2019. Asm2Vec: Boosting Static Representation Robustness for Binary Clone Search against Code Obfuscation and Compiler Optimization. 2019 IEEE Symposium on Security and Privacy (SP). 472-489 (2019). https://doi.org/10.1109/SP.2019.00003

[25] Y. Ding, W. Dai, S. Yan, and Y. Zhang. 2014. Control flow-based opcode behavior analysis for Malware detection. *Computers & Security* 44 (2014), 65–74. https://doi.org/10.1016/j.cose.2014.04.003

[26] Yuxin Ding, Miaomiao Shao, Cai Nie, and Kunyang Fu. 2022. An Efficient Method for Generating Adversarial Malware Samples. *Electronics* 11, 1 (2022), 154. https://doi.org/10.3390/electronics11010154

[27] Ann-Kathrin Dombrowski, Maximilian Alber, Christopher J. Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. 2019. Explanations can be manipulated and geometry is to blame. https://doi.org/10.48550/ARXIV.1906.07983

[28] Gianni D'Angelo, Eslam Farsimadan, Massimo Ficco, Francesco Palmieri, and Antonio Robustelli. 2023. Privacy-preserving malware detection in Android-based IoT devices through federated Markov chains. *Future Generation Computer Systems* 148 (2023), 93–105. https://doi.org/10.1016/j.future.2023.05.021

[29] Mojtaba Eskandari, Zeinab Khorshidpour, and Sattar Hashemi. 2013. HDM-Analyser: a hybrid analysis approach based on data mining techniques for malware detection. *J. Comput. Virol.* 9, 2 (may 2013), 77–93. https://doi.org/10.1007/s11416-013-0181-8

[30] Jeffrey Fairbanks, Andres Orbe, Christine Patterson, Janet Layne, Edoardo Serra, and Marion Scheepers. 2021. Identifying ATT&CK Tactics in Android Malware Control Flow Graph Through Graph Representation Learning and Interpretability. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 5602–5608. https://doi.org/10.1109/BigData52589.2021.9671343

[31] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab. 2015. A review on feature selection in mobile malware detection. *Digital Investigation* 13 (2015), 22–37. https://doi.org/10.1016/j.diin.2015.02.001

[32] Premanand Ghadekar, Tejas Adsare, Neeraj Agrawal, Dhananjay Deore, and Tejas Dharmik. 2024. Multi-Class Malware Detection using modified GNN and Explainable AI. In *2024 1st International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU)*. 1–8. https://doi.org/10.1109/IC-CGU58078.2024.10530706

[33] Daniel Gibert, Carles Mateu, and Jordi Planes. 2020. HYDRA: A multimodal deep learning framework for malware classification. *Computers & Security* 95 (2020), 101873. https://doi.org/10.1016/j.cose.2020.101873

[34] D. Gibert, C. Mateu, and J. Planes. 2020. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications* 153, 102526 (2020). https://doi.org/10.1016/j.jnca.2019.102526

[35] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. 2018. A Survey of Methods for Explaining Black Box Models. *Comput. Surveys* 51, 5 (Aug 2018), 42. https://doi.org/10.1145/3236009

[36] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. 2018. Lemna: Explaining deep learning based security applications. In *proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 364–379. https://doi.org/10.1016/j.cose.2021.102198

[37] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo. 2018. A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting. *Future Generation Computer Systems* 85 (2018), 88–96. https://doi.org/10.1016/j.future.2018.03.007

[38] X. Han and B. Olivier. 2020. Interpretable and Adversarially-Resistant Behavioral Malware Signatures. *In Proceedings of the 35th Annual ACM Symposium on Applied Computing. Association for Computing Machinery* 35 (2020), 1668–1677. https://doi.org/10.1145/3341105.3373854

[39] William Hardy, Lingwei Chen, Shifu Hou, Yanfang Ye, and Xin Li. 2016. DL4MD: A deep learning framework for intelligent malware detection. In *Proceedings of the International Conference on Data Science (ICDATA)*. 61. https://api.semanticscholar.org/CorpusID:22913382

[40] Jerome Dinal Herath, Priti Prabhakar Wakodikar, Ping Yang, and Guanhua Yan. 2022. CFGExplainer: Explaining Graph Neural Network-Based Malware Classification from Control Flow Graphs. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 172–184. https://doi.org/10.1109/DSN53405.2022.00028

[41] Weiwei Hu and Ying Tan. 2018. Black-box attacks against RNN based malware detection algorithms. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*. https://doi.org/10.48550/arXiv.1705.08131

[42] Weiwei Hu and Ying Tan. 2022. Generating adversarial malware examples for black-box attacks based on GAN. In *International Conference on Data Mining and Big Data*. Springer, 409–423. https://doi.org/10.48550/ARXIV.1702.05983

[43] Giacomo Iadarola, Rosangela Casolare, Fabio Martinelli, Francesco Mercaldo, Christian Peluso, and Antonella Santone. 2021. A Semi-Automated Explainability-Driven Approach for Malware Analysis through Deep Learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8. https://doi.org/10.1109/IJCNN52387.2021.9533803

[44] G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone. 2021. Towards an interpretable deep learning model for mobile malware detection and family identification. *Computers & Security* 105 (2021), 102198. https://doi.org/10.1016/j.cose.2021.102198

