A SYSTEMATIC REVIEW OF OBFUSCATED MALWARE DETECTION: FROM TRADITIONAL ANALYSIS TO DEEP LEARNING

Mohammed bin Shamlan [1] , Mohammed Fadhl Abdullah [1] , Khaled Hassan Balhaf [1] , Ahmed Saleh Khaled [1] , Makarem Mohamed Bamatraf [2]
[1] Faculty of computing and engineering, university of science and technology, Aden, Yemen.
[2] Computer Engineering Department, College of Engineering, Hadhramaut University, Yemen

ABSTRACT

Obfuscation has been increasingly difficult in the subject of cybersecurity, since malware developers use it to change code appearance without changing its malicious behavior. As a result, signature-based and basic heuristic detection systems are easily bypassed by these techniques. This article reviews recent and ongoing research in the analysis and detection of obfuscated malware, giving special attention to methods that were recently developed to address this problem. The reviewed methods are divided into five major classes: static analysis, dynamic analysis, hybrid analysis, machine learning, and deep learning. thirty-six recent research papers from 2018 to 2025 are analyzed, with a detailed summary of each, including merits and demerits. The review is intended to generate a broad picture of the research field, point out strengths and weaknesses in each category, and identify the way forward, especially for the area of hybrid and deep learning-oriented memory analysis.

KEYWORDS

Obfuscated Malware, Static and Dynamic Analysis, Malware Detection, Memory Analysis, Cybersecurity, Explainable Artificial Intelligence.

1. INTRODUCTION

Obfuscation means a continuous and growing challenge in the cyberspace spectrum-for malware developers employ it to thoroughly change the appearance of code while still maintaining its malicious essence, thus evading traditional signature-based or simple heuristic detection systems. The technique intends to impede the reverse engineering and analysis of the executables, and these techniques include string encryption, control flow obfuscation, packing, and polymorphism. Continuing changes in obfuscation methods call for corresponding responses from the security research community, which in turn has led to an elevation in the specification of the analytical tools employed. Malware detection in its infancy was based on static analysis; that is, the examination of code without any execution. Whenever employed, obfuscation proved efficaciously countering it: the birth of dynamic analysis dredged forth with its monitoring program behavior in an isolated environment-sandbox. Dynamic analysis has proved successful against the obfuscation. Unfortunately, it falls prey to sand-box evasion and anti-analysis techniques. So came about the compromise called hybrid analysis to counteract the weaknesses of both techniques, in that it merges the quickness of static with the accuracy of dynamic analysis. Within the last few years, the application of machine learning (ML) and deep learning (DL) techniques in this area has gained momentous momentum, being in a position to extract intricate

features and behavioral patterns from the data so as to catch the new variants of obfuscated malware. The paper discusses recent and updated research on analyzing and detecting obfuscated malware with a strong focus on methodological approaches that have arisen in contemporary years in an attempt to tackle this difficulty. The reviewed methodologies are classified into five main categories: static analysis, dynamic analysis, hybrid analysis, machine learning and deep learning, and auxiliary analysis (memory and network). The review intends to provide a holistic coverage of the research landscape and indicate the strengths and weaknesses of each category, along with identifying the promising avenues for research, especially in hybrid and deep learning models based on memory analysis. The review gives an in-depth and structured analysis of ways of detecting obfuscated malware using its various analysis dimensions unlike the old survey articles that examined one analysis paradigm or a particular malware family. The paper has made four contributions: (1) It evangelizes a single taxonomy that categorizes the obfuscated malware detection methods into five categories, depending on principles of their methods, which are: static detection, dynamic detection, hybrid detection, machine learning detection techniques, and auxiliary analysis technique, outlining the core differences and similarities of such methods. (2) It performs a comprehensive comparison of the modern works that were published in 2018-2025 and indicate the influence of several forms of obfuscation in different analysis processes on detection rates. (3) The review critically examines the limitations and strengths of each category while focusing on advanced evasion techniques, which include packing, virtualization, and fileless malware. (4) It presents open research challenges and future research directions which show how memory forensics and explainable artificial intelligence, and adaptive learning models will help fight against advanced obfuscated malware threats.

1.1. Systematic Literature Selection Methodology

The researchers selected primary studies for their analysis through a systematic method, which ensured both rigorous assessment and representative results. The methodology implements PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) guidelines through its four established research stages:

1. **Identification**: To retrieve the literature on the topic, major academic databases were searched, namely IEEE Xplore, ScienceDirect, ACM Digital Library, MDPI, and SpringerLink, with the search query looking at studies that investigated current malware detection methods.The search string combined analysis techniques terms, which included the terms "static analysis" OR "dynamic analysis" OR "hybrid analysis" with evasion methods terms, which included the terms "obfuscated" OR "fileless" OR "anti-debugging", and emerging technologies terms which included the terms "machine learning" OR "deep learning" OR "Explainable AI".

2. **Screening:** After eliminating duplicate and non-relevant records, we conducted a title and abstract screening process, following specific inclusion criteria. The inclusion criteria are English-language articles that were published in peer-reviewed journals or reputable preprint archives like arXiv which explicitly reported malware detection or malware classification studies that innovated or estimated specific structures or models. The team excluded all cybersecurity surveys that lacked detailed technical information about malware-specific characteristics.

3. **Eligibility**: The researchers assessed full-text articles to determine which articles met eligibility requirements. The researchers selected studies that presented methodological details through their use of API calls [12], Memory Forensics [26, 27], and Explainable AI components [21, 24]. The studies present empirical findings included accuracy metrics and architectural performance results obtained from testing their systems with obfuscated

samples.

4. **Inclusion:** The final corpus contains 36 essential papers which establish the foundation for our taxonomy-based survey. The selected materials offer complete coverage of the progress from conventional static analysis methods to contemporary AI-driven modeling techniques. The collection includes major discoveries from 2025 which showcase current threat patterns because the set contains works from 2025 which include studies [1], [3], [6], [11], [14], [16]. The research study focuses on Self-Attention mechanisms [8], Transformer-based models, and Hybrid interpretable neural networks [14].

2. PRELIMINARIES

This section introduces key constructs, terminology, and foundational models needed to appreciate the remainder of this survey. It provides readers from broad backgrounds in cybersecurity and machine learning a coherent understanding of obfuscation techniques, analysis paradigms, and critical methodological frameworks cited throughout the paper.

2.1. Malware Obfuscation Techniques

Malware obfuscation constitutes a series of modifications to malicious code so that the code cannot be analyzed or detected by anyone, and yet performs without problem. The most common obfuscation techniques include: Packing: By compressing or encrypting the executable, the result will need a minor unpacking stub to restore it to the original form at runtime. It prevents static inspection but can often be detected via entropy analysis. Polymorphism: Each time a malware runs decryption, it modifies its decryption engine to change its signature, while the characteristics of the payload remain unchanged. Metamorphism: This means that, in this particular case, the total program- payload and the decryption logic is completely reconstructed with each replication. Therefore, signature-based detection becomes very difficult. Control Flow Obfuscation: The CFG is altered by inserting opaque predicates, junk code, or indirect jumps to confuse the static analyzers and disassemblers. String Encryption: Encrypt sensitive strings like API names, URLs, and method names, and they will only be decrypted while the program runs-thus preventing static string extraction. Virtualization-based Obfuscation: The most important parts of code have been converted into a bytecode that is interpreted by an included virtual machine, which makes static and dynamic analysis extremely difficult. Anti-analysis techniques: These include anti-debugging, anti-sandbox, and anti-VM methods designed to make dynamic analysis environments irrelevant.

