The Indonesian Journal of Computer Science
www.ijcs.net
Volume 15, Issue 3, June 2026
https://doi.org/10.33022/ijcs.v15i3.5143

An Extensive Analysis and Taxonomy of Explainable Artificial Intelligence for Malware Identification

Dauan Aziz[1], Firas Mohammed Amien[2], Raghad Zuhair Yousif [3]
dauanaziz@gmail.com[1], f.almukhtar@cue.edu.krd[2], raghad.yousif@epu.edu.iq[3]
[1,3]Erbil Polytechnic University
[2]Catholic University of Erbil

Article Information

Received : 18 Apr 2026
Revised : 20 May 2026
Accepted : 9 Jun 2026

Keywords

malware detection, Explainable AI (XAI), machine learning, model interpretability, Windows PE, SHAP, deep learning, LIME, cybersecurity, post-hoc explanation

Abstract

As malware continues to evolve in sophistication and scale, traditional detection methods struggle to keep pace, especially when facing obfuscated or zero-day threats. In response, Machine Learning (ML) and Deep Learning (DL) techniques have shown significant promise in enhancing malware detection through pattern recognition and anomaly classification. However, their increasing complexity has introduced major interpretability challenges, particularly in high-stakes cybersecurity contexts. This paper provides a comprehensive survey of eXplainable Artificial Intelligence (XAI) methods applied to malware detection across diverse computing platforms, including Windows PE files, PDF, Linux, and hardware-based systems. We propose a novel taxonomy that categorizes explainable malware detection approaches by model transparency, explanation technique (model-agnostic or model-specific), and deployment environment. We also discuss major trends, highlight underexplored domains, and outline future research directions aimed at enhancing real-time interpretability, adversarial robustness, and human-in-the-loop integration. This work aims to bridge the gap between high-performance malware detection models and actionable, transparent security decision-making.

Attribution-ShareAlike 4.0 International License

A. Introduction

In today's rapidly evolving digital environment, malware continues to pose a significant threat, leading to substantial financial damages and interrupting essential services worldwide. The emergence of increasingly advanced malware variants, especially zero-day threats, has diminished the efficacy of conventional detection mechanisms. Consequently, the cybersecurity community is witnessing an urgent demand for intelligent, adaptive malware detection frameworks capable of addressing these sophisticated attack vectors.

Machine Learning (ML) and Deep Learning (DL) have demonstrated significant potential in detecting both known and unknown malware strains, despite this difficulty. These techniques have demonstrated strong capabilities in modeling complex data patterns, making them suitable for detecting anomalies and zero-day attacks. Yet, the complexity of such models tends to increase, leading to lost interpretability and eventually making them into a black box, the inner decision processes of which might not be easily comprehensible to cybersecurity analysts (Roscher, Bohn, Duarte & Garcke, 2020). This lack of transparency prevents trust and diagnostic assessment and leads to difficulties in identifying false positives and model failures.

In an attempt to fill this interpretability gap, there is a movement of increasing interest in the area of Explainable Artificial Intelligence (XAI). XAI is designed to make AI systems more transparent and comprehensible by providing understandable rationales behind model decisions. Using XAI in malware detection improves trust, facilitates root-cause analysis, and gives security experts the confidence to act on model outputs (Saranya & Subhashini, 2023).

Numerous earlier reviews have examined XAI from a range of angles, including methods, theoretical underpinnings, and domain-specific applications. Research like (Das & Rad 2020, Xu, Uszkoreit, Du, Fan, Zhao & Zhu 2019, Islam, Eberle, Ghafoor & Ahmed 2021, Weber, Lapuschkin, Binder & Samek 2023, Chamola, Hassija, Sulthana, Ghosh, Dhingra & Sikdar 2023) has provided summaries of XAI techniques that are interpretable both mathematically and visually. For instance, Mohseni et al. (Mohseni, Zarei & Ragan 2021) suggest evaluation frameworks for XAI systems, while Danilevsky et al. (Danilevsky, Qian, Aharonov, Katsis, Kawas & Sen 2020) focus on XAI in natural language processing (NLP). Confalonieri et al. (Confalonieri, Coba, Wagner & Besold 2021) chart the development of XAI in expert systems, whereas Li et al. (Li, Cao, Shi, Bai, Gao, Qiu, Wang, Gao, Zhang, Xue & Chen 2022) distinguish between knowledge-driven and data-driven interpretability approaches. Furthermore, Speith (Speith 2022) offers a thorough taxonomy of XAI techniques and related difficulties.

Additional noteworthy studies are Milani et al. (Milani, Topin, Veloso & Fang 2024), who survey explainable reinforcement learning, and Saeed and Omlin (Saeed & Omlin 2023), who talk about the difficulties and potential paths in XAI. From a cybersecurity perspective, Charmet et al. (Charmet, Tanuwidjaja, Ayoubi, Gimenez, Han, Jmila, Blanc, Takahashi & Zhang 2022) look into the integration of XAI inside cybersecurity more generally, while Nasser and Nasser (Nasser & Nassar 2023) discuss hardware-assisted ML approaches for malware detection. However, none of these surveys focus specifically on explainable AI in malware detection and classification.

Although interest in XAI has surged between 2018 and 2024, there remains a noticeable gap in the literature addressing explainable methods tailored to malware analysis. Most existing surveys either overlook this specific domain or fail to distinguish clearly between "interpretability" and "explainability" in the context of AI. Lin and Chang (Lin & Chang, 2021b) are among the few who explicitly categorize interpretable malware detection models.

The current survey fills this gap by providing a comprehensive examination of malware detection methods that are both interpretable and explainable. Our work synthesizes previous methodological developments and presents a new taxonomy for explainability approaches in this field. The overarching aim is to present a consolidated view of current efforts, practical implementations, and unresolved research questions in explainable malware detection.

The key contributions of this paper are as follows:

- We present a comprehensive survey of XAI models and techniques applied across diverse domains, highlighting their relevance and potential in malware analysis.
- A detailed review of ML-based malware detection strategies is provided, establishing the intersection between explainable methods and cybersecurity tools.
- Lastly, we propose future research directions for enhancing explainability in malware detection systems, encouraging the development of more transparent and domain-generalizable models.

Our literature review was conducted through a systematic search of major scholarly databases, including Google Scholar, IEEE Xplore, ScienceDirect, ResearchGate, arXiv, ACM Digital Library, and Springer. A comprehensive set of keywords was employed to capture the relevant literature, such as "explainable machine learning," "explainable artificial intelligence," "XAI," "explainable malware detection," "interpretable machine learning," and "interpretable malware analysis." This ensured a broad and inclusive collection of studies, focusing specifically on the convergence of explainability and malware detection.

B. Malware Detection Approaches

Malware detection remains a critical component of cybersecurity frameworks, aiming to identify and mitigate malicious threats before they can compromise systems. Broadly, detection strategies are classified into two principal categories: file-based classification methods and online behavior-based techniques. All these paradigms bring with them new strengths to increase the accuracy of detection and system resilience.

B.1. File Classification Approaches

File classification methods examine files that can execute programs and find out whether they are potentially malicious. Based on the detection of a suspicious file, these techniques are dependent on one or all of the following analysis techniques: Static analysis, dynamic analysis, and a combination of both. The methods are done in static checking the file without running it, dynamic checking the behavior at the actual run time in a controlled environment, and the hybrid methods are those that take advantage of both to maximize the detection.

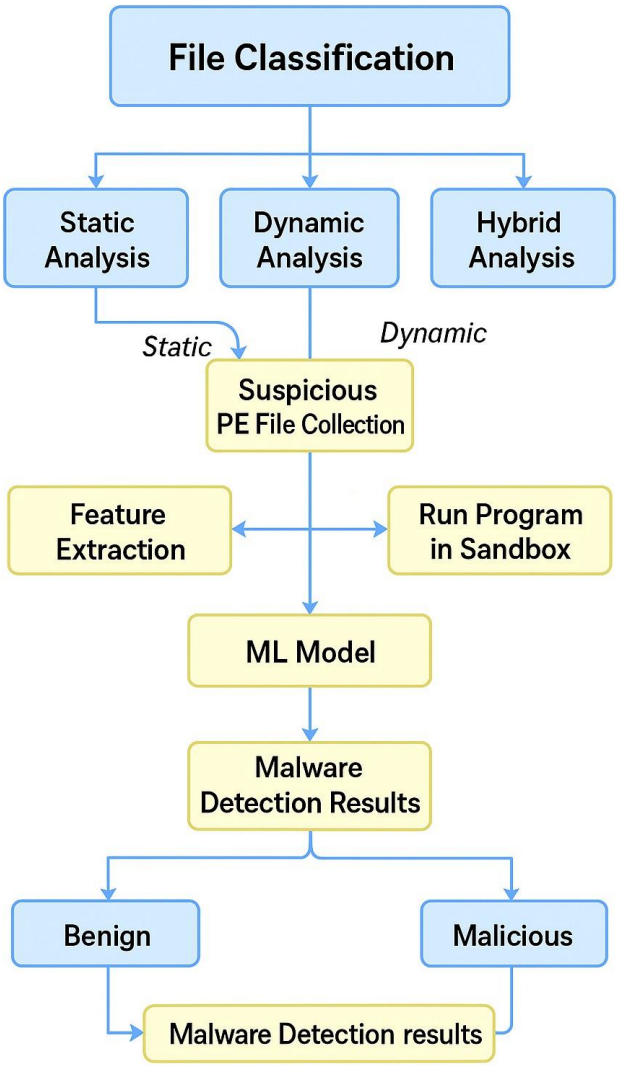

Figure 1. Overview of file-based malware detection approaches including static, dynamic, and hybrid analysis techniques. Suspect PE files are then processed through feature extraction when performed through code analysis (static) or behavioral analysis in a sandbox (dynamic). Techniques that are extracted inform machine learning (ML) models that label benign or malicious files.

B.1.1. Analysis of Statics

Static analysis looks at an executable code's structure and contents without ever running the code. This is intended to find the malicious motive by finding patterns or cryptographic signatures that are consistent with malware samples that are known (Damodaran, Di Troia, Visaggio, Austin, & Stamp 2017). It frequently entails reverse engineering of the binary code and retrieving important data, including cryptographic hashes, opcodes, and API calls. According to studies by Hou et al. (Hou, Saas, Chen, Ye & Bourlai, 2017) and Kim and Lee (Kim & Lee, 2021), alternative methods of creating the approach have been presented, such as

using stacked autoencoders to rebuild Application Programming Interface (API) calls within Portable Executable (PE) files.

Such properties as opcode patterns, N-Grams, and API traces are then fed into the machine learning models to determine the type of file. To provide an example, Shankarapani et al. (Shankarapani, Ramamoorthy, Movva & Mukkamala, 2011) used opcode-based and API-based sequences to determine code fragments looking similar to malicious behavior. Although static analysis is both quick in detecting malware and resource sparing, it is usually unsuccessful in identifying obfuscated or polymorphic malware and does not record real-time malicious behavior.

B.1.2. Dynamic Analysis

Dynamic analysis is the act of running the suspicious file in a security sandbox (or other) and monitoring its actions in real-time (Damodaran et al. 2017, Kakisim, Nar, Carkaci & Sogukpinar 2019). This is an especially efficient method against zero-day malware that does not get noticed by a static signature detecting tool. During execution behavioral data is gathered in the form of system calls, memory consumption and number of functions called, and used as features to input machine learning classifiers.