[45] Rafiqul Islam, Ronghua Tian, Lynn M Batten, and Steve Versteeg. 2013. Classification of malware based on integrated static and dynamic features. *Journal of Network and Computer Applications* 36, 2 (2013), 646–656. https://doi.org/10.1016/j.jnca.2012.10.004

[46] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. 1996. Artificial neural networks: A tutorial. *Computer* 29, 3 (1996), 31–44. https://doi.org/10.1109/2.485891

[47] G. Jain, M. Sharma, and B. Agarwal. 2019. Optimizing semantic LSTM for spam detection. *International Journal of Information Technology* 11, 2 (2019), 239–250. https://doi.org/10.1007/s41870-018-0157-5

[48] Aditya K., Slawomir Grzonkowski, and Nhien An Lekhac. 2018. Enabling Trust in Deep Learning Models: A Digital Forensics Case Study. In *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)*. 1250–1255. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00172

[49] Arzu Gorgulu Kakisim, Mert Nar, and Ibrahim Sogukpinar. 2020. Metamorphic malware identification using engine-specific patterns based on co-opcode graphs. *Computer Standards & Interfaces* 71 (2020), 103443. https://doi.org/10.1016/j.csi.2020.103443

[50] Abhishek Karnik, Suchandra Goswami, and Ratan Guha. 2007. Detecting Obfuscated Viruses Using Cosine Similarity Analysis. In *First Asia International Conference on Modelling & Simulation (AMS'07)*. 165–170. https://doi.org/10.1109/AMS.2007.31

[51] Izhar Ahmed Khan, Nour Moustafa, Dechang Pi, Karam M Sallam, Albert Y Zomaya, and Bentian Li. 2021. A New Explainable Deep Learning Framework for Cyber Threat Discovery in Industrial IoT Networks. *IEEE Internet of Things Journal* (2021). https://doi.org/10.1109/JIOT.2021.3130156

[52] Martin Kinkead, Stuart Millar, Niall McLaughlin, and Philip O'Kane. 2021. Towards explainable CNNs for Android malware detection. *Procedia Computer Science* 184 (2021), 959–965. https://doi.org/10.1016/j.procs.2021.03.118

[53] S. Kumar and B. Janet. 2022. DTMIC: Deep transfer learning for malware image classification. *Journal of Information Security and Applications* 64 (2022), 103063. https://doi.org/10.1016/j.jisa.2021.103063

[54] Aditya Kuppa and Nhien-An Le-Khac. 2020. Black box attacks on explainable artificial intelligence (XAI) methods in cyber security. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8. https://doi.org/10.1109/IJCNN48605.2020.9206780

[55] A. Kuppa and N.-A. Le-Khac. 2021. Adversarial XAI Methods in Cybersecurity. *IEEE Transactions on Information Forensics and Security* 16 (2021), 4924–4938. https://doi.org/10.1109/TIFS.2021.3117075

[56] Yuma Kurogome, Yuto Otsuki, Yuhei Kawakoya, Makoto Iwamura, Syogo Hayashi, Tatsuya Mori, and Koushik Sen. 2019. EIGER: Automated IOC Generation for Accurate and Interpretable Endpoint Malware Detection. In *Proceedings of the 35th Annual Computer Security Applications Conference* (San Juan, Puerto Rico, USA) *(ACSAC '19)*. Association for Computing Machinery, New York, NY, USA, 687–701. https://doi.org/10.1145/3359789.3359808

[57] C. Lacave and F. J. Diez. 2000. A Review of Explanation Methods for Bayesian Networks. *Knowledge Engineering Review* 17 (2000), 2002. https://doi.org/10.1017/S026988890200019X

[58] Pavel Laskov et al. 2014. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE symposium on security and privacy*. IEEE, 197–211. https://doi.org/10.1109/SP.2014.20

[59] Miles Q Li, Benjamin CM Fung, Philippe Charland, and Steven HH Ding. 2021. I-MAD: Interpretable malware detector using Galaxy Transformer. *Computers & Security* 108 (2021), 102371. https://doi.org/10.1016/j.cose.2021.102371

[60] Wei-Jen Li, Salvatore Stolfo, Angelos Stavrou, Elli Androulaki, and Angelos D Keromytis. 2007. A study of malcode-bearing documents. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 4th International Conference, DIMVA 2007 Lucerne, Switzerland, July 12-13, 2007 Proceedings 4*. Springer, 231–250. https://doi.org/10.1007/978-3-540-73614-1_14

[61] Y. Li and Q. Liu. 2021. A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. *Energy Reports* 7 (2021), 8176–8186. https://doi.org/10.1016/j.egyr.2021.08.126

[62] Yuzhou Lin and Xiaolin Chang. 2021. Towards Interpretable Ensemble Learning for Image-based Malware Detection. *arXiv preprint arXiv:2101.04889* (2021). https://doi.org/10.48550/arXiv.2101.04889

[63] Hong Liu, Chen Zhong, Awny Alnusair, and Sheikh Rabiul Islam. 2021. FAIXID: a framework for enhancing ai explainability of intrusion detection results using data cleaning techniques. *Journal of Network and Systems Management* 29, 4 (2021), 1–30. https://doi.org/10.1007/s10922-021-09606-8