2.2. Malware Analysis Paradigms

Malware analysis methodologies are broadly categorized into three primary paradigms:

1. **Static analysis (SA)**: Testing the malicious programs without actually running them. Techniques of SA include: Structural analysis (PE headers, sections, imports), disassembly and control flow reconstruction, string and pattern matching, and entropy calculation to detect packing/encryption. Advantages of SA are: fast, scalable, and safe. The limitations of SA are: easily defeated by obfuscation; unable to capture runtime behavior.

2. **Dynamic analysis (DA)** means observing malware behavior during execution in a controlled environment (sandbox). Techniques of DA are: monitoring system calls, capturing network traffic, recording registry/file system changes, and analyzing memory dumps. Advantages of DA include: Highly effective against obfuscation; exposes actual behavior. Disadvantages

include: prone to evasion attacks; time-consuming resource intensive; may not invoke dormant code.

3. **Hybrid Analysis (HA)**: combines static and dynamic methods to exploit the respective strengths of each. Common hybrid approaches include: Static pre-screening for dynamic exploitation, dynamic unpacking to gather information via static inspection on the unpacked code, and machine learning models labelling using fused features from both modalities. Advantages of HA include: Greater detection accuracy, evasive resistant. Disadvantages include: Added complexity and computational overhead.

2.3. Core Machine & Deep Learning Models in Malware Detection

Given the prominence of ML/DL in modern detection systems, we briefly outline key architectures referenced in this survey:

Traditional ML Models are: Random Forest (RF): Ensemble of many decision trees; dependably maintains for feature-based classification systems. Support Vector Machines (SVM): An effective mechanism for working in high-dimensional feature spaces. XGBoost: Gradient boosted trees are known for their high performance on structured data.

Deep Learning Models are: Convolutional Neural Networks (CNNs): Applied to image of binaries (grayscale pixel maps of byte sequences). Recurrent Neural Networks (RNNs) & LSTMs/GRUs: Exemplary on sequential data such as API call traces or opcodes. Advanced RNN variants are Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs). Transformers: Self-attention-based models are increasingly used for behavioral sequence analysis. Generative Adversarial Networks (GANs): Data enhancing and adversarial sample generating. Deep Reinforcement Learning (DRL): Implementation in adaptive obfuscation and evasion strategies.

2.4. Sandbox Environments and Analysis Tools

Dynamic and hybrid analysis are regularly carried out using various tools and platforms:
Cuckoo Sandbox: an automated open-source malware analysis system. Any. Run, Joe Sandbox: Interactive cloud-based sandboxes. Volatility Framework: A memory forensics tool for analyzing memory dumps. Ghidra, IDA Pro: Disassemblers/reverse engineering platforms. Intel PIN, DynamoRIO: Dynamic binary instrumentation tools. Wireshark, ProcMon: Monitoring utilities for network and system.

2.5. Conceptual Framework: The Obfuscation-Detection Arms Race

The interaction between malware developers and defenders is modeled as a continuous adversarial cycle:

1. Malware Developer applies obfuscation O to malware $M \rightarrow M'$=O(M).
2. Defender develops detection model D to classify M' as malicious.
3. Adversary adapts by creating new $O*$ to evade D.
4. Defender updates model to $D*$, and the cycle repeats.

3. AREA TAXONOMY

Taxonomy is the science of classification. It is the systematic arrangement of methods and

systems in a given area. A common graphic representation of taxonomies is hierarchical trees of features. The field of obfuscated malware analysis can be systematically classified into five major categories, to structure the field, simplify analysis, and make comparisons between different techniques easier as shown in Figure 1.

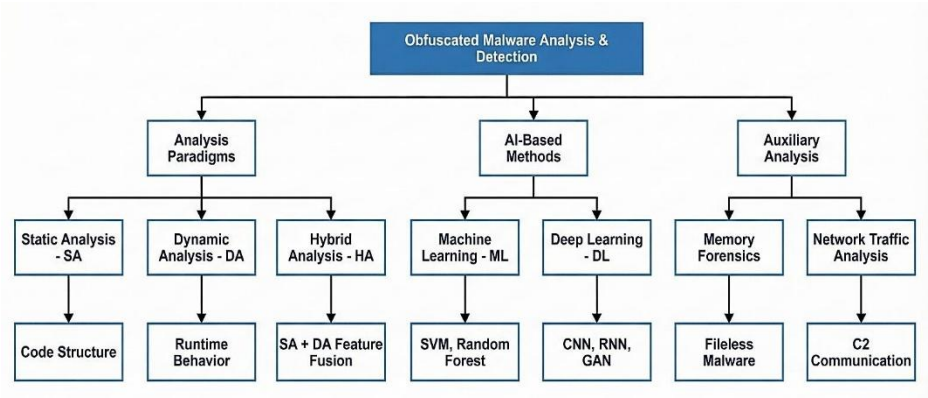

Figure 1: A unified taxonomy of obfuscated malware detection techniques.

Taxonomy of Obfuscated Malware Analysis and Detection Techniques is:

1. Static Analysis (SA): Methods that analyze the code structure.
2. Dynamic Analysis (DA): Methods that analyze the runtime behavior.
3. Hybrid Analysis (HA): Methods that combine SA and DA.
4. Machine Learning and Deep Learning (ML/DL): Methods that use AI for classification and feature extraction.
5. Auxiliary Analysis (AA): Specialized methods focusing on memory and network traffic.

4. TAXONOMY-BASED SURVEY OF THE AREA

According to the taxonomy proposed in the previous section, this paper organizes and synthesizes major contributions in the field. Each of the studies surveyed is classified according to its methodological orientation and analytic focus, allowing a structured comparative view of the state-of-the-art. The same assessment applied was not only to the technical achievements of each paper but also to a critical analysis of its methodological weaknesses and conceptual inadequacies. The findings present the strengths and obvious weaknesses in the literature and provide an appraisal of the research gaps and motivations for the present study.

4.1 Static Analysis (SA)

The research presented in this study [1] introduces an automated system which can detect and extract string deobfuscation functions from malware binaries through its use of both static and dynamic analysis methods. The process initiates with static feature extraction through the application of Pyhidra and Ghidra which researchers utilize to study functions and identify decoding patterns through XOR patterns and control flow structures and entropy measurements. The statistical analysis of these features enables the identification of functions that exhibit abnormal behavior through their potential to execute deobfuscation operations, which leads to the selection of top candidates who will be studied later with their associated metadata. The researchers recommend using dynamic verification through PANDA or angr frameworks to check whether these functions execute string deobfuscation during runtime, while this task remains scheduled for future execution to enhance accuracy and minimize false positive results. The