It has been demonstrated by Firdausi et al. (Firdausi, Lim, Erwin & Nugroho 2010) and Luckett et al. (Luckett, McDonald & Dawson 2016) that dynamic features like system call traces are helpful in teaching models like k-NN, Support Vector Machines (SVM), and Decision Trees. Further research by Pirscoveanu et al. (Pirscoveanu, Hansen, Larsen, Stevanovic, Pedersen & Czech 2015) and Tobiyama et al. (Tobiyama, Yamaguchi, Shimada, Ikuse & Yagi 2016) also introduced the use of convolutional neural networks (CNNs) and API call patterns to enhance malware detection during execution. Nevertheless, even though it is computationally expensive to perform dynamic analysis it involves a large consumption of time because a secure environment requires it to run on its run time.

B.1.3. Analysis of Hybrids

The goal of hybrid analysis is to provide a more thorough examination of questionable files by combining the benefits of static and dynamic approaches. (Tzermias, Sykiotakis, Polychronakis & Markatos 2011). Such a comprehensive method has been used in several studies. As an example, Santoso et al. (Santoso, Heryadi, Warnars, Wulandhari, Lukas & Abdurachman, 2019) used the combination of a hybrid approach (Artificial Neural Networks (ANN) and CNN malware classification).

Tong and Yan (Tong & Yan, 2017) introduced the comparative approach in which the patterns of dynamic behavior of unknown applications are compared to the patterns of known beneficial and malicious apps. Their experiments hint that using a combination of static and dynamic features increases the accuracy of classification.

B.2. Online Malware Detection

Unlike file classification methods that evaluate specific files, online malware detection focuses on real-time monitoring of system-wide activities to identify

malicious behavior as it occurs. This approach does not depend on prior file identification and is particularly useful for detecting stealthy or low-profile threats.

Watson et al. (Watson, Shirazi, Marnerides, Mauthe & Hutchison 2016) published one of the first attempts in this field, training a Support Vector Machine (SVM) with system performance indicators and obtaining a 90% accuracy rate. While Abdelsalam et al. (Abdelsalam, Krishnan & Sandhu 2017) presented a sequential k-means clustering technique designed for virtualized environments, specifically targeting three-tier architectures on OpenStack, Azmandian et al. (Azmandian, Moffie, Alshawabkeh, Dy, Aslam & Kaeli 2011) investigated intrusion-based models. Although effective, their system exhibited reduced detection capabilities for low-resource-consuming malware. The efficiency of different CNN architectures for malware identification in Infrastructure as a Service (IaaS) cloud platforms was examined by McDole et al. (McDole, Abdelsalam, Gupta & Mittal 2020). In their subsequent study (McDole, Gupta, Abdelsalam, Mittal & Alazab 2021), they compared the performance metrics at the process level of various deep learning models in comparable cloud-based settings.

Additionally, Kimmel et al. (Kimmell, Abdelsalam & Gupta 2021, Kimmel, McDole, Abdelsalam, Gupta & Sandhu 2021) assessed several machine learning techniques for identifying malicious activities within virtual machines, emphasizing CNNs due to their robust 2D representation capabilities. In a more advanced approach, Abdelsalam et al. (Abdelsalam, Krishnan, Huang & Sandhu 2018) employed three-dimensional CNNs to capture temporal and spatial patterns, thereby enhancing accuracy and the detection of stealthy malware.

C. Explainability in Machine Learning

Deep learning (DL) architectures have empowered machines to uncover intricate hierarchical patterns within data, significantly improving their performance in tasks such as classification and anomaly detection. These models achieve remarkable predictive accuracy by constructing multiple layers of feature representations. However, the increased model depth and complexity often lead to a lack of transparency, making it difficult to comprehend the internal rationale behind their decisions (Linardatos, Papastefanopoulos & Kotsiantis 2020, Arrieta, Díaz-Rodríguez, Del Ser, Bennetot, Tabik, Barbado, Garcia, Gil-Lopez, Molina, Benjamins, Chatila & Herrera 2020).

In contrast, white-box models are designed to be interpretable, enabling users to track and comprehend the ways in which input variables influence output projections. Such transparency is especially critical in high-stakes domains where comprehending the reasoning behind predictions is essential—such as healthcare, where a misclassification in cancer diagnosis could lead to serious consequences (Hassija, Chamola, Mahapatra, Singal, Goel, Huang, Scardapane, Spinelli, Mahmud & Hussain 2024, Dwivedi, Dave, Naik, Singhal, Omer, Patel, Qian, Wen, Shah, Morgan & Ranjan 2023). For clinicians and patients alike, trust in predictive tools depends heavily on the ability to explain the model's outcomes.

This is where Explainable Artificial Intelligence (XAI) becomes pivotal. As illustrated in Figure 2, conventional AI workflows focus on training data and producing predictions for end users. XAI enhances this process by not only delivering high-performance outputs but also providing human-understandable

justifications for those results. The integration of explainability increases model credibility, supports error analysis, and facilitates decision-making.

Blanco-Justicia and Domingo-Ferrer (Blanco-Justicia & Domingo-Ferrer 2019) list seven essential characteristics of successful XAI systems that improve usability and transparency:

- Accuracy: Measures how well the XAI model performs on new, unseen instances. High predictive accuracy is foundational.
- Fidelity: Evaluates the degree to which the explanation accurately captures the behavior of the underlying model. Great-fidelity explanations match predictions with great accuracy.
- Consistency: Ensures that explanations remain uniform when the same model is trained on similar data.
- Stability: Indicates whether similar inputs yield similar explanations, reinforcing the model's reliability.
- Degree of Importance: Makes it easier to understand how each feature contributes to the model's prediction.
- Novelty: Evaluates whether explanations generalize to previously unseen or outlier inputs.
- Representativeness: Emphasizes the need for explanations that are broadly applicable across various decision contexts.

A generic classification of machine learning models is shown in Figure 3 to provide a deeper understanding of explainability in ML. A taxonomy that divides models into two major types is shown in this graphic: Transparent and Opaque.

Transparent models—such as decision trees, linear regression, or rule-based systems—are inherently interpretable and generally do not require additional tools for post-hoc explanation. However, as model complexity increases (as denoted by the dashed arrow in Figure 3), even transparent models may benefit from supplementary explainability mechanisms to improve user understanding.

Opaque models, which are typically referred to as conventional ML or deep learning models, often deliver higher accuracy but lack inherent interpretability. Their decision-making processes are not directly observable and thus require post-hoc explainability techniques to make predictions understandable to humans.

Two main categories can be used to further categorize post-hoc explainability:

- Model-Independent Approaches: These techniques are flexible and can be applied to any type of ML model, providing broad applicability across different architectures.
- Model-Specific Methods: Customized for particular model types (such as tree-based or neural network models), these techniques provide more customized insights but are limited in scope.

The next section will explore a range of ML models and explainability techniques, with an emphasis on post-hoc methods. It will also address open research problems associated with explainable ML in security-critical applications like malware detection.

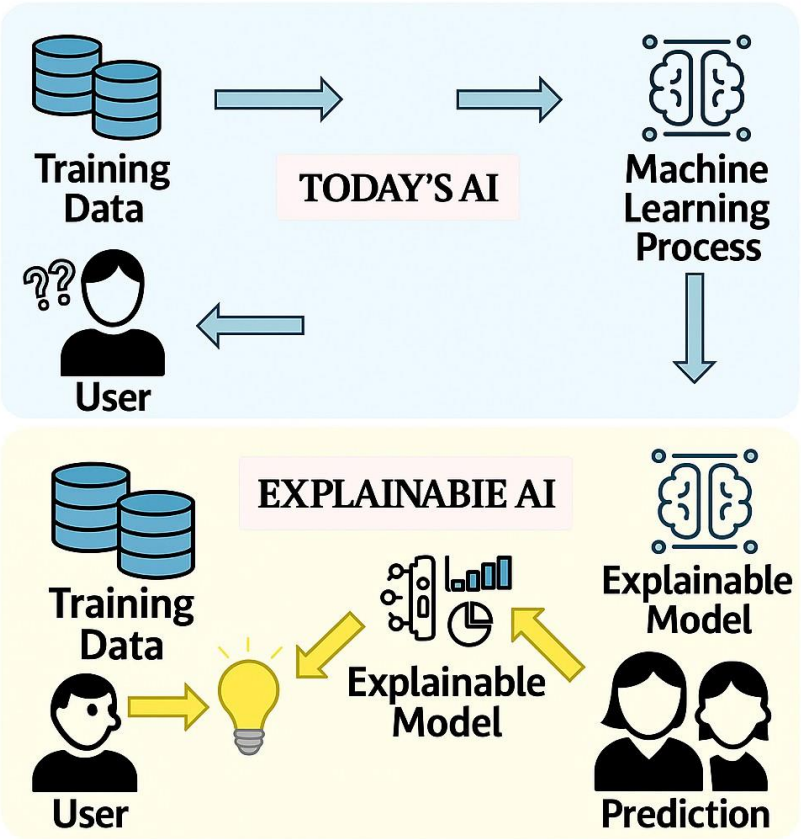

Figure 2. Comparison of conventional AI and Explainable AI (XAI) pipelines. While traditional systems output predictions, XAI systems provide interpretable explanations along with high-accuracy outputs.

The taxonomy in Figure 3 organizes Explainable AI (XAI) into four categories: Cause-Effect Discovery, which uses structured and constraint-based learning to uncover causal links (e.g., PC Algorithm, Hill Climbing); Causal AI, which explains through associations, interventions, and counterfactuals (e.g., Bayesian Networks, Structural Causal Models); Transparent AI, which relies on interpretable models like decision trees, random forests, and feature selection; and Neuro-AI, which seeks to explain complex black-box models such as neural networks and deep learning. Together, these approaches highlight different strategies for making AI more understandable and trustworthy.

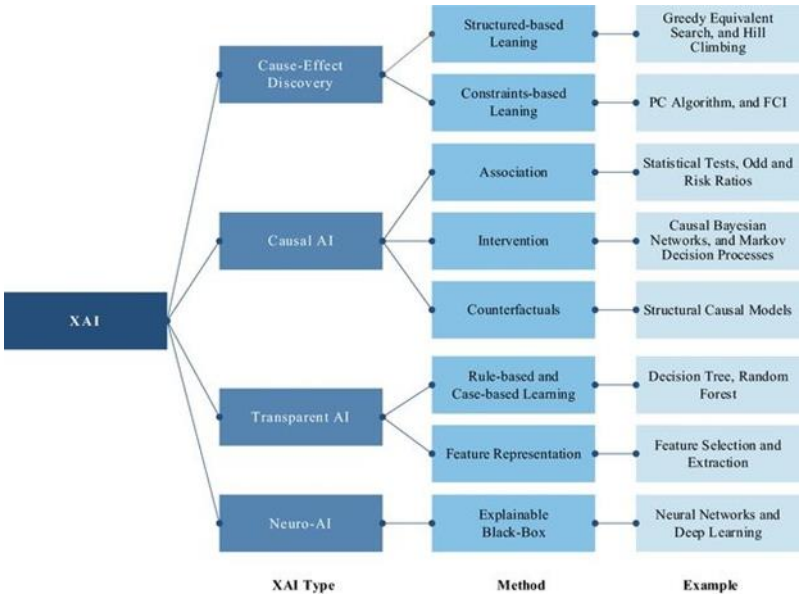

Figure 3. Taxonomy of machine learning models in the context of explainability. ML models are categorized as transparent or opaque. Post-hoc explainability is necessary primarily for opaque models (Sahoh & Choksuriwong 2023).

C.1. Models for Transparent Machine Learning

Transparent machine learning models give consumers a clear understanding of how they make decisions since they are naturally interpretable. These models are designed to be self-explanatory, allowing for straightforward tracing from input features to final predictions. Rule-based systems, Bayesian classifiers, decision trees, linear and logistic regression, k-Nearest Neighbors (k-NN), and Generalized Additive Models (GAMs) are typical examples of these models.