[64] L. Liu and B. Wang. 2016. Malware classification using gray-scale images and ensemble learning. *3rd International Conference on Systems and Informatics (ICSAI)* 1018-1022 (2016). https://doi.org/10.1109/ICSAI.2016.7811100

[65] Liu Liu, Bao-sheng Wang, Bo Yu, and Qiu-xi Zhong. 2017. Automatic malware classification and new malware detection using machine learning. *Frontiers of Information Technology & Electronic Engineering* 18, 9 (2017), 1336–1347. https://doi.org/10.1631/FITEE.1601325

[66] W. Liu, P. Ren, K. Liu, and H. Duan. 2011. Behavior-Based Malware Analysis and Detection. *2011 First International Workshop on Complexity and Data Mining* 39-42 (2011). https://doi.org/10.1109/IWCDM.2011.17

[67] Zhi Lu and Vrizlynn LL Thing. 2022. "How Does It Detect A Malicious App?" Explaining the Predictions of AI-based Malware Detector. In *2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance*

and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). IEEE, 194–199. https://doi.org/10.1109/BigDataSecurityHPSCIDS54978.2022.00045

[68] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 4768–4777.

[69] Z. Ma, H. Ge, Y. Liu, M. Zhao, and J. Ma. 2019. A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms. *IEEE Access* 7 (2019), 21235–21245. https://doi.org/10.1109/ACCESS.2019.2896003

[70] Gabriel Maciá-Fernández, José Camacho, Roberto Magán-Carrión, Pedro García-Teodoro, and Roberto Therón. 2018. UGR '16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers & Security* 73 (2018), 411–424. https://doi.org/10.1016/j.cose.2017.11.004

[71] Andreas Madsen, Siva Reddy, and Sarath Chandar. 2022. Post-hoc Interpretability for Neural NLP: A Survey. *ACM Comput. Surv.* 55, 8, Article 155 (dec 2022), 42 pages. https://doi.org/10.1145/3546577

[72] Samaneh Mahdavifar. 2021. *Explainable deep learning for detecting cyber threats*. Ph. D. Dissertation. University of New Brunswick. https://unbscholar.lib.unb.ca/handle/1882/14572

[73] Samaneh Mahdavifar, Dima Alhadidi, and Ali A Ghorbani. 2022. Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder. *Journal of network and systems management* 30 (2022), 1–34. https://doi.org/10.1007/s10922-021-09634-4

[74] Samaneh Mahdavifar and Ali A Ghorbani. 2019. Application of deep learning to cybersecurity: A survey. *Neurocomputing* 347 (2019), 149–176. https://doi.org/10.1016/j.neucom.2019.02.056

[75] Samaneh Mahdavifar and Ali A Ghorbani. 2023. CapsRule: Explainable Deep Learning for Classifying Network Attacks. *IEEE Transactions on Neural Networks and Learning Systems* (2023). https://doi.org/10.1109/TNNLS.2023.3262981

[76] Samaneh Mahdavifar, Andi Fitriah Abdul Kadir, Rasool Fatemi, Dima Alhadidi, and Ali A Ghorbani. 2020. Dynamic android malware category classification using semi-supervised deep learning. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, 515–522. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00094

[77] Samaneh Mahdavifar, Nasim Maleki, Arash Habibi Lashkari, Matt Broda, and Amir H Razavi. 2021. Classifying malicious domains using DNS traffic analysis. In *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, 60–67. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech52372.2021.00024

[78] Al-Ani Mustafa Majid, Ahmed Jamal Alshaibi, Evgeny Kostyuchenko, and Alexander Shelupanov. 2023. A review of artificial intelligence based malware detection using deep learning. *Materials Today: Proceedings* 80 (2023), 2678–2683. https://doi.org/10.1016/j.matpr.2021.07.012

[79] Mohammad Saiful Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkari, Natalia Stakhanova, and Ali A Ghorbani. 2016. Detecting malicious urls using lexical analysis. In *Network and System Security: 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings 10*. Springer, 467–482. https://doi.org/10.1007/978-3-319-46298-130

[80] D. Martens, B. Baesens, T. Van Gestel, and J. Vanthienen. 2007. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research* 183, 3 (2007), 1466–1476. https://doi.org/10.1016/j.ejor.2006.04.051

[81] S. M. Mathews. 2019. Explainable Artificial Intelligence Applications in NLP, Biomedical, and Malware Classification: A Literature Review. In *Intelligent Computing*, R. Bhatia Arai and S. Kapoor (Eds.). International Publishing, Springer, 1269–1292. https://doi.org/10.1007/978-3-030-22868-2_90

[82] Akshay Mathur, Laxmi Mounika Podila, Keyur Kulkarni, Quamar Niyaz, and Ahmad Y Javaid. 2021. NATICUSdroid: A malware detection framework for Android using native and custom permissions. *Journal of Information Security and Applications* 58 (2021), 102696. https://doi.org/10.1016/j.jisa.2020.102696

[83] M. Melis, D. Maiorca, B. Biggio, G. Giacinto, and F. Roli. 2018. Explaining Black-box Android Malware Detection. *26th European Signal Processing Conference (EUSIPCO)* 524 (2018), 524–528. https://doi.org/10.23919/EUSIPCO.2018.8553598

[84] Alan Mills, Theodoros Spyridopoulos, and Phil Legg. 2019. Efficient and interpretable real-time malware detection using random-forest. In *2019 International conference on cyber situational awareness, data analytics and assessment (Cyber SA)*. IEEE, 1–8. https://doi.org/10.1109/CyberSA.2019.8899533