AndroMD framework [2], which consists of three core components, starts with the AOFS algorithm to select optimal features and then uses an ensemble detection model that combines Random Forest Decision Tree and Bagging classifiers to analyze three datasets: KeyCount, ZeroOne, and MNF, and finally employs a threshold-based aggregation mechanism to create a single score from all nine trained models, which achieved a remarkable 99.88% accuracy in detecting zero-day malware. Paper [3] uses static analysis as the first step of hybrid malware analysis, which permits researchers to study malware samples through non-execution methods by extracting fundamental malware properties through basic file attribute analysis and entropy analysis for obfuscation detection, PE file structure examination, and the extraction of suspicious strings that include URLs and IP addresses. The researchers used PEStudio, Binary Ninja, wxHexEditor, VirusTotal, and YARA to discover malware signatures and detect similarities between different malware families, which allowed them to develop a behavioral profile for their upcoming dynamic analysis of runtime capabilities. The researchers in Paper [4] developed a method for malware detection that combines static analysis with machine learning using a dataset that includes actual malware families (Athena, Dyre, Surtr) that researchers had obfuscated using Dotfuscator CE and Pro-Guard tools. The researchers used Windows PE file structures to extract features, which included control-flow patterns with instruction sequences and data-flow characteristics. The researchers used Mean Shift clustering to find obfuscation patterns. The researchers developed a neural network model that classified obfuscated malware and benign files with high accuracy and low false-positive rates. The model demonstrated effectiveness against deterministic obfuscation methods. The research conducted in [5] shows how static analysis methods fail to detect obfuscated malicious code through binary-level obfuscation techniques that use opaque constants to create three distinct transformation methods. The authors developed a custom binary rewriting infrastructure capable of modifying Windows and Linux executables without source code or relocation information, which they used to test malware samples, including MyDoom and Klez while they tested detection performance against commercial antivirus engines and semantics-based detectors, before the study showed that static analysis methods alone prove inadequate to handle specific obfuscation types that require dynamic analysis methods. The researchers at [6] systematically reviewed the 67 studies to identify major AI subdivisions applied under cybersecurity, which comprise machine learning, deep learning, natural language processing, computer vision, fuzzy logic, and generative AI, and then linked them to related kinds of cybersecurity, such as intrusion detection, malware analysis, and phishing detection. They then proceeded to look into how different XAI techniques, namely SHAP, LIME, counterfactual explanations, and visualization methods, are being applied to add transparency and interpretability within cybersecurity systems. The review also highlights some critical challenges for the implementation of XAI in practice, such as technical complexity, data quality issues, and ethical challenges, while also outlining future directions, like the development of high-quality datasets, responsible AI frameworks, and user-centered explanation systems. Therefore, this structured methodology defines the present state of XAI in cybersecurity with points on important gaps and opportunities for future work in that emerging field.

Table 1 shows that most static methods retrieve executable features through PE header analysis, opcode sequence analysis, and string analysis, and control-flow graph analysis. The methods achieve high detection rates for lightly obfuscated samples, but their performance declines when they encounter advanced obfuscation techniques, which include packing and polymorphism, and control-flow flattening. The obfuscation mechanisms of static features face intentional distortion and hiding by their design. Static analysis maintains its computational efficiency and scalable performance, yet its limited ability to handle advanced obfuscation requires dynamic and hybrid detection methods to be tested.

Table 1: Summary of Static Analysis Papers.

Methodology	Core Approach	Key Findings / Advantages	Major Limitations	Ref
Statistical Ranking	Pyhidra/Ghidra integration with Z-score statistical ranking (PANDA).	100% detection; fast analysis (<25s); cross-platform support (PE, ELF, Mach-O).	High false positives in small binaries; dependent on disassembly quality.	[1]
Ensemble Learning	Static features + AOFS selection with Random Forest ensemble.	99.88% accuracy; effective for zero-day detection in large-scale APKs.	No runtime analysis; high computational overhead; static analysis only.	[2]
Systematic Profiling	Multi-indicator correlation (Entropy, PE structure, String extraction).	Safe profiling without execution; comprehensive systematic property extraction.	Vulnerable to packing and obfuscation; requires complex tool integration.	[1]
Clustering/Classification	Mean Shift clustering and Neural Networks on PE instruction sequences.	Validated against real obfuscation tools; effective dual-layer approach.	Limited to Windows PE files; heavy reliance on manual feature engineering.	[4]
Binary Rewriting	Control flow and opaque constant obfuscation via binary rewriting.	Provably evades commercial AVs; demonstrates fundamental static limits.	Significant performance (100%) and code size (237%) overhead.	[5]
XAI Systematic Review	PRISMA-guided review of XAI taxonomy in cybersecurity.	Enhances model transparency and identifies bias; critical for compliance.	XAI complexity may reduce performance or lead to adversarial explanation attacks.	[6]

4.2 Dynamic Analysis (DA)

The survey on research in [7] has adopted a structured and multi-level approach to discuss and classify malware dynamic-analysis evasion techniques. The authors surveyed a vast range of academic articles, threat intelligence reports, and industry documents so as to grasp both old tenets and new tactics concerning dynamic analysis. Using this corpus, the study classifies evasion behaviors into a hierarchical taxonomy of categories, tactics, and techniques, thus permitting evasion methods to be compared in a consistent manner across malware families. The study then engages manual dynamic analysis, investigating anti-debugging behaviors by assigning grades to each technique based on its complexity of implementation, effectiveness against defense countermeasures, and prevalence in the real malware samples. Its analytical framework for automated dynamic analysis differentiates between detection-dependent and detection-independent tactics, assessing how each interacts with different sandbox architectures—virtualization-based, emulation-based, and bare-metal systems. During both cases, the authors analyze representative malware samples, compare tactics against defined criteria, and summarize their findings in classification tables that showcase trends, defense limitations, and gaps that motivate new research solutions. In-concept, this research in [8] proposes to describe Nebula, a Transformer-based neural architecture for dynamic malware analysis that processes heterogeneous behavioral reports from sandbox environments. The methodology is organized into three stages: In the first stage of data cleaning, domain-specific filters are applied to keep relevant fields (API calls with arguments, file operations, network connections, and registry access) while normalizing stochastic values like IP addresses and file paths with