C.1.1. Rule-Based Models

Rule-based learning systems operate by generating interpretable logic-based rules from data. Typically, these models are based on straightforward IF-THEN structures, in which a condition is captured by the IF clause and the related prediction is determined by the THEN clause. Such rules may function independently or be aggregated for ensemble decision-making. For instance, Soares et al. (Soares, Angelov, Costa, Castro, Nageshrao & Filev 2021) demonstrated the effectiveness of rule-based logic in approximating deep reinforcement learning behaviors, thereby enhancing the interpretability of otherwise opaque systems. Owing to their structural clarity, rule-based models are often employed both as standalone predictors and as surrogate models to explain complex ML systems.

C.1.2. Models of Regression

One of the most popular interpretable models is logistic regression, along with linear regression. Linear regression offers an intuitive understanding of how features impact output through coefficient weights, making it a staple in domains like economics and biomedical science. Logistic regression extends this capability by producing probabilistic classifications, allowing for nuanced decision

thresholds. Lundberg (Lundberg, Erion, Chen, DeGrave, Prutkin, Nair, Katz, Himmelfarb, Bansal, & Lee, 2019) proposed combining logistic regression with gradient-boosted trees to improve both performance and explainability in synthetic datasets. Despite their interpretability, regression models can benefit from visualization techniques or feature importance measures to make them more accessible to non-expert users.

C.1.3. Decision Trees

Decision trees offer clear and structured decision paths, allowing domain experts to follow each branch to its associated prediction. This makes them a powerful tool for both classification and explanation. Blanco-Justicia and Domingo-Ferrer (Blanco-Justicia & Domingo-Ferrer, 2019) employed decision trees as surrogate models in their work. approximating more complex ML systems by training trees on specific subsets of the original data. However, as decision trees grow in size—especially in large-scale real-world datasets—their structure can become overly complex, reducing their practical interpretability. In such cases, post-hoc methods may be needed to simplify explanations.

C.1. 4. Bayesian Models

Bayesian learning techniques offer probabilistic insights into data relationships, making them suitable for applications that require transparent reasoning under uncertainty. These models quantify the strength of correlations between variables, which is particularly beneficial in medical and scientific contexts. Arrieta et al. (Arrieta et al. 2020) showed how Bayesian inference may be used in healthcare analytics to help better identify the relationships between diseases and symptoms. Similarly, Naive Bayes classifiers have been widely used for classification problems, leveraging Bayes' theorem to compute class probabilities (Rana, Gudla & Sung 2018). Their transparency and computational efficiency make them highly interpretable, especially in low-dimensional settings.

C.1.5. k-Nearest Neighbors (k-NN)

The k-NN algorithm classifies a test instance based on the majority class among its nearest neighbors or computes an average in regression tasks. The transparency of k-NN depends on the clarity of the selected features, the choice of distance metric, and the number of neighbors considered. Although models trained on high-dimensional data may lose interpretability, a well-tuned k-NN with a small and meaningful feature set remains highly interpretable. Aslam et al. (Aslam, Khan, Aljishi, Alnamer, Alzawad, Almomen & Alramadan 2022) demonstrated how k-NN and other supervised models can be paired with XAI techniques to improve transparency and trust in predictions.

C.1.6. Models of generalized additives (GAMs)

GAMs offer a balance between predictive power and interpretability by extending linear models with non-linear functions for individual features. These models are particularly useful in highlighting the individual contribution of each feature without introducing excessive model complexity. Each feature is modeled using a smooth function, making the effect of each variable directly visible. Yang et

al. (Yang, Zhang & Sudjianto 2021) introduced GAMI-Net, a neural network-based extension of GAMs, designed to preserve the model's transparency while improving accuracy. GAMI-Net outperformed traditional GLMs while maintaining interpretability, making it suitable for high-stakes applications.

Despite their intrinsic transparency, some scenarios still warrant post-hoc explanation, particularly when transparent models are trained on large or high-dimensional datasets. Naturally interpretable models may need further explanation aids to preserve clarity in complex decision-making situations, as shown by the dashed arrow in Figure 3. This highlights the evolving nature of explainability, where transparency is a spectrum and may not always suffice without supplementary techniques.

C.2. Opaque Machine Learning Models

Although interpretability is by definition inherent in transparent models, they are usually accompanied by lower predictive performance on complex tasks. Conversely, opaque, or black-box, models often are more accurate and better at generalizing; they are also sometimes less interpretable in human terms. Such models cannot be directly used without integrating post-hoc explainability methods to make their decision processes interpretable. The subsequent subsections outline some of the commonly utilized machine learning opaque models and the methods that are used to make them more explainable.

C.2.1. The Random Forest

Random Forest (RF) is an ensemble training algorithm that combines the results of a set of decision trees in order to enhance the classification and regression results. The feature space of the input is divided in a different way in each decision tree, and predictions are usually made based on the average or a majority vote of the trees. This bagging method minimizes the variance and avoids the problem of overfitting, which is characteristic of a single decision tree, by training the trees on a bootstrapped version of the given data and a random selection of features.

The more trees there are and how deep they become in order to accommodate more complicated datasets, the less understandable the whole picture is. Therefore, Random Forests can be deemed to be opaque, whereas they consist of interpretable elements. In order to make them more explainable, Zhao et al. (Zhao, Wu, Lee & Cui 2019) have suggested a system of visual analytics where the user can traverse the decision boundary and tree interactions in the forest. This would enhance the understanding by the user as to how the individual features can affect the predictions of the ensemble.

C.2.2. Support Vector Machine (SVM)

Support Vector machines are a potent supervised learning method that may be used to do classification, regression, and even anomaly detection. They operate with the help of constructing an optimal hyperplane that sequesters data points of various classes in the largest margin, which is immediately connected with the ability of the model to generalize.

Even though the predictive power of SVMs is robust, their complexity rate grows enormously in high-dimensional space, especially when being applied with a non-linear kernel. This makes the inner decision-making process black. To solve this, explanatory techniques of post-hoc, like surrogate modeling, have been considered. To get an example, Vieira and Digiampietri (Vieira & Digiampietri, 2020) used decision trees to approximate and extract human-readable rules from trained SVM models. These surrogate models will offer a little understanding into the reasoning of the SVM classification and create trust and readability.

C.2.3. Multi-Layer Neural Networks

Deep neural networks (DNNs) also known as multi layer neural network are largely considered to be the most accurate in performing tasks that involve unstructured data e.g. images, audio and text. These models are based on many layers of hidden elements that learn about hierarchy of data. Their inner transformations are, however, highly non-linear, highly complex, and thus highly opaque and incomprehensible.

Sharma et al. (Sharma, Schommer & Vivarelli 2020) examined how to predict credit default risk using a multi-layer perceptron (MLP). To understand the behavior of the model they applied sensitivity analysis, which is a post-hoc technique used to understand how small differences in the input characteristics impact on the output. This method enables the stakeholder to determine powerful variables and know the rationale behind certain predictions.

Neural networks in spite of all their benefits have the drawback of users not being able to interpret their outputs without additional post-hoc explanations with such methods as saliency maps, SHAP values, or local surrogate models (e.g., LIME). Since the application of these models to sensitive fields rises, interpretability is a matter of essential consideration.

C.3. Methods for Post-Hoc Explainability That Are Independent of Model

In high-stakes applications, such as malware detection, the interpretation of decisions made by such complicated inductive models is crucial, in contrast to low-stakes activities where models are employed to develop robotic navigator plans. Techniques for post-hoc explanations are essential in these situations, especially when the inner workings of the models are not interpretable. Among the best approaches regarding classification of post-hoc strategies is a distinction between model-agnostic and model-specific. Interestingly, model-agnostic algorithms are flexible to use in various architectures irrespective of the respective model (Islam et al. 2021). Even though model-agnostic methods are flexible, they tend to behave similarly to the original model, leading possibly to the loss of precision in the produced explanations. These methods are broadly classified into three ways, which are, in particular, visual, local, and global explanation.

C.3.1. Worldwide Explanations

Global explanation methods strive to give a general picture of model functioning at the level of complete data. Instead of paying attention to the values of particular predictions, they demonstrate what characteristics tend to determine the outcomes and what systemic interactions exist between characteristics.

One popular way of addressing interpretability in global models is to use surrogate models, i.e., simple, interpretable models like decision trees or linear regressions, to do a crude approximation of how more complex models behave. Surrogate models make it easier to interpret a model, but they face the risk that all the complexity of the original model might not be reflected in the surrogate model, particularly when feature interactions are non-linear or otherwise complex (Islam et al. 2021). In cybersecurity settings, the surrogate models may assist an analyst in finding out whether such factors as API call patterns or metadata attributes are primary characteristics in the malware classification problem.

Islam et al. (Islam et al. 2021) illustrated this method in which their classifier was trained (a CART, aka Classification and Regression Tree) to replicate the choices made by a Random Forest classifier. They got to find that surrogate models can be used to inform conclusions about model behavior, particularly when the intended use is interpretability, and not paramount accuracy.

C.3.2. Local Explanations

In local explanation techniques, there is an emphasis on the explanation of individual predictions rather than overall behavior. Such techniques are handy for cases like: "Why has this file been labeled as malware?" Local approaches support investigations, bias analysis, and fairness evaluations by assisting stakeholders in comprehending the role of each feature in particular decision contexts.

LIME: Ribeiro et al. (Ribeiro, Singh & Guestrin 2016) developed Local Interpretable Model-Agnostic Explanations (LIME), a model-agnostic technique that uses basic surrogate models to approximate the model's decision boundary in the neighborhood of a particular instance. Although LIME provides useful local insights, its reliability depends on the local fidelity score—how accurately it approximates the complex model's behavior in that region. Magesh et al. (Magesh, Myloth & Tom 2020) applied LIME to interpret CNN-based predictions for early Parkinson's detection, showing its practical utility in sensitive domains.

KernelSHAP: Derived from game theory, SHAP (Shapley Additive exPlanations) values introduced by Lundberg and Lee (Lundberg & Lee, 2017) measure the contribution of each feature to a specific prediction. KernelSHAP is the model-agnostic version of SHAP and offers theoretically grounded, consistent attributions. While highly interpretable, SHAP's computational cost can be substantial, especially for large models or datasets. Specialized versions such as TreeSHAP and DeepSHAP aim to address this issue for specific model types.

Shapley Values: Based on coalition game theory, each feature is treated as a "player" and the model's prediction as the "payout." The contribution of each feature is then computed to determine how it influenced the outcome. Shapley values ensure fairness through principles like consistency and local accuracy.

Counterfactual Explanations: These methods identify the small feature modifications that would produce a different forecast. Counterfactual explanations do not require access to the internal structure of the model, making them model-agnostic and intuitive for end-users. According to Molnar (Molnar 2022), this method is especially useful in adversarial contexts like malware classification, where knowing how to reverse a classification might expose flaws in the system.

Logic Explained Networks (LENs): First-Order Logic (FOL) outputs and human-readable predicates are two ways that LENs improve neural network interpretability (Ciravegna, Barbiero, Giannini, Gori, Lió, Maggini, & Melacci, 2023). These networks can operate in both supervised and unsupervised settings, generate rule-based explanations, and even mimic complex models while offering transparent, symbolic reasoning.

C.3.3. Visual Explanations

Visual explanation methods provide graphical interpretations of how features influence model predictions, making them accessible to a broader audience. Techniques such as Partial Dependence Plots (PDP), Individual Conditional Expectation (ICE), and Accumulated Local Effects (ALE) plots are commonly used.

Partial Dependence Plot (PDP): PDP illustrates the marginal effect of one or two features on the predicted outcome (Molnar 2022). It is useful for understanding average model behavior but assumes feature independence, which may limit its accuracy in datasets with correlated variables.

Conditional Expectation for Individuals (ICE): Goldstein et al. proposed ICE plots, that indicate the influence of a feature on the predication of individual instances (Goldstein, Kapelner, Bleich & Pitkin 2015). In contrast to PDPs, ICE plots give finer scale resolution and can capture heterogeneity between samples, and thus can be applied to an explorative analysis of complex models.