[85] Jeff Mitchell, Niall McLaughlin, and Jesus Martinez-del Rincon. 2024. Generating sparse explanations for malicious Android opcode sequences using hierarchical LIME. *Computers & Security* 137 (2024), 103637. https://doi.org/10.1016/j.cose.2023.103637

[86] Hamad Naeem, Bing Guo, Muhammad Rashid Naeem, and Danish Vasan. 2019. Visual malware classification using local and global malicious pattern. *Journal of Computers* 6 (2019), 73–83. https://doi.org/10.3966/199115992019123006006

[87] A. P. Namanya, A. Cullen, I. U. Awan, and J. P. Disso. 2018. The World of Malware: An Overview. *IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)* (2018), 420–427. https://doi.org/10.1109/FiCloud.2018.00067

[88] Antonio Nappa, M Zubair Rafique, and Juan Caballero. 2015. The MALICIA dataset: identification and analysis of drive-by download operations. *International Journal of Information Security* 14, 1 (2015), 15–33. https://doi.org/10.1007/s10207-014-0248-7

[89] Lakshmanan Nataraj, Sreejith Karthikeyan, Gregoire Jacob, and Bangalore S Manjunath. 2011. Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security*. 1–7. https://doi.org/10.1145/2016904.2016908

[90] Smita Naval, Vijay Laxmi, Muttukrishnan Rajarajan, Manoj Singh Gaur, and Mauro Conti. 2015. Employing Program Semantics for Malware Detection. *IEEE Transactions on Information Forensics and Security* 10, 12 (2015), 2591–2604. https://doi.org/10.1109/TIFS.2015.2469253

[91] Stavros D Nikolopoulos and Iosif Polenakis. 2017. A graph-based model for malware detection and classification using system-call groups. *Journal of Computer Virology and Hacking Techniques* 13, 1 (2017), 29–46. https://doi.org/10.1007/s11416-016-0267-1

[92] S. Niksefat, P. Kaghazgaran, and B. Sadeghiyan. 2017. Privacy issues in intrusion detection systems: A taxonomy, survey and future directions. *Computer Science Review* 25 (2017), 69–78. https://doi.org/10.1016/j.cosrev.2017.07.001

[93] Ori Or-Meir, Nir Nissim, Yuval Elovici, and Lior Rokach. 2019. Dynamic malware analysis in the modern era—A state of the art survey. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 1–48. https://doi.org/10.1145/3329786

[94] Zhixin Pan, Jennifer Sheldon, and Prabhat Mishra. 2020. Hardware-Assisted Malware Detection using Explainable Machine Learning. In *2020 IEEE 38th International Conference on Computer Design (ICCD)*. 663–666. https://doi.org/10.1109/ICCD50377.2020.00113

[95] Younghee Park and Douglas Reeves. 2011. Deriving common malware behavior through graph clustering. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. 497–502. https://doi.org/10.1016/j.cose.2013.09.006

[96] Lukas Pirch, Alexander Warnecke, Christian Wressnegger, and Konrad Rieck. 2021. Tagvet: Vetting malware tags using explainable machine learning. In *Proceedings of the 14th European Workshop on Systems Security*. 34–40. https://doi.org/10.1145/3447852.3458719

[97] Paul Prasse, Jan Brabec, Jan Kohout, Martin Kopp, Lukas Bajer, and Tobias Scheffer. 2021. Learning explainable representations of malware behavior. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 53–68. https://doi.org/10.1007/978-3-030-86514-6_4

[98] Yanchen Qiao, Weizhe Zhang, Zhicheng Tian, Laurence T Yang, Yang Liu, and Mamoun Alazab. 2022. Adversarial Malware Sample Generation Method Based on the Prototype of Deep Learning Detector. *Computers & Security* (2022), 102762. https://doi.org/10.1016/j.cose.2022.102762

[99] Mohammad Muhibur Rahman, Anushua Ahmed, Mutasim Husain Khan, Mohammad Rakibul Hasan Mahin, Fahmid Bin Kibria, Dewan Ziaul Karim, and Mohammad Kaykobad. 2023. CNN vs Transformer Variants: Malware Classification Using Binary Malware Images. In *2023 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*. IEEE, 308–315. https://doi.org/10.1109/COMNETSAT59769.2023.10420585

[100] Asma Razgallah, Raphaël Khoury, Sylvain Hallé, and Kobra Khanmohammadi. 2021. A survey of malware detection in Android apps: Recommendations and perspectives for future research. *Computer Science Review* 39 (2021), 100358. https://doi.org/10.1016/j.cosrev.2020.100358

[101] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

[102] Ishai Rosenberg, Shai Meir, Jonathan Berrebi, Ilay Gordon, Guillaume Sicard, and Eli Omid David. 2020. Generating end-to-end adversarial examples for malware classifiers using explainability. In *2020 international joint conference on neural networks (IJCNN)*. IEEE, 1–10. https://doi.org/10.1109/IJCNN48605.2020.9207168

[103] Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. 2020. Query-efficient black-box attack against sequence-based malware classifiers. In *Annual Computer Security Applications Conference*. 611–626. https://doi.org/10.1145/3427228.3427230

[104] Ishai Rosenberg, Asaf Shabtai, Lior Rokach, and Yuval Elovici. 2018. Generic black-box end-to-end attack against state of the art API call based malware classifiers. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 490–510. https://doi.org/10.1007/978-3-030-00470-5_23

[105] I. A. Saeed, A. Selamat, and A. M. A. Abuagoub. 2013. A Survey on Malware and Malware Detection Systems. *International Journal of Computer Applications* 67, 16 (2013), 25–31. https://doi.org/10.5120/11480-7108

[106] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. 2019. *Explainable AI: interpreting, explaining and visualizing deep learning*. Vol. 11700. Springer Nature.