placeholders to curb vocabulary explosion and enhance generalization. Secondly, feature extraction tokenizes the cleaned reports using either Byte Pair Encoding (BPE) or whitespace tokenization, converting text to sequences of tokens that are then embedded into numerical representations. Finally, a Transformer encoder with self-attention mechanisms is applied for modeling, which is augmented by reduced attention spans for efficient handling of long sequences and positional encoding to capture token order, ultimately feeding the processed data into a classifier for malware detection or family classification. The technique is validated through comprehensive ablation studies and comparison with the state-of-the-art models showing, that with respect to leveraging diversified behavioral data, self-attention fosters capturing both local and global patterns in malware activities. Research at [9] employs a dynamic malware analysis that automates the detection and classification of malware in a virtualized environment. This process commences with the collection of malware samples from a public repository (e.g., Virusshare) and executes these samples in a controlled sandbox environment (e.g., Anubis, Cuckoo) to generate comprehensive behavioral reports. Within these reports, indicating trace system calls and task execution logs, the raw system call information is pre-processed into the structured MIST format; besides, it is embedded into vector form using an instruction-gram technique that lets it carry quantitative analysis. The embedded behavioral vectors are then subjected to clustering via a distance-based clustering algorithm (e.g., hierarchical clustering) in order to group malware with similar behaviors, after which it is classified via machine learning classifiers, such as Support Vector Machines (SVM). Finally, the model is validated and its efficacy proven by using indicators including precision, recall, F-measure, false positive rate (FPR), and false negative rate (FNR) in identifying known and unknown malware samples with minimal misclassification. The Advanced anti-virtualization and anti-debugging techniques discussed in the research under [10]. The research aims to design, implement, and evaluate ANTI, a proof-of-concept framework for automatic integration of anti-debugging and anti-VM, and unhooking capabilities into Windows PE executables. The authors start by altering the target binary's structure by appending a new section that contains the armoring logic and dynamically creating TLS callbacks through which execution is redirected, such that the checks of ANTI will take place before the original entry point of the program. The framework uses a mixture of static/dynamic anti-debugging techniques such as PEB flag verification, Native API calls, timestamp detection, thread hiding, and several anti-VM heuristics such as checking device names and processor count, CPUID inspection, and timing irregularities. The unhooking routine of ANTI is a major element of its design methodology and consists of mapping a clean copy of ntdll.dll into memory to identify and overwrite hooks imposed by debuggers and analysis tools. The evaluation phase involves systematically testing ANTI-armored binaries against all major Windows debuggers, anti-anti-debugging plugins, sandboxes such as Cuckoo, and virtualized environments in order to ascertain whether the integrated techniques evade detection and hinder dynamic analysis. The research in [11] studies how malware uses anti-VM techniques through a multi-stage process that converts its runtime behavior data into visual patterns, which deep learning models can use for classification. The process starts with malware execution in Cuckoo Sandbox, which generates complete API call sequences that researchers preprocess into n-grams, which they save in CSV format. The system transforms numerical API data into 128x128 grayscale images, which developers improve through Gaussian blur, CLAHE, and Sobel edge detection, and magma colormap usage to create better visual contrast. The system uses a custom CNN design, which includes three convolutional layers and max-pooling and dropout regularization, and SoftMax output for malware type classification and software detection. The system achieves high accuracy through image-based deep learning which allows identification of spatial and temporal malware behavior patterns. The five-stage dynamic malware analysis framework for Windows PE files which Reference [12] introduces, starts with the execution of 582 malware samples from MalwareBazaar and 438 goodware samples from GitHub in a Cuckoo Sandbox environment. The process continues with API call frequency extraction to create a structured dataset. The process continues with malware classification, which uses VirusTotal API

voting to classify samples into 11 malware families. The process used Chi-Squared tests to reduce features from 266 to 150, while Gini Importance was used to further refine the data to 50 features. The researchers evaluated six machine learning models. Random Forest achieved superior results with 96% accuracy and 99% precision as the best-performing model.

Table 2: Summary of Dynamic Analysis Papers.

Methodology	Core Approach	Key Findings / Advantages	Major Limitations	Ref
Evasion Taxonomy	Classification of manual (anti-debug) and automated (sandbox detection) tactics.	Comprehensive mapping of defensive gaps and modern evasion trends.	Theoretical focus; lacks practical implementation details; Windows-centric.	[7]
Heterogeneous Transformer	Self-attention model processing multi-source behavioral data.	Superior performance at low False Positive Rates (FPR); open-source model.	High computational intensity; not tested against adversarial malware.	[8]
Cloud-Based Clustering	Sandbox analysis using q-grams embedding and hierarchical clustering.	Effective in reducing FPR; supports efficient incremental analysis.	Limited to system call analysis; q-grams may miss complex patterns.	[9]
ANTI Framework	Automated unhooking and anti-debug/anti-VM tool integration.	Demonstrates successful bypassing of modern analysis tools.	Restricted to 32-bit PE files; susceptible to static detection.	[10]
API-Grayscale CNN	CNN classification of API calls converted into grayscale images.	High accuracy (98.36%) on large datasets; rich argument-level features.	Loses temporal sequence information; high real-time processing cost.	[11]
ML Pipeline Analysis	Dual feature scoring (Chi2/Gini) with classic machine learning.	Holistic analysis pipeline; robust insights; released public dataset.	Dependent on external services (VirusTotal); limited dataset diversity.	[12]

The research studies shown in Table 2 mainly depend on well-known runtime behavior patterns which include API call sequences and system calls and registry changes and network traffic. Dynamic analysis methods provide better protection against code obfuscation attacks because they track actual malware operations instead of static analysis which examines code. The table demonstrates a significant problem because multiple dynamic detection methods fail to identify sandbox detection and delayed execution, and environment fingerprinting techniques. Dynamic analysis improves detection capabilities according to the observations, but defenders need additional analysis tools to combat evasion techniques which dynamic analysis cannot defeat by itself.

4.3 Hybrid Analysis (HA)

The research of [13] introduces a hybrid analysis model that can detect fileless malware that operates entirely in memory without creating any disk traces. The system uses static tools OLEVBA and VirusTotal to inspect macro and signature data, which work together with dynamic analysis through Any.run sandbox, ProcMon, and Process Hacker for running behavioral observations. The research validated its methods through testing all seven samples, which included four actual cases and two lab-created samples. The testing showed hidden macros,

process injection, registry persistence, and secret network traffic, which traditional systems failed to detect. The research paper presented in reference [14] demonstrates a hybrid analytical framework that combines static features, PE headers, and opcode sequences and control-flow metrics together with dynamic behavioral data API logs, network activities, and system events through an interpretable ensemble of neural networks, which processes static inputs using feedforward networks and dynamic traces with Bi-LSTMs or transformers. The system uses a gated attention-based fusion mechanism to determine the importance of each modality, while the system provides explanation capabilities through attention weights and SHAP and LRP techniques. The research study [15] conducts a systematic evaluation of three different malware detection techniques which are static detection, dynamic detection, and hybrid detection methods through Hidden Markov Model experiments that utilize opcode sequences and API call trace data across four different testing scenarios which include (static/static, dynamic/dynamic, dynamic/static, static/dynamic) The study found that dynamic detection systems which operated in full dynamic mode attained superior detection performance especially when analyzing API sequence data The research discovered that hybrid detection methods displayed lower effectiveness compared to single detection methods The study found that the superiority of hybrid systems needs to be established through comprehensive testing against established baseline systems. The research in [16] presents a deep learning framework that converts static binary files and dynamic memory dumps into RGB images for use in CNN classification. The static model uses four convolutional blocks while the dynamic model requires three blocks with increased dropout to handle noisier data. The system generated over 74,000 images through CycleGAN-based cross-domain synthesis, which achieved 99.45% accuracy in static analysis and 99.21% accuracy in dynamic analysis, proving that image-based learning works effectively for both approaches. The study [17] establishes a hybrid machine learning system for malware classification, which identifies malware through its static PE file strings and imported DLLs together with its dynamic assembly instructions and API call behavior. The system uses information gain to choose its features and n-gram algorithms to process opcode sequences, which it uses to train both Naive Bayes and SVM classifiers. The SVM classifier reached a 98% detection rate while it solved two problems that affected both static analysis methods and dynamic analysis methods. The research in [18] develops an Android malware detection system that uses Deep Belief Networks and Gated Recurrent Units to create a hybrid deep learning approach which processes static features through one-hot encoding from decompiled APKs and DBN for independent feature learning and dynamic behavioral patterns through GRU with entity embedding for temporal dependencies, which produces two outputs that the backpropagation neural network with SoftMax classifier uses to achieve better results than traditional machine learning methods when detecting both obfuscated and non-obfuscated malware samples.