Local Effects Accumulated (ALE): ALE plots, which do not require the feature independence assumption required by PDP (Krämer, Stang, Nagl & Schäfers 2023), provide a visual representation of how features impact the predictions. They super efficient and they are applicable in sectors like real estate and cybersecurity where it would be important to know the effect of features. Kramer et al. (Krämer et al. 2023) demonstrated ALE's applicability by identifying influential factors in property valuation models.

C.4. Model-Specific Post-Hoc Explainability Methods

While model-agnostic techniques offer flexibility across various architectures, model-specific methods are tailored for particular model types, enabling more precise and optimized interpretability. These approaches can be categorized by their scope—local (instance-level explanation), global (model-wide behavior), and visual (graphical representations)—each enhancing interpretability in its own context.

TreeSHAP and DeepSHAP are two well-known model-specific techniques that support tree-based and deep learning models, respectively. Additionally, saliency maps—including methods like DeepLIFT, Grad-CAM, and Layer-wise Relevance Propagation—serve as visual tools to explain feature contributions in neural networks.

C.4.1. TreeSHAP and DeepSHAP

TreeSHAP and DeepSHAP are specialized implementations of the SHAP framework, which itself is grounded in Shapley value theory. TreeSHAP is optimized for decision tree ensembles and offers polynomial-time computation of exact SHAP values, enhancing both scalability and interpretability. Athanasiou et

al. (Athanasiou, Sfrintzeri, Zarkogianni, Thanopoulou & Nikita 2020) demonstrated the application of TreeSHAP in a cardiovascular risk prediction system, where it generated personalized, feature-based explanations to support clinical decision-making.

DeepSHAP, on the other hand, is designed for deep neural networks. It builds upon the DeepLIFT framework and estimates Shapley values using conditional expectations derived from a background dataset. This allows DeepSHAP to evaluate feature contributions to a model's output with a focus on local interpretability. Davagdorj et al. (Davagdorj, Bae, Pham, Theera-Umpon & Ryu 2021) successfully employed DeepSHAP to uncover critical risk factors in the prediction of non-communicable diseases, offering intuitive, patient-specific insights from neural network predictions.

C.4.2. Feature Relevance Explanations

This family of methods emphasizes the quantification of how features influence a model's predictions. It includes techniques like feature importance, extraction, and contribution assessments.

Adebayo and Kagal (Adebayo & Kagal 2016) suggested an efficient iterative approach to ranking features in terms of their importance-removing and changing features and observing the effect on the performance of a model. An outcome of this process is more well-groomed datasets, and a selected predictors with only those that hold the most importance stand to be used in custom model optimization.

Building on this concept of interaction effects, Friedman and Popescu (Friedman & Popescu 2008) introduced the so-called H-statistic, counting how much the prediction variance is explained by the interactions between features. Such a measure allows revealing intricate relationships that can be masked by orderly importance scores which is useful in better understanding model behavior.

C.4.3. Saliency Maps

In visual or high-dimensional tasks in a task, saliency maps are essential in interpreting neural network predictions. These methods also determine the most influential components of the input that contributes to decision of the model.

Layer-wise Relevance Propagation (LRP) evaluates individual features in the input and provides them with an importance score according to their participation in the overall result. Using the LRP in the malware detection field, Wang et al. (Wang, Zhu, Tong, Yin, Feng, Shi, & Meng 2021) managed to visualize malicious code traces in the binary files, facilitating the process of interpretable security.

DeepLIFT, based on the reference baseline, the way it was introduced by Shrikumar et al. (Shrikumar, Greenside & Kundaje 2017), compares the neuron activations to determine the relative contribution of each feature. This is a strong method of analyzing the inner science of neural models, especially in biomedical and security fields.

Grad-CAM Spatial representations can be obtained using (Gradient-weighted Class Activation Mapping), which creates maps by utilizing a CNN's output gradients about the final convolutional layer. It also shows the parts of the input

image that have the strongest impact on the output, which can help in both understanding the model and in debugging.

Integrated Gradients (IG) is an alternative gradient-based technique that determines the average gradient based on input changes as they depart from a baseline and return to their actual values. It is a cumulative path-based strategy that shows the splitting at each feature level by its contribution to the output and the results in an aggregated and meaningful picture of the behavior of the model.

Collectively, these approaches increase the explainability of model complexity and fill the in-between gap between prospective performance and comprehensible insight. Their tailored design ensures compatibility with specific architectures, delivering meaningful and detailed explanations across both classification and regression tasks.

D. Methods for Explainable Malware Detection and Classification

Building on the prior discussion of explainable machine learning (ML) principles and post-hoc interpretability methods, this section focuses on their practical applications in malware detection and classification. The reviewed approaches are categorized based on the target system environment, including Windows PE files, hardware-level systems, PDF-based malware, and Linux-based threats. This taxonomy, illustrated in Figure 4, serves as a framework to evaluate how explainability methods adapt to diverse computing ecosystems.

By classifying explainable malware detection strategies according to system type, we underscore how the nature of executable environments influences the design and implementation of interpretability techniques. Summary tables are included for quick reference, capturing core insights from existing research, their limitations, and the explainable AI (XAI) methods used.

D.1. Windows PE-Based Malware Approaches

Due to the widespread adoption of the Windows operating system in desktop environments, Windows Portable Executable (PE) files represent a primary target for malware authors. This section examines important explainable malware detection research that is unique to PE files, arranged according to the three types of XAI methods: image-based, model-agnostic, and gradient-based. A brief comparison of these efforts is given in Table 1.

The PE file format, an extension of the Common Object File Format (COFF), begins with a legacy MS-DOS header followed by the COFF header, an optional header, and multiple section headers. These sections facilitate modular execution, supporting memory management via paging. This structured format presents unique challenges and opportunities for malware detection, particularly when developing explainable ML solutions.

Despite its importance, the field of explainable detection tailored specifically to PE files remains underexplored. Feature extraction procedures between raw binary format and traditional form can be messy and time-consuming, also. Raff et al. (Raff, Barker, Sylvester, Brandon, Catan-Zaro & Nicholas 2018) addressed this by presenting MalConv, a CNN-based model inspired by speech (Graves, Mohamed & Hinton 2013), natural language (Zhang & LeCun 2015), and image classification (Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke & Rabinovich

2015) architectures that can function on complete executable files as raw bytes. MalConv uses the architecture of an immutable analyzer, consisting of CNNs and subsequent global max pooling and dense layers. Such an arrangement will lead to position-invariance feature extraction, thus enhancing the generalizability of the malware samples. MalConv has therefore formed a baseline model to research the approach of explainable malware detection by utilizing PE files.

D.1.1. Gradient-Based Approaches

Gradient-based explainability algorithms estimate how input features can influence the model prediction by computing gradients with respect to the input. The gradients obtained will give details of which parts of a PE file influence classification the most literally pixel-by-pixel explanation.

Bose et al. (Bose, Barao & Liu 2020) studied the MalConv model in detail by performing it through the open-source framework emberMalConv. In their research, the ability to localize the patterns of interest in uncoded data by the model, with the help of identifying the byte sequences in an executable that attract most of the attention during classification, was emphasised. Ember, a framework for static PE malware detection, was used to train and evaluate model behavior.

Their gradient-based framework extended MalConv's capabilities by generating gradient embeddings and interpolating between samples. This allowed the construction of a decision boundary and the development of a correlation heatmap that illustrated how CNN filters interact. Notably, their analysis revealed that some filters specialize in detecting malicious traits, while others generalize across broader file characteristics.

This approach demonstrates the power of gradient-based techniques to provide transparency in deep models. These techniques, however, frequently call for sizable labeled datasets and are susceptible to adversarial attacks, in which changing conspicuous bytes can evade detection. Nevertheless, when fine-grained attribution is essential, gradient-based XAI proves invaluable.

Table 1. Summary of Explainable Windows PE Malware Detection Studies

Paper/Year	Focus/Objective	Contribution	Limitation	XAI Technique
Bose et al. 2020	Examined MalConv and suggested a framework for explanation based on end-to-end gradients.	To make decision boundaries more clear, gradient analysis was introduced, mapping malicious file embeddings and interpolating between correctly classified samples.	Limited interpretability of Neural Network decisions	Gradient Analysis
Mathews 2019	Identify feature-engineering flaws and global traits distinguishing malware classes	Proposed an explainability framework using content-based and PE-file statistical features, with LIME for model interpretation	Lacks comparison with other model-agnostic XAI methods	LIME
Pirch et al. 2021	Develop an explainable CNN to predict malware tags for clustering and organization	Assigns a relevance score to each feature ("token") for tagging malware, evaluated by descriptive accuracy and sparsity	Restricted to a behavioral-based approach	LIME

Marais et al. 2021	Improve model interpretability to reduce false positives	Developed a CNN-based malware model compared with LGBM, XGBoost, and DNN, using GradCam++ to highlight critical pixels for reduced false positives	Missing attention mechanisms; limited to static analysis and higher computation time	GradCam++
Li et al. 2021	Introduce an interpretable feed-forward neural network (IFFNN) for malware detection	Proposed IFFNN with high accuracy and interpretability; tested with MNIST for qualitative assessment	Interpretation fidelity requires more validation	IFFNN
Chen et al. 2019	Validate explanation fidelity in dynamic malware analysis	Extended LIME for vision-based interpretation in dynamic malware analysis, with two case studies showing interpretability benefits	Does not compare LIME with more advanced XAI methods	LIME
Lin and Chang 2021	Implement a deep ensemble detector for image-based malware with interpretable outputs	Proposed Selective Deep Ensemble Learning (SDEL) and used LIME, SHAP, and LRP to interpret image-based malware predictions	Needs more base classifiers and original malware files to locate semantic features	LIME, SHAP, LRP
Alani et al. 2023	Detect obfuscated Windows malware from memory dumps using minimal features	Achieves >99% detection using only five memory-dump features, with SHAP-based interpretability	Dependent on a limited feature set and dataset-specific performance	SHAP
Ciaramella et al. 2024	Propose a CNN-based approach for ransomware detection using image conversion	Uses Grad-CAM to visualize high-impact image regions, improving ransomware detection interpretability	Focused mainly on ransomware, with unclear generalization to other malware	Grad-CAM
Anthony et al. 2024	Adapt LENs for interpretable large-scale malware detection	Provides high-fidelity FOL rule explanations, balancing accuracy with transparency	Generating/optimizing FOL rules is computationally heavy, limiting real-time application	LENs
Gulmez et al. 2024	Provide XRAI, a CNN method with XAI support that combines API, DLL, and Mutex features to identify ransomware	Integrates API, DLL, and Mutex capabilities into a two-layer CNN, achieving 99.4% TPR with explanations from LIME and SHAP	High overhead for dynamic feature extraction	LIME, SHAP
Aryal et al. 2024	To improve evasion, use SHAP to identify and disrupt key areas of Windows PE malware	Shows how SHAP-guided perturbation increases the rates of adversarial evasion while maintaining the functionality of the virus	Restricted to certain Windows PE samples; larger malware families were not examined	SHAP
Ghadekar et al. 2024	Provide a deeper GCN model with integrated GradCAM for multi-class malware detection	Accuracy on combined byte/ASM pictures is 97%; GradCAM identifies important areas	GradCAM and real-time use may be limited by time-consuming preprocessing (byte/ASM to image)	Grad-CAM

D.1.2. Model-Agnostic Methodology

In order to better comprehend predictions without needing access to the internal workings of the model, model-agnostic post-hoc explainability techniques seek to reduce sophisticated machine learning models to more interpretable local surrogate models.