[107] M. Saqib, B. C. M. Fung, P. Charland, and A. Walenstein. 2024. GAGE: Genetic Algorithm-based Graph Explainer for Malware Analysis. In *Proc. of the 40th IEEE International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Utrecht, Netherlands, 2258–2270.

[108] V Sai Sathyanarayan, Pankaj Kohli, and Bezawada Bruhadeshwar. 2008. Signature generation and detection of malware families. In *Information Security and Privacy: 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008. Proceedings 13*. Springer, 336–349. https://doi.org/10.1007/978-3-540-70500-0_25

[109] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

[110] M.G. Schultz, E. Eskin, F. Zadok, and S.J. Stolfo. 2001. Data mining methods for detection of new malicious executables. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*. 38–49. https://doi.org/10.1109/SECPRI.2001.924286

[111] Ali Shafiei, Vera Rimmer, Ilias Tsingenopoulos, Lieven Desmet, and Wouter Joosen. 2022. Position Paper: On Advancing Adversarial Malware Generation Using Dynamic Features. In *Proceedings of the 1st Workshop on Robust Malware Analysis* (Nagasaki, Japan) *(WoRMA '22)*. Association for Computing Machinery, New York, NY, USA, 15–20. https://doi.org/10.1145/3494110.3528244

[112] F. Shahzad and M. Farooq. 2012. ELF-Miner: using structural knowledge and data mining methods to detect new (Linux) malicious executables. *Knowledge and Information Systems* 30, 3 (2012), 589–612. https://doi.org/10.1007/s10115-011-0393-5

[113] Larissa Shamseer, David Moher, Mike Clarke, Davina Ghersi, Alessandro Liberati, Mark Petticrew, Paul Shekelle, and Lesley A Stewart. 2015. Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015: elaboration and explanation. *Bmj* 349 (2015). https://doi.org/10.1136/bmj.g7647

[114] Madhu K Shankarapani, Subbu Ramamoorthy, Ram S Movva, and Srinivas Mukkamala. 2011. Malware detection using assembly and API call sequences. *Journal in computer virology* 7 (2011), 107–119. https://doi.org/10.1007/s11416-010-0141-5

[115] Yashovardhan Sharma, Simon Birnbach, and Ivan Martinovic. 2023. RADAR: A TTP-based Extensible, Explainable, and Effective System for Network Traffic Analysis and Malware Detection. In *Proceedings of the 2023 European Interdisciplinary Cybersecurity Conference* (Stavanger, Norway) *(EICC '23)*. Association for Computing Machinery, New York, NY, USA, 159–166. https://doi.org/10.1145/3590777.3590804

[116] J. Singh and J. Singh. 2021. A survey on machine learning-based malware detection in executable files. *Journal of Systems Architecture* 112 (2021), 101861. https://doi.org/10.1016/j.sysarc.2020.101861

[117] Jagsir Singh and Jaswinder Singh. 2021. A survey on machine learning-based malware detection in executable files. *Journal of Systems Architecture* 112 (2021), 101861. https://doi.org/j.sysarc.2020.101861

[118] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. 2020. Fooling LIME and SHAP: Adversarial Attacks on Post Hoc Explanation Methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society* (New York, NY, USA) *(AIES '20)*. Association for Computing Machinery, New York, NY, USA, 180–186. https://doi.org/10.1145/3375627.3375830

[119] Santosh K Smmarwar, Govind P Gupta, and Sanjay Kumar. 2023. XAI-AMD-DL: An Explainable AI Approach for Android Malware Detection System Using Deep Learning. In *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*. IEEE, 423–428. https://doi.org/10.1109/AIC57670.2023.10263974

[120] S. M. Sohi, J.-P. Seifert, and F. Ganji. 2021. RNNIDS: Enhancing network intrusion detection systems through deep learning. *Computers & Security* 102 (2021), 102151. https://doi.org/10.1016/j.cose.2020.102151

[121] Diego Soi, Alessandro Sanna, Davide Maiorca, and Giorgio Giacinto. 2024. Enhancing android malware detection explainability through function call graph APIs. *Journal of Information Security and Applications* 80 (2024), 103691. https://doi.org/10.1016/j.jisa.2023.103691

[122] A. Souri and R. Hosseini. 2018. A state-of-the-art survey of malware detection approaches using data mining techniques. *Human-Centric Computing and Information Sciences* 8, 1 (2018), 3. https://doi.org/10.1186/s13673-018-0125-x

[123] G. Srivastava, R. H. Jhaveri, S. Bhattacharya, S. Pandya, Maddikunta Rajeswari, P. K. R., G. Yenduri, J. G. Hall, M. Alazab, and T. R. Gadekallu. 2022. *XAI for Cybersecurity: State of the Art, Challenges, Open Issues and Future Directions.* https://doi.org/10.48550/ARXIV.2206.03585