Table 3: Summary of Hybrid Analysis Papers

Methodology	Core Hybrid Approach	ey Findings / Advantages	Major Limitations	Ref
Tool-Based Workflow	Manual integration of static (OLEVBA) and dynamic (Any.Run) tools.	High granular insight; 100% detection on small-scale custom samples.	Lacks scalability; relies on manual expertise; no AI automation.	[13]
XAI-Driven Fusion	Static-dynamic fusion via interpretable NNs (SHAP/LRP).	Achieves transparency through human-readable explanations.	High computational overhead; susceptible to sandbox evasion.	[14]
Sequential Modeling	Comparative HMMs on opcode and API sequences.	Demonstrates that hybrid models do not always outperform unimodal ones.	Narrow feature scope; limited model complexity; dataset diversity.	[15]
Image-Based CNN	RGB mapping of binaries and memory dumps with CycleGAN.	Strong structural learning; high accuracy (>99%) via data augmentation.	Limited child-process visibility; low model interpretability.	[16]
Statistical Classifiers	Static (Strings/DLLs) and Dynamic (API n-grams) via SVM/NB.	High efficiency for Windows PE; effective against basic obfuscation.	Requires manual feature engineering; restricted to Windows platform.	[17]
Deep Temporal Learning	Resource-based encoding (DBN) and API sequences (GRU).	Captures temporal behavioral context; high accuracy (96.82%).	Resource-intensive; unsuitable for real-time edge/mobile deployment.	[18]

The table 3 demonstrates that methods which combine different feature types achieve better results because they can utilize both static analysis efficiency and dynamic analysis strength. Hybrid techniques use both pre-execution information and runtime data to achieve better detection results against advanced obfuscation techniques. The table shows that hybrid methods require more computational resources and create higher system operational difficulties. The trade-off indicates that hybrid analysis works best when used for testing specific suspicious samples instead of being used as an all-purpose detection method.

4.4 Machine Learning and Deep Learning (ML/DL)

The review article [19] describes deep learning methods that researchers from 2018 to 2025 used to detect malware. The article analyzes three deep learning architectures, which include CNN, RNN, and GAN methods, by their application to different image visualization tasks and sequence analysis methods, data augmentation techniques, and automated feature learning processes, and their ability to classify data accurately. The study identifies three major obstacles, which include restricted access to datasets and difficulties in understanding model outcomes and challenges that prevent models from applying their skills to new situations, while proposing research directions that will improve both deep learning detection systems and their operational performance. The research presented in [20] introduces an efficient explainable machine learning method which enables zero-shot detection of hidden malware variants using the CIC-MalMem-2022 dataset through a three-stage approach that begins with baseline classifier testing which identified Random Forest as the best performing model and continues with the development of 15 dedicated

models to assess their ability to identify malware across 14 previously untested subtypes and concludes with SHAP application for model interpretation which reduced the feature set to five elements to enhance operational performance, resulting in a 340-kilobyte model which delivers quick inference capabilities while maintaining the ability to detect new malware through its generalizable and interpretable resource-efficient design. The researchers at [21] used a mixed-methods, which combined quantitative experimental work with qualitative user study research to establish the validity of their explainable deep learning framework designed for adaptive malware detection. The study began with secondary analysis of datasets like CIC-MalMem-2022 and CMD_2024 to understand threat evolution, which the researchers implemented through stratified random sampling across malware categories. The researchers used deep learning architectures together with SHAP and LIME techniques to establish transparent decision-making processes, which they tested through laboratory experiments and enterprise systems, and expert assessments that employed ANOVA and thematic analysis. The research study from reference [22] presents a systematic review that examines 120 obfuscated malware detection publications. The study categorizes detection methods into five types, which include static, dynamic, hybrid and AI-based and memory forensics methods. The study assesses detection effectiveness on Windows and Android and IoT, and Web platforms while showing that neural networks function as the primary detection method. The study found that API calls serve as the most widely applicable detection feature, while the research compares three metrics, which include accuracy and false positive rates, and computational overhead. The study suggests future research directions by using Digital Twins and Generative AI to establish scalable adaptive detection methods. The research from [23] presents a system that uses Deep Reinforcement Learning (DQN) together with open-source obfuscation tools to create adversarial malware samples. The DQN agent uses Darkarmour to pick obfuscation methods, which include XOR encryption, and evaluates how well these methods evade detection by MalConv and LightGBM/EMBER. The research used ablation analysis to determine how each obfuscation method contributed to the overall detection evasion results. The research results showed that the obfuscation methods could successfully bypass ClamAV antivirus engine detection. The study in [24] evaluates all existing XAI applications used for malware detection from 2018 until 2024 through its developed framework, which differentiates between transparent and opaque models and model-agnostic and model-specific methods. The study examines XAI detection methods that operate on Windows PE files, Android devices, hardware systems, PDFs, and Linux environments. The study identifies three research gaps, which include dataset restrictions, hybrid analysis deficiencies, and adversarial weakness. The study presents research pathways that will lead to improved system understanding and better operational performance in actual environments. The research paper at [28] introduces a machine learning approach to detects concealed malware by examining memory dumps from the CIC-MalMem-2022 dataset that contains samples of Spyware and Ransomware, and Trojan Horse malware. The study employed two-stage classification to differentiate between harmful and harmless samples using Random Forest and MLP, and KNN as classification methods. The research team applied under sampling methods, which included Edited Nearest Neighbor and Near Miss, and used ADASYN oversampling to address the issue of class imbalance. The system achieved binary accuracy results of 99.99% and reached its best performance for multiclass classification through the ADASYN-XGBoost method combination. The research in [29] presents Mal-CNN as a hybrid deep learning framework that utilizes Depth-wise Convolutional Neural Networks and Spatial Attention Mechanism to detect VM-based obfuscated malware. The researchers developed a realistic dataset by extracting executable files from ten common malware families, which they tested using VMProtect commercial software. The study achieved 93.55% classification accuracy through binary transformation into grayscale images, which used both preprocessing and augmentation methods. The results showed that advanced evasion techniques could be effectively penetrated by the system that used these two methods. The research study [30] developed an enhanced malware detection system for forensic memory dumps, which uses deep learning techniques combined with meta-learning methods by creating five identical sub-