Mathews (Mathews 2019) proposed an explainability framework to classify two distinct malware families targeting Windows platforms. Using both

hexadecimal and assembly-level representations of Portable Executable (PE) files, the technique extracts statistical and content-based features. The model gains knowledge about global differences between malware categories with the use of these extracted attributes, which are representative of the PE file structure. In order to emphasize the significance of particular aspects in the classification process, the Local Interpretable Model-Agnostic Explanations (LIME) technique was used.

Pirch et al. (Pirch et al. 2021) designed an explainable CNN-based malware tagging system using a behavioral approach. Their model assigns relevance scores, referred to as "tokens," to features for clustering, family classification, and sandbox tagging. The interpretability of the model was validated using two metrics: accuracy, which assesses how well the explanation conveys the key elements, and descriptive sparsity, which aids in removing explanations that are unnecessary or redundant. The model achieved over 90% accuracy across all tagging categories.

Alani et al. (Alani et al. 2023) created a machine learning system that uses minimum features to identify obfuscated malware based on memory dumps. The model incorporates classifiers including Gaussian Naive Bayes (GNB), Random Forest (RF), logistic regression, decision trees, and Extreme Gradient Boosting (XGB), with XGB showing the best results. Features were extracted via the VolMemLyzer tool and refined through Recursive Feature Elimination (RFE), resulting in five key features: total services, average DLLs per process, mutant handles, kernel drivers, and shared services. The model had detection sensitivity of more than 99 percent and used SHapley Additive exPlanations (SHAP) to make it more interpretable. The approach is limited by its reliance on dataset-specific features and potential adversarial vulnerability, despite its relative efficiency.

In order to develop the Logic Explanation Networks (LENs), Anthony et al. (Anthony et al. 2024) focused on improving the visibility of malware detection. This is done by these networks in producing informative, interpretable First-Order Logic (FOL) rules in explaining the predictions. LEN variant adapted to the EMBER dataset obtained high classification performance and allowed generating interpretable input. LENs strike a balance between accuracy and transparency but face challenges in FOL rule generation and optimization for real-time deployment.

Gulmez et al. (Gulmez et al. 2024) introduced XRan, a ransomware detection framework that combines dynamic analysis with XAI-enabled deep learning. The system captures API call sequences, DLL access, and Mutex behavior through sandbox execution and encodes them into structured sequences for a two-layer CNN model. XRan achieved a True Positive Rate (TPR) of up to 99.4% and leveraged both LIME and SHAP to provide post-hoc interpretability. Experiments using five distinct datasets, including VirusShare, Sorel-20M, and VXHeaven, validated the robustness of the model, though it incurs a high overhead due to dynamic feature extraction.

Aryal et al. (Aryal et al. 2024) proposed a SHAP-guided adversarial attack strategy for evaluating CNN-based Windows PE malware detectors. By calculating SHAP values at the byte level, the most impactful regions of the malware were identified within the PE structure. Perturbations were then selectively injected into high-impact regions, both at the section and subsection levels, significantly increasing evasion rates while preserving functional integrity. This approach

demonstrates the utility of SHAP in adversarial scenarios, although it is constrained by a limited malware family scope.

Overall, model-agnostic frameworks such as LIME, SHAP, and LENs provide flexible and broadly applicable interpretability solutions for Windows PE malware detection. While their post-hoc nature can result in inconsistent explanations across samples, their model independence and usability make them valuable tools in practical cybersecurity applications.

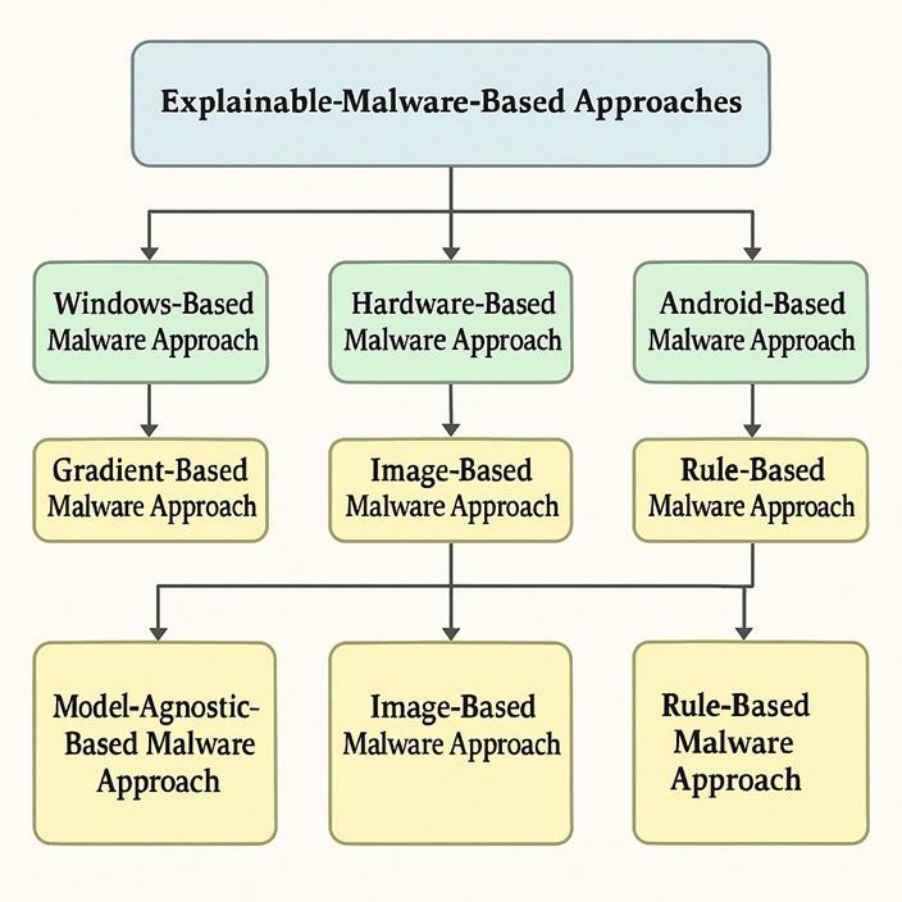

Figure 4. Target system-specific explainable malware detection techniques include Linux, Windows PE, hardware, and PDF.

E. Malware Techniques Based on Hardware

Assuming that conventional solutions, such as antivirus software, can be readily tricked by malicious code, numerous researchers have created hardware-based malware detectors. Research that takes explainability into account is still rare, despite the fact that there are many studies on hardware-assisted malware detection. This demonstrates the urgent need for innovative systems that can both detect and explain the reasoning behind their findings. Key studies on hardware-based malware categorization utilizing XAI approaches are summarized in Table 2.

Sheldon (Pan, Sheldon, Sudusinghe, Charles & Mishra 2021) used explainable machine learning (ML) techniques to evaluate hardware traces for malware detection. Hardware traces—comprising cache and register data during program

execution—are processed by ML models, and their accuracy is benchmarked against state-of-the-art classifiers. Feedback from these comparisons is then used to refine the detection process.

Pan et al. (Pan, Sheldon & Mishra 2020) introduced a comprehensive framework using hardware traces, structured into three phases: training an ML model (e.g., RNN) on traces collected via the Embedded Trace Buffer (ETB), generating predictions using a synthetic dataset X=(x_1,x_2,…,x_n), and formulating a linear predictive model via regression:

$$y = \sum_{i=1}^{n} a_i x_i + \epsilon \tag{1}$$

The goal is to minimize the error term ϵ:

$$\arg\min_{a} \|Xa - y\|^2 \tag{2}$$

Further optimization is achieved using ridge regression:

$$\arg\min_{a} \|Xa - y\|^2 + \lambda\|a\|^2 \tag{3}$$

To reduce variance, a regularization adjustment is introduced:

$$\arg\min_{a} \|(X - \lambda I)a - y\|^2 \tag{4}$$

Potentially harmful features are identified by larger coefficients in the output of the linear regression. Their method is benchmarked against the PREEMPT detector (Basu, Elnaggar, Chakrabarty, & Karri, 2019), which uses Random Forests and Decision Trees over Hardware Performance Counters (HPCs) but lacks deep feature interpretation.

In follow-up work, the same authors (Pan, Sheldon, & Mishra, 2022) used HPCs and trace buffers to identify malicious activity locations. They developed both RNN and Decision Tree models, combining interpretability with performance. Tree parsing and regression techniques help interpret the model outputs.

Li et al. (Li, Fung, Charland, & Ding 2021a) proposed the I-MAD transformer, an interpretable detector analyzing assembly code at the basic block level. It introduces:

- A deep learning model that can comprehend entire assembly instruction sequences;
- Two pre-training steps to improve understanding of assembly constructs;
- An Interpretable Feed-Forward Neural Network (IFFNN) combining the clarity of logistic regression with the depth of neural models.

In summary, HPCs and ETBs are used by hardware-based XAI solutions to provide fine-grained, runtime-level detection. Linear models and IFFNNs assist in understanding which low-level signals correspond to malicious behavior. However, these methods often require specialized hardware or instrumentation, limiting their practical deployment.

Table 2. Summary of General Malware Approaches with XAI Techniques

Paper/Year	Focus/Objective	Contribution	Limitation	XAI Technique
Chen 2018	Employ image-based static ML with LIME explanations	Visualizes malicious features in grayscale images for improved interpretability	Unverified against adversarial or obfuscated malware; lacks trust metrics	LIME
Briguglio and Saad 2019	To decipher ML-based n-gram analysis for malware detection, use LRP.	LR, RF, and NN were compared using LRP for n-gram feature relevance.	Lacks a thorough interpretability framework analysis	Layer-wise relevance propagation (LRP)
Li et al. 2021	Propose LSH-based clustering for interpretable function-level malware classification	Groups similar assembly functions, offering built-in interpretability	Potential limitations not clearly discussed	Executable functions
Fidel et al. 2020	Exploit SHAP "signatures" to detect adversarial vs. benign samples	Shows SHAP-value distributions differ for adversarial samples, enabling classifier-based detection	The non-differentiability of SHAP limits the method's ability to be readily transferred among detectors.	SHAP
Kumar and Subbiah 2022	Apply SHAP to identify top features in zero-day malware classification	uses waterfall and SHAP bar plots to examine false positives and negatives in zero-day detection.	Does not deeply explore SHAP-based misclassification causes	SHAP
Lee et al. 2022	Reliability indicators plus SHAP can be used to hone alerts about significant security risks.	Used SHAP-driven feature insights to validate the screening technique using IDS/malware datasets.	Omits comparison with alternate XAI tools	SHAP
Galli et al. 2024	Evaluate SHAP, LIME, LRP, and attention to explain LSTM/GRU in behavioral malware detection	Suggested an XAI framework for behavioral models that evaluates various explanation techniques on a range of datasets.	Unclear if results generalize beyond the tested datasets	SHAP, LIME, LRP, and attention mechanisms

F. Desktop Malware: Threats, Features, and Explainability Considerations

F.1. Desktop Malware Landscape and Threat Taxonomy

Desktop-focused malware encompasses a diverse set of families with distinct operational goals and tactics. Common categories include ransomware (data exfiltration and encryption with extortion), information stealers (credential, browser, and wallet harvesting), remote access trojans (RATs) (persistent command-and-control with screen/keystroke capture), loaders/droppers (staged delivery of secondary payloads), wipers (destructive overwrites), fileless/LOLBins abuse (living-off-the-land via signed system tools), cryptominers (resource hijacking), and adware/PUAs. Across these families, observable behaviors on endpoints—process creation chains, API/system-call sequences, memory allocation patterns, registry and service manipulation, file I/O bursts, and network beacons—provide discriminative signals for learning-based detectors that operate on static and dynamic features (Damodaran et al. 2017, Kakisim et al. 2019, Firdausi et al. 2010, Luckett et al. 2016, Pirscoveanu et al. 2015, Tobiyama et al. 2016). Real-time, system-wide telemetry in online detection further captures stealthy or low-profile threats and supports sequence-aware modeling of attack tactics and techniques (Watson et al. 2016, Azmandian et al. 2011, Abdelsalam et al. 2017, McDole et al. 2020, McDole et al. 2021, Kimmell et al. 2021, Kimmel et al. 2021, Abdelsalam et al. 2018).