[124] T. Stevens. 2020. Knowledge in the grey zone: AI and cybersecurity. *Digital War* 1, 1 (2020), 164–170. https://doi.org/10.1057/s42984-020-00007-w

[125] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai. 2018. Lightweight Classification of IoT Malware Based on Image Recognition. *IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)* 2 (2018), 664–669. https://doi.org/10.1109/COMPSAC.2018.10315

[126] Trong-Nghia To, Hien Do Hoang, Phan The Duy, and Van-Hau Pham. 2023. MalDEX: An Explainable Malware Detection System Based on Ensemble Learning. In *2023 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*. 1–6. https://doi.org/10.1109/MAPR59823.2023.10288922

[127] K. Vredenburgh. 2021. *The Right to Explanation*. Vol. 30. Wiley Online Library. 209–229 pages. https://doi.org/10.1111/jopp.12262

[128] Gérard Wagener, Radu State, and Alexandre Dulaunoy. 2008. Malware behaviour analysis. *Journal in computer virology* 4 (2008), 279–287. https://doi.org/10.1007/s11416-007-0074-9

[129] Hua Wang, Cuiqin Ma, and Lijuan Zhou. 2009. A brief review of machine learning and its application. In *2009 international conference on information engineering and computer science*. IEEE, 1–4. https://doi.org/10.1109/ICIECS.2009.5362936

[130] Q. Wang, H. Yang, G. Wu, K.-K. R. Choo, Z. Zhang, G. Miao, and Y. Ren. 2022. Black-box adversarial attacks on XSS attack detection model. *Computers & Security* 113 (2022), 4102554. https://doi.org/10.1016/j.cose.2021.102554

[131] Zihao Wang, K. W. Fok, and V. L. L. Thing. 2022. Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study. *Computers & Security* 113 (2022), 102542. https://doi.org/10.1016/j.cose.2021.102542

[132] Zhiqiang Wang, Q. Liu, and Y. Chi. 2020. Review of Android Malware Detection Based on Deep Learning. *IEEE Access* 8 (2020), 181102–181126. https://doi.org/10.1109/ACCESS.2020.3028370

[133] Bozhi Wu, Sen Chen, Cuiyun Gao, Lingling Fan, Yang Liu, Weiping Wen, and Michael R Lyu. 2021. Why an android app is classified as malware: Toward malware classification interpretation. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 2 (2021), 1–29. https://doi.org/10.1145/3423096

[134] Tobias Wüchner, Martín Ochoa, and Alexander Pretschner. 2015. Robust and effective malware detection through quantitative data flow graph metrics. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 12th International Conference, DIMVA 2015, Milan, Italy, July 9-10, 2015, Proceedings 12*. Springer, 98–118. https://doi.org/10.1007/978-3-319-20550-2_6

[135] T. Wüchner, M. Ochoa, and A. Pretschner. 2014. Malware Detection with Quantitative Data Flow Graphs. In *Proceedings of the 9th ACM Symposium on Information*. Computer and Communications Security, 271–282. https://doi.org/10.1145/2590296.2590319 https://doi.org/10.1145/2590296.2590319.

[136] Fei Xiao, Zhaowen Lin, Yi Sun, and Yan Ma. 2019. Malware detection based on deep learning of behavior graphs. *Mathematical Problems in Engineering* 2019, 1 (2019), 8195395. https://doi.org/10.1155/2019/8195395

[137] G. Xiao, J. Li, Y. Chen, and K. Li. 2020. MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks. *J. Parallel Distributed Comput.* 141 (2020), 49–58. https://doi.org/10.1016/j.jpdc.2020.03.012

[138] Hiromu Yakura, Shinnosuke Shinozaki, Reon Nishimura, Yoshihiro Oyama, and Jun Sakuma. 2017. Malware Analysis of Imaged Binary Samples by Convolutional Neural Network with Attention Mechanism. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (Dallas, Texas, USA) *(AISec '17)*. Association for Computing Machinery, New York, NY, USA, 55–56. https://doi.org/10.1145/3128572.3140457

[139] Anli Yan, Zhenxiang Chen, Haibo Zhang, Lizhi Peng, Qiben Yan, Muhammad Umair Hassan, Chuan Zhao, and Bo Yang. 2021. Effective detection of mobile malware behavior based on explainable deep neural network. *Neurocomputing* 453 (2021), 482–492. https://doi.org/10.1016/j.neucom.2020.09.082

[140] J. Yang, T. Li, G. Liang, Y. Wang, T. Gao, and F. Zhu. 2020. Spam transaction attack detection model based on GRU and WGAN-div. *Computer Communications* 161 (2020), 172–182. https://doi.org/10.1016/j.comcom.2020.07.031

[141] Wei Yang, Deguang Kong, Tao Xie, and Carl A. Gunter. 2017. Malware Detection in Adversarial Settings: Exploiting Feature Evolutions and Confusions in Android Apps. In *Proceedings of the 33rd Annual Computer Security Applications Conference* (Orlando, FL, USA) *(ACSAC '17)*. Association for Computing Machinery, New York, NY, USA, 288–302. https://doi.org/10.1145/3134600.3134642