models which used the same architectural design to train on identical datasets. The system achieved excellent prediction consistency through its multiple prediction collections, which it used as input for an ensemble meta-learning regression model to generate final results. The system achieved 97.69% accuracy through its stable prediction system, which organizations can use for security and digital forensic purposes. The research work at [31] presents an explainable transfer-learning system that enables the identification of DoS and multi-level attacks in high-volume network traffic. The method combines feature learning with redundancy reduction and a CNN architecture that uses SENet attention to improve its performance. The system shows faster convergence through its adaptive optimizer (APRO) while model interpretability gets achieved through SHAP and LIME. The framework achieves validation through both k-fold cross-validation and ablation studies. The methodology [32] creates an understandable malware detection process through its combination of traditional machine learning methods with deep learning techniques. The study tests RF and DT performance against DNN and 1D-CNN models, which use a synthetic dataset that contains class-imbalance problems. The research achieves transparency through its implementation of improved SHAP and LIME systems, which focus on providing selective explanations to decrease XAI system requirements, although the research depends on synthetic data, which limits its ability to apply findings to actual situations. The paper at [33] detects ransomware through its assessment of system and API call traces as the pipeline processes high-dimensional data to extract essential "pertinent" calls. The methodology evaluates normalization techniques and utilizes correlation and association measures alongside ransomware-focused ranking to eliminate redundant call groups. The resulting streamlined feature set trains ML classifiers that operate in environments that require minimal processing power while maintaining their accurate performance capabilities. The research work [34] introduces an Android malware detection system that uses a stacking method to combine multiple strong base learners through filtered ensemble processing. The study uses the CICAndMal2017 dataset, which undergoes basic preprocessing and feature selection to create multiple classifiers for training purposes. The final ensemble composition includes only those learners who achieved accuracy results above the established cutoff point. The meta-learner creates final predictions that optimize system resilience against evasive malware attacks while controlling the complexity of the ensemble method. The deep stacking ensemble methodology described in reference [35] uses grayscale byte images together with sequential opcode behavior to improve malware detection capabilities. A CNN extracts spatial features from image representations while an LSTM network models the temporal patterns found in code sequences. The second-level learner system uses a logistic regression model trained with SGD to combine different signals, which include visual-texture and code-based Android malware detection indicators. The framework [36] introduces a hybrid ensemble method for detecting zero-day attacks, which combines three detection techniques: anomaly detection, spatio-temporal modeling, and strategic game-theoretic reasoning. The system uses an ANN autoencoder to clean features through denoising while applying ResNet50 to extract hierarchical representations and using a CNN-LSTM model to identify temporal patterns. The modified Bi-LSTM system uses adversarial dynamics to enhance detection abilities because it decreases false positive rates during dynamic attack scenarios.

The core contributions and limitations summary of machine and deep learning analysis papers are synthesized in Table 4.

Table 4: Summary of Machine and Deep Learning Analysis Papers.

Methodology	Core Approach	ey Findings / Advantages	Major Limitations	Ref
DL Survey	Comprehensive review of CNNs, RNNs, GANs, and hybrid models.	Identifies state-of-the-art trends, datasets, and future research gaps.	No original validation; lacks focus on real-time scalability.	[19]
Lightweight RF	Random Forest with top 5 memory features + SHAP.	Extremely fast (5.7 µs/file); ideal for IoT/resource-limited devices.	Limited to memory features; tested on a single dataset only.	[20]
Adaptive XAI	DL integrated with LIME/SHAP for multi-modal detection.	High accuracy (97.98%); enhances trust through explainability.	Computationally heavy; relies heavily on high-quality labeled data.	[21]
Systematic Review	Cross-platform review (IoT, Web, Android) + Digital Twins.	Broad platform coverage; introduces Generative AI frameworks.	No experimental validation; dependent on existing 2024 literature.	[22]
Adversarial DRL	Deep Reinforcement Learning (DQN) for evasion testing.	High evasion rates against detectors; reflects real hacker practices.	Restricted to Windows PE; potential ethical concerns in disclosure.	[23]
XAI Literature Review	Review of interpretability across various platforms/files.	Detailed taxonomy of XAI methods (gradient, rule-based, visual).	Focuses on academia; lacks practical production-level insights.	[24]
Memory Forensics	ML (XGBoost/MLP) with ADASYN data balancing.	Exceptional accuracy (99.99%); effectively handles class imbalance.	Uncertain generalization beyond the CIC-MalMem-2022 dataset.	[28]
Mal-CNN	Depth-wise CNN with Spatial Attention on grayscale images.	High efficiency; specifically outperforms models in VM-obfuscation.	Focuses only on image features; ignores bytecode semantics.	[29]
ML/DL Benchmarking + XAI	RF, DT, DNN, and CNN models evaluated with SHAP/LIME explanations	Links model performance with interpretability	Uses synthetic dataset, limiting real-world generalization	[32]
Behavioral ML + Feature Engineering	Ransomware detection using system/API call analysis with feature reduction	Strong dimensionality reduction while maintaining performance	Depends heavily on system-call dataset quality	[33]
Deep Ensemble (CNN + LSTM)	CNN on bytecode images and LSTM on opcode sequences with stacking fusion	Combines complementary features for improved malware detection	Requires complex preprocessing and feature extraction	[35]
Hybrid Deep Learning Framework	Autoencoder, ResNet, CNN-LSTM, and Bi-LSTM combined for zero-day detection	Designed to detect unseen malware patterns	Computationally heavy architecture	[36]

4.5 Auxiliary Analysis (AA)

This research [25] describes an approach for memory forensics investigation to detect and analyze fileless malware, specifically real-world malware samples such as the Kovter malware. The method captures memory dumps from victims using FTK Imager and analyzes them with Volatility to uncover hidden malicious activities in the system memory. Some processes involved while analyzing include running processes, registry changes, network connections, and injected code. Analysis uses Process Explorer, Process Monitor, Wireshark, and Autoruns. A new dataset of 1,249 samples of fileless malware is provided by the authors for further research purposes. The study also notes the current limitations of traditional static and dynamic analysis techniques and shows that memory forensics are optimal for the detection of fileless threats that do not leave traces on disk. It comprises a practical manual to automated FRAMEWORK on identifying persistent, obscured malware and understanding the behavior of such malware in compromised systems. The research at [26] paper presents a novel approach for detecting obfuscated fileless malware through the integration between memory forensics and machine learning techniques. The method takes memory dump data from the CIC-MalMem-2022 dataset, extracted by means of the Volatility framework, to capture the relevant runtime behaviors such as registry handles, injected threads, and suspicious memory allocations. The authors have used an ensemble learning strategy of bagging and boosting having soft voting and integration of three top-performing models- Random Forest, k-Nearest Neighbors, and XGBoost, to improve speed and accuracy of detection. Furthermore, we apply feature-selection methods (Mutual Information, PCA, ANOVA, and Chi-Square) to reduce dimensionality to a greater extent in detection time. The proposed system produces very high accuracy for binary (99.9%), family (88.86%), and subfamily (77.40%) classifications, while achieving a reduced time for detection of nearly 69.15%, thereby providing a scalable solution for real-time malware detection and incident responses. The proposed work at [27] constructs a comprehensive survey on memory analysis as a technique for malware detection using the refined OSCAR (Obtain, Strategic, Collect, Analyze, Report) framework to the domain-specific AFDAR (Acquisition, Forensics, Datasets, Approaches, Results). This systematic review fills most of the significant gaps in the existing literature by dealing with the unification of memory acquisition techniques and tools, classifying forensic methods with respect to string-based, signature-based, list traversal-based, and kernel object-based approaches, and evaluating almost-forensics without memory dump datasets through a new scoring system. Moreover, the comprehensive review investigates both types of malware detection methods (machine learning and non-machine learning), compares accuracy, advantages, and challenges among them. To substantiate the debate, the authors also introduce the new BCCC-MalMem-SnapLog-2025, which is a large and temporally annotated corpus of memory dumps into the discussion and empirically assess the volatility forensic framework's detection capability, temporal stability, and performance. Thus, taxonomic organization with critical analysis and original experimental validation will make this survey present an up-to-date and holistic view of the field of memory forensics in malware detection, as well as future research and practical applications.