Platform specifics shape the feature space and the attack surface. On Windows desktops, Portable Executable (PE) structure, import tables, section layouts, and raw byte streams enable both static and byte-level image representations that feed modern classifiers such as MalConv and its derivatives (Raff et al. 2018, Bose et al. 2020). Obfuscation, packing, and anti-analysis may distort these artifacts, pushing detectors toward behavior-centric features gathered during sandboxed execution or endpoint monitoring (Kakisim et al. 2019, Pirscoveanu et al. 2015, Tobiyama et al. 2016, McDole et al. 2020, McDole et al. 2021). These realities elevate the role of explainability: security teams require faithful, actionable rationales that expose which structural fields, byte regions, or runtime events influenced a verdict, and why similar samples diverge. Post-hoc, model-agnostic tools (e.g., LIME/SHAP) and gradient-based analyses provide local/global attributions and saliency over bytes, APIs, and traces, improving trust, triage, and error analysis (Islam et al. 2021, Speith 2022, Arrieta et al. 2020, Blanco-Justicia & Domingo-Ferrer 2019, Ribeiro et al. 2016, Lundberg & Lee 2017, Bose et al. 2020). In this survey's taxonomy, desktop malware therefore serves as a unifying context where model performance and operational transparency must co-exist: opaque detectors benefit from consistent, high-fidelity explanations, while transparent baselines remain valuable references for auditing and drift monitoring (Linardatos et al. 2020, Arrieta et al. 2020, Blanco-Justicia & Domingo-Ferrer 2019).

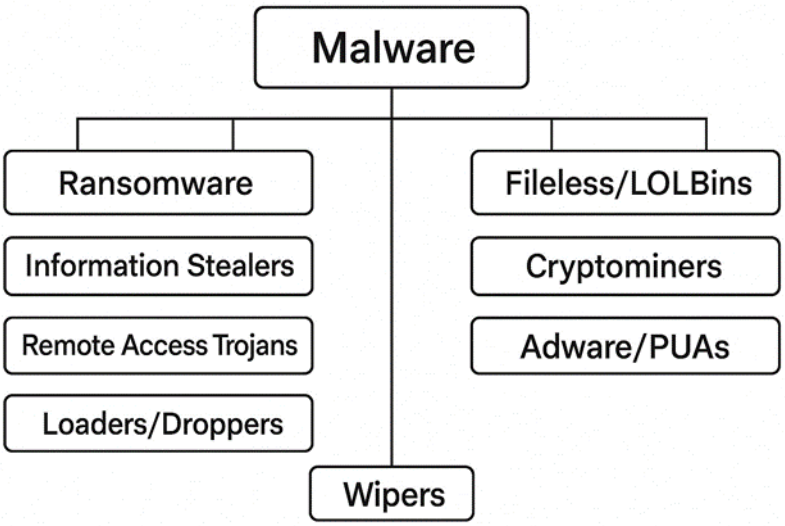

Figure 5. Taxonomy of desktop malware families, highlighting categories and functional attack objectives.

As shown in Figure 5, desktop malware can be classified into multiple categories, including ransomware, information stealers, remote access trojans (RATs), loaders/droppers, wipers, fileless malware leveraging LOLBins, cryptominers, and adware/PUAs. This taxonomy highlights the diversity of threats that target desktop platforms, each with distinct operational goals, persistence mechanisms, and detection challenges, thereby underscoring the need for explainable AI approaches tailored to their unique behaviors.

F.2. Executable Formats and Feature Spaces on Desktops

Executable file formats form the structural foundation of desktop malware and define the available feature spaces for analysis. On Windows systems, the Portable Executable (PE) format dominates, offering rich metadata such as import/export tables, section headers, relocation information, and raw byte streams. These attributes are widely used for both static and image-based representations, enabling models such as MalConv to operate directly on byte sequences (Raff et al. 2018, Bose et al. 2020). Linux malware primarily relies on the Executable and Linkable Format (ELF), while macOS threats leverage the Mach-O format. Each format provides unique opportunities for feature extraction but also presents distinct obfuscation and packing challenges.

Feature engineering approaches for desktop malware analysis can be broadly grouped into several categories. At the static level, opcode sequences, n-grams, API call references, and entropy windows from packed/unpacked binaries have demonstrated discriminative power (Damodaran et al. 2017, Hou et al. 2017, Kim & Lee 2021, Shankarapani et al. 2011). Dynamic analysis expands the feature space by capturing runtime traces such as system calls, memory allocation patterns, and network communications, thereby improving resilience against obfuscation (Kakisim et al. 2019, Firdausi et al. 2010, Luckett et al. 2016, Pirscoveanu et al. 2015, Tobiyama et al. 2016). Hybrid approaches combine these views to increase robustness by integrating static PE or ELF header features with behavioral logs.

More recent efforts explore byte-to-image transformations, control-flow graph (CFG) embeddings, and memory-dump attributes to support deep learning classifiers with higher capacity for complex pattern recognition (McDole et al. 2020, McDole et al. 2021, Kimmell et al. 2021, Kimmel et al. 2021, Abdelsalam et al. 2018). Collectively, these diverse feature spaces highlight the importance of explainability in clarifying which attributes drive predictions, whether structural headers, API sequences, or runtime behaviors.

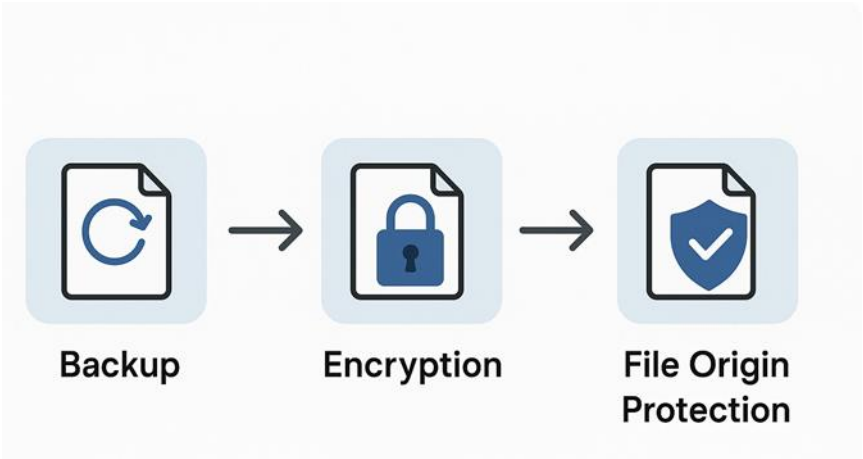

Figure 6. Workflow of desktop file handling, highlighting backup, encryption, and protection mechanisms.

Figure 6 illustrates a simplified workflow of desktop file security. The process begins with file Backup, ensuring data availability in case of corruption or loss. Next, Encryption is applied to safeguard file contents from unauthorized access. Finally, File Origin Protection ensures authenticity and integrity, preventing tampering and validating trust in executable files. Together, these stages emphasize the layered defense mechanisms necessary for protecting desktop systems from evolving malware threats.

F.3. Human-Centered Explainability in Desktop Malware Detection

While advanced malware detectors for desktop environments achieve high accuracy, their adoption in operational settings often depends on the clarity of their explanations. Desktop-focused threats frequently employ obfuscation, polymorphism, or fileless execution strategies that obscure traditional features, making post-hoc interpretability indispensable for analysts. Explainable Artificial Intelligence (XAI) techniques offer the ability to transform complex detections into actionable insights, supporting faster triage, error analysis, and informed response.

Local and global XAI approaches have been increasingly applied to desktop malware classifiers. Model-agnostic tools such as LIME and SHAP highlight which byte regions, PE header fields, or API calls most influenced a classification decision, enabling analysts to validate whether the model focuses on semantically meaningful attributes rather than spurious correlations (Ribeiro et al. 2016, Lundberg & Lee 2017, Bose et al. 2020, Alani et al. 2023, Gulmez et al. 2024). Visual explanations like Grad-CAM and Layer-wise Relevance Propagation (LRP) provide saliency maps over raw binaries or byte-to-image representations, illustrating

which regions of executable code or dynamic traces contribute to a malicious verdict (Wang et al. 2021, Shrikumar et al. 2017, Bose et al. 2020). Logic Explained Networks (LENs) further enhance transparency by producing interpretable rule sets that approximate underlying deep models (Anthony et al. 2024).

In desktop environments, these explainability methods serve multiple operational roles. They reduce false positives by clarifying benign behaviors mistaken for malicious ones, exposing the effects of evasion tactics such as packing or adversarial perturbations, and guiding reverse engineering by pointing directly to suspicious byte regions or API sequences. Integrating XAI explanations into endpoint detection and response (EDR) dashboards allows security analysts to interactively explore model rationales, improving both trust and decision-making. Ultimately, embedding human-centered explainability into desktop malware detection aligns technical advances with practical cybersecurity needs, ensuring that detection systems are not only accurate but also transparent, trustworthy, and resilient against adversarial threats.

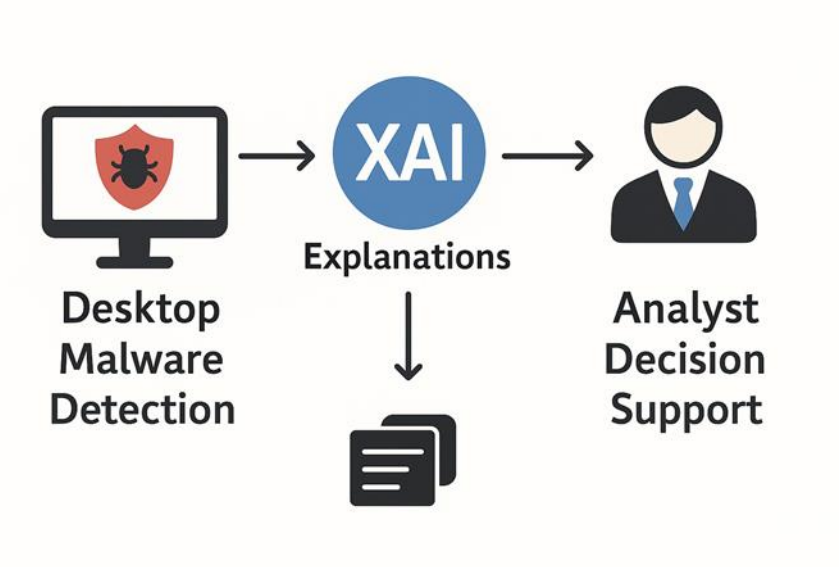

Figure 7. Explainable desktop malware detection pipeline linking XAI explanations with analyst decisions.

Figure 7 presents a conceptual workflow where desktop malware detection models generate outputs that are enriched with XAI-based explanations. These explanations highlight relevant features or behaviors, providing clarity into why a file or process was classified as malicious. The output is then passed to security analysts, who use the explanations to validate results, investigate uncertain cases, and make better-informed operational decisions. By connecting model predictions with interpretable insights, this framework bridges the gap between automated detection and human-centered cybersecurity analysis.

G. Future Directions

Despite the considerable advancements in integrating explainability into malware detection systems, several research gaps and challenges persist. These limitations present valuable opportunities for future exploration and innovation in this rapidly evolving field. This section outlines potential future research

directions that can enhance both the effectiveness and transparency of explainable malware detection.

G.1. Real-Time and Resource-Efficient Explainability

Numerous post-hoc XAI methods, such as SHAP and LIME, need a lot of processing and are not appropriate for real-time implementation in environments with constraints, like embedded systems or edge devices. Future research should focus on designing lightweight, real-time explainability models that maintain fidelity without compromising on detection speed or accuracy.