[142] Xin Yao. 1993. A review of evolutionary artificial neural networks. *International journal of intelligent systems* 8, 4 (1993), 539–567. https://doi.org/10.1007/s10462-011-9270-6

[143] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing* (2024), 100211. https://doi.org/10.1016/j.hcc.2024.100211

[144] Yanfang Ye, Lingwei Chen, Shifu Hou, William Hardy, and Xin Li. 2018. DeepAM: a heterogeneous deep learning framework for intelligent malware detection. *Knowledge and Information Systems* 54 (2018), 265–285. https://doi.org/10.1007/s10115-017-1058-9

[145] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula. 2017. Autoencoder-based feature learning for cyber security applications. *International Joint Conference on Neural Networks (IJCNN)* (2017), 3854–3861. https://doi.org/10.1109/IJCNN.2017.7966342

[146] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu. 2021. A survey of visual analytics techniques for machine learning. *Computational Visual Media* 7, 1 (2021), 3–36. https://doi.org/10.1007/s41095-020-0191-7

[147] H. Zhang, L. Huang, C. Q. Wu, and Z. Li. 2020. An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset. *Computer Networks* 177 (2020), 107315. https://doi.org/10.1016/j.comnet.2020.107315

[148] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. 2020. Interpretable Deep Learning under Fire. In *Proceedings of the 29th USENIX Conference on Security Symposium (SEC'20)*. USENIX Association, USA, Article 94, 18 pages.

[149] M. Zheng, M. Sun, and J. C. S. Lui. 2013. Droid Analytics: A Signature Based Analytic System to Collect, Extract, Analyze and Associate Android Malware. In *12th IEEE International Conference on Trust, Security and Privacy in Computing and*. Communications, 163–171. https://doi.org/10.1109/TrustCom.2013.25

[150] Yajin Zhou and Xuxian Jiang. 2012. Dissecting android malware: Characterization and evolution. In *2012 IEEE symposium on security and privacy*. IEEE, 95–109. https://doi.org/10.1109/SP.2012.16

[151] D. Zhu, H. Jin, Y. Yang, D. Wu, and W. Chen. 2017. DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data. *IEEE Symposium on Computers and Communications (ISCC)* (2017), 438–443. https://doi.org/10.1109/ISCC.2017.8024568

Appendix

A. Background

A1. Rise of AI

With an incalculable amount of data gathered daily, it is not feasible to analyze and correlate them using only the intervention of a human agent. To automate and systematically analyze and explore big data, researchers have started predicting and classifying data using statistical and mathematical concepts known as traditional ML models. Traditional ML models include, but are not limited to linear regression, logistic regression, polynomial regression, DT, RF, SVM, and K-mean clustering. For instance, the foundational concepts of machine learning have been extensively discussed by Carbonell et al. [21], while the practical applications of ANNs in data classification are detailed by Jain et al. [46]. Further, Wang et al. [129] provide a brief review of ML-applications across various fields such as medicine, agriculture, and environmental science, demonstrating the versatility of ML techniques. Finally, the development and applications of evolutionary ANNs are reviewed by Yao [142], illustrating their role in enhancing the capabilities of traditional neural networks. However, traditional ML methods are not efficient in solving complex problems, especially when the decision boundary is highly non-linear. Consequently, DL algorithms [8, 109], inspired by the working of the human brain, came into the picture. The deep network architectures in the DL models can extract high-level representations of the input data using several non-linear complex layers.

A2. Black-boxing in AI

AI has revolutionized our life by offering effective and efficient traditional ML and DL-based algorithms that mimic what humans can think and do. Although these models can achieve human performance in a wide range of applications, they are unable to explain their output results in a human-understandable way. They can classify inputs into different categories, but cannot explain why a particular decision was made. Each AI-based model can provide a different level of explainability. For example, DT are to some extent inherently explainable, because of the rules they generate during the classification process, whereas SVM's predictions are is too complicated to be understand among all ML models. ANN improved the performance of every AI model, but due to their inherent robustness and complexity, they almost provide the least amount of reliability, interpretability, and transparency [2] (check Figure 7).

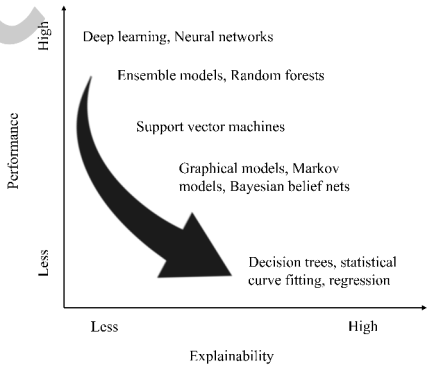


Fig. 7. Relation between performance and explainability of ML/DL models.

A3. 'Right to Explanation' and XAI

'Right to Explanation' (RTE) was the primary motivation for explainable AI models [127]. According to this law, any group or individual has the right to know the explanation behind every personal, legal, or commercial decision made by any professional or legal executive [127], e.g., rejection of loan application, health insurance coverage, etc. Because any prediction or decision automatically made using ANN has only a result without an explanation of how that result was obtained, DL methods are in violation of the law. For example, in digital forensics, a false prediction/classification may lead us to the wrong criminal. Therefore, a model needs to be transparent in order to be able to rely on it for automated forensics results.