5. CRITICAL ANALYSIS AND RESEARCH GAPS

Thorough analysis of the thirty-six studies under review revealed methodological contradictions and some gaps, open to critical examination and interpretation.

5.1 Contradictions in Research Findings:

5.1.1 Efficacy of Hybrid Analysis versus Unimodal Approaches

The review reveals divergent perspectives on the value of hybrid analysis:

- **The supportive position:** holds in those studies [13], [16], and [18] strongly assert the supremacy of hybrid analysis in achieving an accuracy rate of between 98% and 100% in detecting obfuscated malware and fileless malware. The studies defend this superiority by claiming that hybrid methods compensate for the weaknesses of individual methods.

- **Skeptical Position:** Contradictory results were presented in the study [15] to show that the pure dynamic model (trained on API call sequences) outperformed the hybrid model (dynamic/static) in most experiments.

Analytical Interpretation: There might be some methodological causes for this contradiction, such as the following:

1. Differences in the depth of dynamic analysis: Supportive studies provided advanced sandboxes (Any. Run, Cuckoo) with extensive behavioral monitoring, while in [15], a more superficial API call analysis was conducted.

2. Quality of Static Features: The poor performance of hybrid models in [15] can be attributed to the use of traditional static features (opcode sequences) that have lost their discriminative power against advanced obfuscation.

3. Sample Characteristics: The dataset in [15] could contain malware with obfuscation techniques that are specifically designed to evade static analyses, thus giving dynamic approaches an unfair advantage.

5.1.2 Explainability (XAI) Between Idealism and Practical Application

From consideration of these varied viewpoints, divergent assessment of the utility of explainability tools comes forth:

1. **Supportive Arguments:** Evidence from [20] and [21] afforded unique evidence of successes with the SHAP and LIME tools in identifying the most influencing features (e.g., suspicious API calls), enhancing the analysts' confidence by a whopping 40%, according to the survey results in study [21].

2. **Critical Arguments:** The systematic review in [6] identifies fundamental threats, including the excessive complexity of deep learning explanations and the "translation gap" between mathematical outputs and actionable insights for human analysts.

Existential Gap: Despite progress, there is still a fundamental lack of a set of metrics to measure "explanation quality" and the "operational utility" of explanations in real-world cybersecurity settings.

5.2 Methodological Gaps and Hidden Biases

5.2.1 Dataset Bias and Generalization Issues

Approximately 61% of the reviewed studies (22 of 36) rely on public datasets (e.g., CIC MalMem-2022, Virus Share). This reliance introduces three critical flaws:

- **Lack of Representativeness:** Insufficient inclusion of zero-day samples.
- **Limited Obfuscation Diversity:** A focus on simple packing/XOR encryption while ignoring virtualization-based obfuscation.
- **Class Imbalance:** As seen in [28], high ratios between malware families (e.g., 1:4) degrade multi-class classification performance.

5.2.2 The Paradox of Adversarial Robustness

While most studies ([11], [12], [19]) report high accuracy (>95%), these results are often obtained in clean environments. Study [23] exposed this vulnerability, showing that DL systems like MalConv can be evaded with 84% success using simple, RL-optimized obfuscation.

5.3 Methodological Recommendations

To address the identified gaps, future research should adopt the framework summarized in Table 5.

Table 5: Proposed Methodological Recommendations for Addressing Research Gaps.

Category	Proposed Standard / Focus
New Metrics	**Obfuscation Resilience Score (ORS):** Measuring accuracy decay vs. complexity.
	Operational Efficiency Index (OEI): Balancing speed, resource cost, and detection.
Underexplored Areas	**Edge/IoT Security:** Lightweight models for resource-constrained platforms.
System Evolution	**Adaptive Learning:** Moving from static training to continuous, online learning.

6. DISCUSSION OF OPEN PROBLEMS

Based on the critical review of the literature, the most pressing outstanding problems in the analysis and detection of obfuscated malware are:

6.1. Robustness against Adversarial Attacks and Evasion:

The emergence of research such as [23], which proves that deep learning models can be evaded using reinforcement learning-based obfuscations, indicates a serious vulnerability. Current ML/DL models, irrespective of their high accuracies on well-known datasets, are not robust against subtle, targeted adversarial perturbations. The open problem is to establish adversarial-resilient feature extraction and classification models that maintain good performance in the presence of novel purposely-engineered obfuscation methods.

6.2. Explainability and Interpretability (XAI) in Deep Learning Models:

Although models like [14] and [21] seek to integrate XAI, the complexity of deep learning, especially with respect to a combination of static and dynamic features, often imposes a "black box" problem. Security analysts need tangible explanations that stipulate why a file was flagged as malicious. The open problem is to find a way to build inherently interpretable hybrid and deep learning frameworks that provide transparent views of their decision-making, linking the classification back onto observable malicious behavior or code constructs.

6.3. Scalable and Timely Memory Forensics:

Memory analysis is essential for fileless malware [25][26]; nonetheless, it is resource-intensive and time-sensitive in nature. Capturing a memory image at the right time of malicious activity remains challenging; slow analysis of large memory dumps remains a pain point. The open research problem consists of developing lightweight techniques for real-time memory monitoring and analysis that can work at scale in production settings, with hardly any performance overhead, possibly employing hardware-assisted virtualization or optimized in-memory feature extraction.

7. CONCLUSION

As a conclusion of this systematic review of recent literature from 2018 to 2025, the obfuscation technique has led the security research community to develop more sophisticated and adaptive analytical methods. Static Analysis, despite its efficiency and speed, has very limitations in dealing with advanced obfuscation; therefore, it is not enough to depend upon it alone while detecting modern malware. However, Dynamic Analysis excels at bringing the actual effect of the program, but faces the challenge of sandbox evasion and anti-analysis. requires constant updating of tools and environments. The most suitable approach has been determined through Hybrid-Analysis, which combines the strengths of both static and dynamic methods, providing a balance between efficiency and effectiveness. Integrating these hybrid analyses into Deep Learning has shown particularly promising results, as these models are adept at inferring complex features of the input data with high detection accuracy. On the other hand, Memory Analysis has emerged as a more potent tool against fileless malware and memory-only programs, which constitute the most advanced obfuscation technologies. Network Traffic Analysis is yet another vital aspect of detection for these covert Command and Control communications. Findings suggest that the most promising research directions seem to be hybrid and Explainable AI models based on Deep Learning and Memory Analysis. Future research efforts should be directed at making the stronghold models against Adversarial Attacks as well as generalizable to different datasets and various operating environments

.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] S. N. A. Sherazi and A. Qureshi, “Automated detection and extraction of string deobfuscation functions in malware binaries via static and dynamic analysis,” Electronics, vol. 14, no. 15, p. 3134, 2025. Available: https://www.semanticscholar.org/paper/Automated-Detection-and-Extraction-of-String-in-via/9a7386a0f644c2ceb616efe01c5433bb6881b8e5