G.2. Unified Benchmarks and Evaluation Protocols

There is currently no standardized benchmark for evaluating explainability in the malware detection context. The development of unified evaluation metrics, interpretability scores, and public benchmark datasets—including adversarial examples—would facilitate fair comparisons and accelerate progress in the field. Such standards should also capture trustworthiness, robustness, and user satisfaction.

G.3. Adversarial Robustness of XAI Models

Recent studies suggest that XAI explanations themselves can be manipulated or fooled by adversarial examples, potentially undermining trust in security-critical applications. Future work must address the adversarial robustness of explainability techniques, ensuring that explanations remain consistent and reliable under attack conditions.

G.4. Integration with Cybersecurity Operations

Explainability outputs need to be seamlessly integrated into the workflows of security analysts, SOCs (Security Operations Centers), and threat intelligence platforms. Human-centered research is required to understand how analysts interpret and act on explanations. This includes designing interactive dashboards, visualization tools, and recommendation systems that translate complex model outputs into actionable insights.

G.5. Combining Causal Inference with Explainability

While current explainability techniques provide associative insights (e.g., feature importance), they often lack causal reasoning. Integrating causal inference frameworks can help identify not just correlations, but underlying causes of malware behavior, offering more meaningful interpretations and stronger generalization to unseen threats.

G.6. Federated and Privacy-Preserving XAI for Malware

As privacy concerns grow, federated learning has emerged as a privacy-aware alternative to centralized training. However, applying XAI in federated malware detection systems remains an open challenge. Future research could explore decentralized explainability mechanisms that ensure transparency without violating user or organizational data privacy.

G.7. Human-in-the-Loop Explainable Malware Detection

Finally, empowering end-users and domain experts to interactively guide or refine machine learning explanations can improve trust, correctness, and relevance. Incorporating feedback loops into explainable malware detection models—where human insights enhance system performance—represents a promising interdisciplinary direction combining cybersecurity, HCI, and AI.

In summary, explainability in malware detection is still in its formative stages, with vast potential for growth. By addressing the above challenges, future work can contribute toward building trustworthy, efficient, and broadly applicable explainable AI systems that enhance malware detection and cybersecurity at large.

H. Conclusion

As cyber threats grow in complexity and stealth, the need for robust, intelligent malware detection systems has never been more urgent. Although models for machine learning and deep learning have shown remarkable skills to detect both known and unknown malware, their opaqueness frequently prevents them from being widely used in actual security operations. This paper has presented a comprehensive survey of Explainable Artificial Intelligence (XAI) techniques applied to malware classification and detection, highlighting their role in bridging the gap between model accuracy and interpretability.

We proposed a structured taxonomy categorizing explainability approaches based on model transparency, post-hoc techniques (model-agnostic and model-specific), and target system environments, including Windows PE files, PDF-based malware, Linux systems, and hardware-level detectors. Through this lens, we reviewed a wide array of studies, outlined their contributions and drawbacks, and provided examples of how XAI techniques like Logic Explained Networks, Grad-CAM, LIME, and SHAP are used to decipher intricate model decisions.

Despite notable progress, several challenges remain, including the lack of standard evaluation frameworks, high computational costs of some explainability methods, limited studies beyond Windows PE, and vulnerability to adversarial manipulation. As discussed in our future directions, addressing these gaps is essential for advancing the field toward more transparent, trustworthy, and operationally effective malware detection solutions.

In the end, adding explainability to malware detection systems is not just a technological improvement; it is also necessary to establish credibility, make human-centered security analysis possible, and encourage adoption in vital cybersecurity infrastructure. As the landscape of threats continues to evolve, so too must our models—and the explanations that make them understandable.

I. References

[1] Abdelsalam, M., Krishnan, R., Huang, Y. & Sandhu, R. (2018), Malware detection in cloud infrastructures using convolutional neural networks, in 'Proc. IEEE 11th Int. Conf. Cloud Computing (CLOUD)', pp. 162–169.

[2] Abdelsalam, M., Krishnan, R. & Sandhu, R. (2017), Clustering-based iaas cloud monitoring, in

[3] 'Proc. IEEE 10th Int. Conf. Cloud Computing (CLOUD)', pp. 672–679.

[4] Adebayo, J. & Kagal, L. (2016), 'Iterative orthogonal feature projection for diagnosing bias in black-box models'.
[5] Alani, M. M., Mashatan, A. & Miri, A. (2023), 'Xmal: A lightweight memory-based explainable obfuscated-malware detector', Computers & Security 133, 103409.
[6] Anthony, P., Giannini, F., Diligenti, M., Homola, M., Gori, M., Balogh, S. & Mojzis, J. (2024), 'Explainable malware detection with tailored logic explained networks'.
[7] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R. & Herrera, F. (2020), 'Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai', Information Fusion 58, 82–115.
[8] Aryal, K., Gupta, M., Abdelsalam, M. & Saleh, M. (2024), 'Explainability guided adversarial evasion attacks on malware detectors', arXiv preprint.
[9] Aslam, N., Khan, I. U., Aljishi, R. F., Alnamer, Z. M., Alzawad, Z. M., Almomen, F. A. & Alramadan, F. A. (2022), 'Explainable computational intelligence model for antepartum fetal monitoring to predict the risk of iugr', Electronics 11(4), 593.
[10] Athanasiou, M., Sfrintzeri, K., Zarkogianni, K., Thanopoulou, A. C. & Nikita, K. S. (2020), An explainable XGBoost–based approach towards assessing the risk of cardiovascular disease in patients with type 2 diabetes mellitus, in 'Proc. IEEE 20th Int. Conf. Bioinf. Bioengineering (BIBE)', pp. 859–864.
[11] Azmandian, F., Moffie, M., Alshawabkeh, M., Dy, J., Aslam, J. & Kaeli, D. (2011), 'Virtual machine monitor-based lightweight intrusion detection', ACM SIGOPS Operating Systems Review 45(2), 38–53.
[12] Basu, K., Elnaggar, R., Chakrabarty, K. & Karri, R. (2019), Preempt: Preempting malware by examining embedded processor traces, in 'Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)', pp. 1–6.
[13] Blanco-Justicia, A. & Domingo-Ferrer, J. (2019), Machine learning explainability through comprehensible decision trees, in 'Proc. Int. Cross-Domain Conf. Machine Learning and Knowledge Extraction, Springer, Cham, Switzerland, pp. 15–26.
[14] Bose, S., Barao, T. & Liu, X. (2020), Explaining ai for malware detection: Analysis of mechanisms of malconv, in 'Proc. Int. Joint Conf. Neural Netw. (IJCNN)', pp. 1–8.
[15] Briguglio, W. & Saad, S. (2020), Interpreting machine learning malware detectors which leverage n-gram analysis, in 'Proc. Int. Symp. Found. Pract. Secur.', pp. 82–97.
[16] Chamola, V., Hassija, V., Sulthana, A. R., Ghosh, D., Dhingra, D. & Sikdar, B. (2023), 'A review of trustworthy and explainable artificial intelligence (xai)', IEEE Access 11, 78994–79015.
[17] Charmet, F., Tanuwidjaja, H. C., Ayoubi, S., Gimenez, P.-F., Han, Y., Jmila, H., Blanc, G., Takahashi, T. & Zhang, Z. (2022), 'Explainable artificial intelligence for cybersecurity: A literature survey', Annals of Telecommunications 77(11–12), 789–812.

[18] Chen, L. (2018), 'Deep transfer learning for static malware classification', arXiv preprint .
[19] Chen, L., Yagemann, C. & Downing, E. (2019), To believe or not to believe: Validating explanation fidelity for dynamic malware analysis, in 'Proc. CVPR Workshops', pp. 48–52.
[20] Ciaramella, G., Iadarola, G., Martinelli, F., Mercaldo, F. & Santone, A. (2023), 'Explainable ransomware detection with deep learning techniques', Journal of Computer Virology and Hacking Techniques 20(2), 317–330.
[21] Ciravegna, G., Barbiero, P., Giannini, F., Gori, M., Lió, P., Maggini, M. & Melacci, S. (2023), 'Logic explained networks', Artificial Intelligence 314, 103822.
[22] Confalonieri, R., Coba, L., Wagner, B. & Besold, T. R. (2021), 'A historical perspective of explainable artificial intelligence', WIREs Data Mining and Knowledge Discovery 11(1), e1391.
[23] Damodaran, A., Di Troia, F., Visaggio, C. A., Austin, T. H. & Stamp, M. (2017), 'A comparison of static, dynamic, and hybrid analysis for malware detection', Journal of Computer Virology and Hacking Techniques 13(1), 1–12.
[24] Danilevsky, M., Qian, K., Aharonov, R., Katsis, Y., Kawas, B. & Sen, P. (2020), 'A survey of the state of explainable ai for natural language processing'.
[25] Das, A. & Rad, P. (2020), 'Opportunities and challenges in explainable artificial intelligence (xai): A survey'.
[26] Davagdorj, K., Bae, J.-W., Pham, V.-H., Theera-Umpon, N. & Ryu, K. H. (2021), 'Explainable artificial intelligence based framework for non-communicable diseases prediction', IEEE Access 9, 123672–123688.
[27] Dwivedi, R., Dave, D., Naik, H., Singhal, S., Omer, R., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G. & Ranjan, R. (2023), 'Explainable ai (xai): Core ideas, techniques, and solutions', ACM Computing Surveys 55(9), 1–33.
[28] Fidel, G., Bitton, R. & Shabtai, A. (2020), When explainability meets adversarial learning: Detecting adversarial examples using shap signatures, in 'Proc. Int. Joint Conf. Neural Netw. (IJCNN)', pp. 1–8.
[29] Firdausi, I., Lim, C., Erwin, A. & Nugroho, A. S. (2010), Analysis of machine learning techniques used in behavior-based malware detection, in 'Proc. 2nd Int. Conf. Adv. Comput., Control, Telecommun. Technol.', pp. 201–203.
[30] Friedman, J. H. & Popescu, B. E. (2008), 'Predictive learning via rule ensembles', Annals of Applied Statistics 2(3), 1–12.
[31] Galli, A., La Gatta, V., Moscato, V., Postiglione, M. & Sperlì, G. (2024), 'Explainability in ai-based behavioral malware detection systems', Computers & Security 141. Art. no. 103842.
[32] Ghadekar, P., Adsare, T., Agrawal, N., Deore, D. & Dharmik, T. (2024), Multiclass malware detection using modified gnn and explainable ai, in 'Proc. 1st Int. Conf. Cognit., Green Ubiquitous Comput. (IC-CGU)', pp. 1–8.
[33] Goldstein, A., Kapelner, A., Bleich, J. & Pitkin, E. (2015), 'Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation', Journal of Computational and Graphical Statistics 24(1), 44–65.
[34] Graves, A., Mohamed, A.-R. & Hinton, G. (2013), Speech recognition with deep recurrent neural networks, in 'Proc. IEEE Int. Conf. Acoust., Speech Signal Process.', pp. 6645–6649.