In the case of malware analysis, DL-based models can be a useful. However, adopting these models could be problematic if the model's decisions are not explainable to the involved stakeholders. Security administration may apply RTE because a false positive malware detection could result in unwanted system disruptions and downtime, and a false negative could leave the system vulnerable to an attack, which may put the organization in jeopardy. By providing explanations for the decisions made by the model, analysts can better understand the reasoning behind the results and improve the accuracy and effectiveness of the malware analysis. Furthermore, security analysts can identify the causes of the incidents to help them in mitigating the risk and adjusting the security policies of the organization accordingly [72].

B. Methodology and article selection

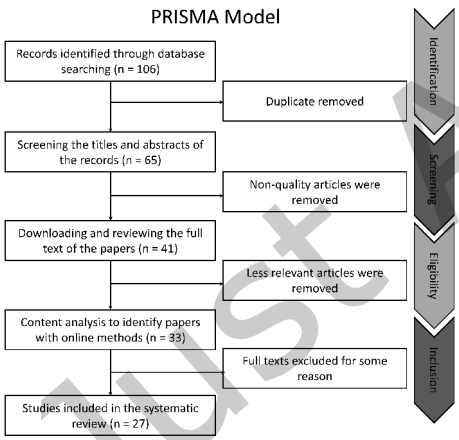


Fig. 8. Flow chart of the methodology chosen for article searching and screening (Step-wise representation)

Table 7. Parameters of the survey

Parameter	
Literature databases	CiteSeerX, ACM digital library, IEEE Explore, SCOPUS, Google Scholar
Journal databases	SpringerLink, Science Direct Journals, Elsevier, IEEE, Archive
Types of publications	Archive, journal articles, conference papers
Inclusion/ Exclusion criteria	Relevant to XAI and malware analysis
Keywords	Malware analysis, XAI, Interpretation, Explainable, Transparent models, Adversarial learning, Adversarial machine learning, Evasion attacks, Poisoning attacks, Deep learning, Adversarial examples, Cyber security, Fragile XAI

We conducted a systematic review for our proposed literature survey. Researchers suggested various frameworks or approaches for conducting literature reviews in a systematic way, such as the PRISMA model [113] (see Fig. 8).

To perform our study we conducted a thorough search for related articles using various databases such as Google Scholar, Scopus, and different journals' websites. Our initial step involved filtering the records and eliminating duplicates. We then proceeded to remove non-qualitative articles. To select the papers once the dataset was finalized we applied eligibility criteria based on specific features such as relevance to XAI and malware analysis. Ultimately we selected 27 articles that met our criteria for evaluation in the study, as shown in Figure 8. The filtering process was based on the relevance of articles to the topic under investigation. Table 7 highlights the parameters we used in our survey.

During the revision of the paper, we repeated the same process with the same selection criteria for papers published from 2023 onward and the analysis. Therefore, Figure 8 does not include the second revision phase.

C. Future Directions and Emerging Approaches

This section provides an overview of these advancements and discusses their potential impact on future research and practical applications.

Unstructured and multi-modal data integration: Unstructured data such as call graphs, CFGs, and API graphs, using Graph-based models like GAGE [107] and CFGExplainer [40], have shown promise in capturing the complex relationships within malware code. These models can provide more interpretable insights by analyzing malicious executables' code and data flow graphs. The robustness and discriminative power of these models suggest they will play a crucial role in advancing EMD. Moreover, the integration of multi-modal data, combining information from different sources such as network traffic, system logs, and binary analysis, offers a comprehensive view of malware behavior. Multi-modal approaches can improve detection accuracy and provide richer explanations by leveraging diverse data types. For instance, HYDRA [33] learns from various sources to maximize the benefits of multiple feature types to reflect the characteristics of malware executables. Future research should focus on developing frameworks that effectively integrate and analyze multi-modal data to enhance both the performance and interpretability of EMD.

Federated learning and privacy-preserving techniques: Recent studies have highlighted the importance of privacy-preserving techniques in malware detection, particularly in environments like IoT devices where data sensitivity is paramount. Federated learning, which enables model training across decentralized devices without sharing raw data, has gained traction. For instance, D'Angelo et al. [28] demonstrated a significant advancement in this area by integrating transfer learning and federated learning to improve regression analysis in malware detection. This approach not only enhances privacy but also maintains high accuracy and efficiency, making it a valuable direction for future research.

Advances in NLP: The application of NLP techniques to malware analysis is an emerging area that leverages the power of models based on Large Language Models (LLMs). For example, this survey [143] highlights that using LLMs such as GPT-4 to detect malware is a promising application. These models (e.g., BERT, GPT) can analyze code and documentation to identify patterns indicative of malicious behavior. By integrating NLP with traditional malware detection methods, researchers can improve the interpretability and accuracy of their models. Future work could explore the synergy between NLP and other explainability techniques to enhance the transparency of malware detection systems.

Quantum machine learning: Quantum computing holds potential for significant advancements in machine learning, including malware detection. Quantum Machine Learning (QML) algorithms can process information at unprecedented speeds, potentially improving the efficiency and accuracy of detection models. For instance, Giovanni et al. [23] present a malware detection method using quantum machine learning, comparing its performance and explainability with CNNs. Although still in the early stages, exploring the application of QML to XAI and malware detection could open new avenues for research and development.

Received 20 August 2023; revised 14 June 2024; accepted 5 July 2024