[35] Gulmez, S., Kakisim, A. G. & Sogukpinar, I. (2024), 'Xran: Explainable deep learning-based ransomware detection using dynamic analysis', Computers & Security 139, 103703.
[36] Hassija, V., Chamola, V., Mahapatra, A., Singal, A., Goel, D., Huang, K., Scardapane, S., Spinelli, I., Mahmud, M. & Hussain, A. (2024), 'Interpreting black-box models: A review on explainable artificial intelligence', Cognitive Computation 16(1), 45–74.
[37] Hou, S., Saas, A., Chen, L., Ye, Y. & Bourlai, T. (2017), Deep neural networks for automatic android malware detection, in 'Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)', pp. 803–810.
[38] Islam, S. R., Eberle, W., Ghafoor, S. K. & Ahmed, M. (2021), 'Explainable artificial intelligence approaches: A survey'.
[39] Kakisim, A. G., Nar, M., Carkaci, N. & Sogukpinar, I. (2019), Analysis and evaluation of dynamic feature-based malware detection methods, in 'Proc. Int. Conf. Secur. Inf. Technol. Commun.', pp. 247–258.
[40] Kim, H. & Lee, T. (2021), Research on autoencoder technology for malware feature purification, in 'Proc. 21st ACIS Int. Winter Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD-Winter)', pp. 236–239.
[41] Kimmel, J. C., McDole, A. D., Abdelsalam, M., Gupta, M. & Sandhu, R. (2021), 'Recurrent neural networks based online behavioural malware detection techniques for cloud infrastructure', IEEE Access 9, 68066–68080.
[42] Kimmell, J. C., Abdelsalam, M. & Gupta, M. (2021), 'Analyzing machine learning approaches for online malware detection in cloud'.
[43] Krämer, B., Stang, M., Nagl, C. & Schäfers, W. (2023), 'Explainable ai in a real estate context—exploring the determinants of residential real estate values', Journal of Housing Research 32(2), 204–245.
[44] Kumar, R. & Subbiah, G. (2022), 'Zero-day malware detection and effective malware analysis using shapley ensemble boosting and bagging approach', Sensors 22(7), 2798.
[45] Lee, E., Lee, Y. & Lee, T. (2022), 'Automatic false alarm detection based on xai and reliability analysis', Applied Sciences 12(13), 6761.
[46] Li, M. Q., Fung, B. C. M. & Abusitta, A. (2021), 'On the effectiveness of interpretable feedforward neural network'.
[47] Li, M. Q., Fung, B. C. M., Charland, P. & Ding, S. H. H. (2021a), 'I-mad: Interpretable malware detector using galaxy transformer', Computers & Security 108, 102371.
[48] Li, M. Q., Fung, B. C. M., Charland, P. & Ding, S. H. H. (2021b), A novel and dedicated machine learning model for malware classification, in 'Proc. 16th Int. Conf. Softw. Technol.', pp. 617– 628.
[49] Li, X.-H., Cao, C. C., Shi, Y., Bai, W., Gao, H., Qiu, L., Wang, C., Gao, Y., Zhang, S., Xue,
[50] X. & Chen, L. (2022), 'A survey of data-driven and knowledge-aware explainable ai', IEEE Transactions on Knowledge and Data Engineering 34(1), 29–49.
[51] Lin, Y. & Chang, X. (2021a), 'Towards interpretable ensemble learning for image-based malware detection'.

[52] Lin, Y. & Chang, X. (2021b), 'Towards interpreting ml-based automated malware detection models: A survey'.
[53] Linardatos, P., Papastefanopoulos, V. & Kotsiantis, S. (2020), 'Explainable ai: A review of machine learning interpretability methods', Entropy 23(1), 18.
[54] Luckett, P., McDonald, J. T. & Dawson, J. (2016), Neural network analysis of system call timing for rootkit detection, in 'Proc. Cybersecurity Symposium (CYBERSEC)', pp. 1–6.
[55] Lundberg, S. & Lee, S. (2017), A unified approach to interpreting model predictions, in 'Proc. 31st Int. Conf. Neural Inf. Process. Syst.', pp. 4768–4777.
[56] Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N. & Lee, S.-I. (2019), 'Explainable ai for trees: From local explanations to global understanding'.
[57] Magesh, P. R., Myloth, R. D. & Tom, R. J. (2020), 'An explainable machine learning model for early detection of parkinson's disease using lime on datscan imagery', Computers in Biology and Medicine 126, 104041.
[58] Marais, B., Quertier, T. & Chesneau, C. (2021), Malware analysis with artificial intelligence and a particular attention on results interpretability, in 'Proc. Int. Symp. Distrib. Comput. Artif. Intell.', pp. 43–55.
[59] Mathews, S. M. (2019), Explainable artificial intelligence applications in nlp, biomedical, and malware classification: A literature review, in 'Proc. Intell. Comput. Comput. Conf.', pp. 1269– 1292.
[60] McDole, A., Abdelsalam, M., Gupta, M. & Mittal, S. (2020), Analyzing cnn based behavioural malware detection techniques on cloud iaas, in 'Proc. Int. Conf. Cloud Computing', pp. 64–79.
[61] McDole, A., Gupta, M., Abdelsalam, M., Mittal, S. & Alazab, M. (2021), Deep learning techniques for behavioral malware analysis in cloud iaas, in 'Malware Analysis Using Artificial Intelligence and Deep Learning', Springer, Cham, Switzerland, pp. 269–285.
[62] Milani, S., Topin, N., Veloso, M. & Fang, F. (2024), 'Explainable reinforcement learning: A survey and comparative review', ACM Computing Surveys 56(7), 1–36.
[63] Mohseni, S., Zarei, N. & Ragan, E. D. (2021), 'A multidisciplinary survey and framework for design and evaluation of explainable ai systems', ACM Transactions on Interactive Intelligent Systems 11(3–4), 1–45.
[64] Molnar, C. (2022), Interpretable Machine Learning, 2nd edn, Christoph Molnar. https:// christophm.github.io/interpretable-ml-book/.
[65] Nasser, Y. & Nassar, M. (2023), 'Toward hardware-assisted malware detection utilizing explainable machine learning: A survey', IEEE Access 11, 131273–131288.
[66] Pan, Z., Sheldon, J. & Mishra, P. (2020), Hardware-assisted malware detection using explainable machine learning, in 'Proc. IEEE 38th Int. Conf. Comput. Design (ICCD)', pp. 663–666.
[67] Pan, Z., Sheldon, J. & Mishra, P. (2022), 'Hardware-assisted malware detection and localization using explainable machine learning', IEEE Transactions on Computers 71(12), 3308–3321.

[68] Pan, Z., Sheldon, J., Sudusinghe, C., Charles, S. & Mishra, P. (2021), Hardware-assisted malware detection using machine learning, in 'Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)', pp. 1775–1780.
[69] Pirch, L., Warnecke, A., Wressnegger, C. & Rieck, K. (2021), Tagvet: Vetting malware tags using explainable machine learning, in 'Proc. 14th Eur. Workshop Syst. Secur.', pp. 34–40.
[70] Pirscoveanu, R. S., Hansen, S. S., Larsen, T. M. T., Stevanovic, M., Pedersen, J. M. & Czech, A. (2015), Analysis of malware behavior: Type classification using machine learning, in 'Proc. Int. Conf. Cyber Situational Awareness, Data Anal. Assessment (CyberSA)', pp. 1–7.
[71] Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B. & Nicholas, C. K. (2018), Malware detection by eating a whole exe, in 'Proc. Workshops 32nd AAAI Conf. Artif. Intell.', pp. 1–10.
[72] Rana, M. S., Gudla, C. & Sung, A. H. (2018), Evaluating machine learning models for android malware detection: A comparison study, in 'Proc. 7th Int. Conf. Netw., Commun. Comput.', pp. 17–21.
[73] Ribeiro, M., Singh, S. & Guestrin, C. (2016), "why should i trust you?" explaining the predictions of any classifier, in 'Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Demonstrations, pp. 1135–1144.
[74] Roscher, R., Bohn, B., Duarte, M. F. & Garcke, J. (2020), 'Explainable machine learning for scientific insights and discoveries', IEEE Access 8, 42200–42216.
[75] Saeed, W. & Omlin, C. (2023), 'Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities', Knowledge-Based Systems 263, 110273.
[76] Sahoh, B. & Choksuriwong, A. (2023), 'The role of explainable artificial intelligence in high-stakes decision-making systems: a systematic review', Journal of Ambient Intelligence and Humanized Computing 14, 7827–7843.
[77] Santoso, I., Heryadi, Y., Warnars, H. L. H. S., Wulandhari, L. A., Lukas & Abdurachman, E. (2019), Malware detection using hybrid autoencoder approach for better security in educational institutions, in 'Proc. IEEE Int. Conf. Eng., Technol. Educ. (TALE)', pp. 1–6.
[78] Saranya, A. & Subhashini, R. (2023), 'A systematic review of explainable artificial intelligence models and applications: Recent developments and future trends', Decision Analytics Journal 7, 100230.
[79] Shankarapani, M. K., Ramamoorthy, S., Movva, R. S. & Mukkamala, S. (2011), 'Malware detection using assembly and api call sequences', Journal of Computer Virology 7(2), 107–119.
[80] Sharma, R., Schommer, C. & Vivarelli, N. (2020), Building up explainability in multi-layer perceptrons for credit risk modeling, in 'Proc. IEEE 7th Int. Conf. Data Sci. Adv. Anal. (DSAA)', pp. 761–762.
[81] Shrikumar, A., Greenside, P. & Kundaje, A. (2017), Learning important features through propagating activation differences, in 'Proc. Int. Conf. Mach. Learn.', pp. 3145–3153.
[82] Soares, E., Angelov, P. P., Costa, B., Castro, M. P. G., Nageshrao, S. & Filev, D. (2021), 'Explaining deep learning models through rule-based approximation and visualization', IEEE Transactions on Fuzzy Systems 29(8), 2399–2407.

[83] Speith, T. (2022), A review of taxonomies of explainable artificial intelligence (xai) methods, in
[84] 'Proc. ACM Conf. Fairness, Accountability, and Transparency, pp. 2239–2250.
[85] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, in 'Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)', pp. 1–9.
[86] Tobiyama, S., Yamaguchi, Y., Shimada, H., Ikuse, T. & Yagi, T. (2016), Malware detection with deep neural network using process behavior, in 'Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)', Vol. 2, pp. 577–582.
[87] Tong, F. & Yan, Z. (2017), 'A hybrid approach of mobile malware detection in android', Journal of Parallel and Distributed Computing 103, 22–31.
[88] Tzermias, Z., Sykiotakis, G., Polychronakis, M. & Markatos, E. P. (2011), Combining static and dynamic analysis for the detection of malicious documents, in 'Proc. 4th Eur. Workshop Syst. Secur.', pp. 1–6.
[89] Vieira, C. P. R. & Digiampietri, L. A. (2020), 'A study about explainable artificial intelligence: Using decision tree to explain svm', Revista Brasileira de Computação Aplicada 12(1), 113– 121.
[90] Wang, H., Zhu, Z., Tong, Z., Yin, X., Feng, Y., Shi, G. & Meng, D. (2021), An effective approach for malware detection and explanation via deep learning analysis, in 'Proc. Int. Joint Conf. Neural Netw. (IJCNN)', pp. 1–10.
[91] Watson, M. R., Shirazi, N.-U.-H., Marnerides, A. K., Mauthe, A. & Hutchison, D. (2016), 'Malware detection in cloud computing infrastructures', IEEE Transactions on Dependable and Secure Computing 13(2), 192–205.
[92] Weber, L., Lapuschkin, S., Binder, A. & Samek, W. (2023), 'Beyond explaining: Opportunities and challenges of xai-based model improvement', Information Fusion 92, 154–176.
[93] Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D. & Zhu, J. (2019), Explainable ai: A brief survey on history, research areas, approaches and challenges, in 'Proc. CCF Int. Conf. Natural Lang. Process. Chin. Comput.', Springer, Cham, Switzerland, pp. 563–574.
[94] Yang, Z., Zhang, A. & Sudjianto, A. (2021), 'Gami-net: An explainable neural network based on generalized additive models with structured interactions', Pattern Recognition 120, 108192.
[95] Zhang, X. & LeCun, Y. (2015), 'Text understanding from scratch'.
[96] Zhao, X., Wu, Y., Lee, D. L. & Cui, W. (2019), 'iforest: Interpreting random forests via visual analytics', IEEE Transactions on Visualization and Computer Graphics 25(1), 407–416.