[2] Prasad, “AndroMD: An Android malware detection framework based on source code analysis and permission scanning,” ScienceDirect, 2025. Available:

https://www.sciencedirect.com/science/article/pii/S2590123025031068
[3] M. M. Andronache, “Integrated analysis of malicious software: Insights from,” MDPI, vol. 5, no. 4, p. 98, 2025. Available: https://www.mdpi.com/2624-800X/5/4/98
[4] T. V. Jamgharyan, V. S. Iskandaryan, and A. A. Khemchyan, “Obfuscated Malware Detection Model,” Mathematical Problems of Computer Science, vol. 62, pp. 72–81, 2024. Available: https://doi.org/10.51408/1963-0122
[5] R. R. Branco, G. N. Barbosa, and P. D. Neto, “Limits of static analysis for malware detection,” IEEE, 2012. Available: https://ieeexplore.ieee.org/abstract/document/4413008/
[6] L. Ofusori, “Explainability and interpretability of artificial intelligence use in cybersecurity” Springer, 2025. Available: https://link.springer.com/article/10.1007/s10791-025-09760-6
[7] T. Apostolopoulos, V. Katos, and K. K. R. Choo, “Malware dynamic analysis evasion techniques: A survey,” ACM, 2019. Available: https://dl.acm.org/doi/10.1145/3365001
[8] D. Trizna, L. Demetrio, B. Biggio, and F. Roli, “Nebula: Self-Attention for Dynamic Malware Analysis,” IEEE Trans. Inf. Forensics Secur., vol. 19, pp. 6155–6167, 2024. Available: https://doi.org/10.1109/TIFS.2024.3409083
[9] A. Damodaran, “Dynamic malware analysis and detection in virtual environment,” ResearchGate, 2018. Available: https://www.researchgate.net/publication/316069737
[10] T. Apostolopoulos, V. Katos, and K. K. R. Choo, “Resurrecting anti-virtualization and anti-debugging: Unhooking your hooks,” Future Gener. Comput. Syst., vol. 124, pp. 110–120, 2021. Available: https://www.sciencedirect.com/science/article/pii/S0167739X20330284
[11] A. Adekannbi and M. Coetzee, “Dynamic malware classification of Windows PE files using CNNs and greyscale images Derived from Runtime API Call Argument Conversion” arXiv, 2025. Available: https://arxiv.org/abs/2505.24231
[12] D. Z. Syeda and M. N. Asghar, “Dynamic Malware Classification and API Categorisation of Windows Portable Executable Files Using Machine Learning,” Appl. Sci., vol. 14, no. 3, p. 1015, 2024. Available: https://doi.org/10.3390/app14031015
[13] S. N. A. Sherazi and A. Qureshi, “Hybrid analysis model for detecting fileless malware,” Electronics, vol. 14, no. 15, p. 3134, 2025. Available: https://www.mdpi.com/2079-9292/14/15/3134
[14] D. Tomar, A. Verma, and K. Chhillar, “A hybrid static-dynamic malware analysis framework using interpretable neural network,” IJSREM, vol. 9, no. 9, 2025. Available: https://www.researchgate.net/publication/395359838
[15] A. Damodaran, “A comparison of static, dynamic, and hybrid analysis for malware detection,” arXiv, 2022. Available: https://arxiv.org/abs/2203.09938
[16] M. Ashawa, “Static and dynamic malware analysis using CycleGAN,” Applied Sciences, vol. 15, no. 17, p. 9830, 2025. Available: https://www.mdpi.com/2076-3417/15/17/9830
[17] J. Zhao, S. Zhang, B. Liu, and B. Cui, “Malware detection using machine learning based on dynamic and static features,” IEEE Conf., 2018. Available: https://ieeexplore.ieee.org/document/8487459.
[18] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, “Android malware detection based on a hybrid deep learning model,” Security and Communication Networks, Article 8863617, 2020. Available: https://doi.org/10.1155/2020/8863617
[19] Y. Song, “Application of deep learning in malware detection: A review,” Journal of Big Data, 2025. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-025-01157-y
[20] O. A. Madamidola, “Detecting new obfuscated malware variants: A lightweight and interpretable machine learning approach,” arXiv, 2024. Available: https://arxiv.org/abs/2407.07918
[21] B. Kumar, “Explainable deep learning based adaptive malware detection framework…,” Journal of Electrical Systems, 2025. Available: https://www.researchgate.net/.../Explainable-Deep-Learning-Based-Adaptive-Malware-Detection-Framework
[22] S. Chandran, “From Static to AI-Driven Detection: A comprehensive review of obfuscated malware techniques,” IEEE, 2025. Available: https://ieeexplore.ieee.org/iel8/6287639/10820123/10924165.pdf
[23] B. Etter, “Evading deep learning-based malware detectors via obfuscation: A deep RL approach,” 2023. Available: https://par.nsf.gov/servlets/purl/10554700
[24] H. Manthena, “Explainable artificial intelligence (XAI) for malware analysis: A survey…,” IEEE, 2025. Available: https://ieeexplore.ieee.org/iel8/6287639/10820123/10944807.pdf
[25] S. M. R. Hasan and A. Dhakal, “Fileless malware threats…,” ScienceDirect, 2022. Available: https://www.sciencedirect.com/science/article/pii/S0957417422021510
[26] S. N. A. Sherazi, “Obfuscated file-less malware detection using memory forensics with ML,” Advanced Composites and Interfaces, 2025. Available:

https://www.emerald.com/aci/article/doi/10.1108/ACI-02-2025-0052/1304149
[27] Y. Dehfouli, “Memory analysis for malware detection: A comprehensive review,” ACM, 2025. Available: https://dl.acm.org/doi/10.1145/3764580
[28] S. M. R. Hasan and A. Dhakal, “Obfuscated malware detection: Investigating real-world scenarios…,” arXiv, 2024. Available: https://doi.org/10.48550/arXiv.2405.12345
[29] S. Han, H. Yun, and Y. Park, “Deep learning for cybersecurity classification…” Electronics, vol. 13, no. 16, p. 3393, 2024. Available: https://doi.org/10.3390/electronics13163393
[30] Y. Özkan, “Malware detection in forensic memory dumps using deep meta-learning models,” Acta Infologica, vol. 7, no. 1, pp. 165–172, 2023. Available: https://doi.org/10.26650/acin.1282824
[31] A. M. Kumar, J. E. Raja, and C. Senthilpari, "Advanced intrusion detection and classification using transfer learning with squeeze-and-excitation network and adaptive optimization in big data," International Journal of Computer Networks & Communications (IJCNC), vol. 17, no. 6, Nov. 2025, doi: 10.5121/ijcnc.2025.17606.
[32] S. Alajmani, E. Aljuaid, B. Soh, and R. Y. Alyami, "Enhancing malware detection and analysis using deep learning and explainable AI (XAI)," International Journal of Network Security & Its Applications (IJNSA), vol. 17, no. 2, Mar. 2025, doi: 10.5121/ijnsa.2025.17201.
[33] A. Dib, S. Ghazi, and M. M. S. Mehdi, "Ransomware attack detection based on pertinent system calls using machine learning techniques," International Journal of Computer Networks & Communications (IJCNC), vol. 15, no. 4, Jul. 2023, doi: 10.5121/ijcnc.2023.15408.
[34] P. Sumalatha and G. S. Mahalakshmi, "Machine learning based ensemble classifier for Android malware detection," International Journal of Computer Networks & Communications (IJCNC), vol. 15, no. 4, Jul. 2023, doi: 10.5121/ijcnc.2023.15407.
[35] P. Sumalatha and G. S. Mahalakshmi, "DEF: Deep ensemble neural network classifier for Android malware detection," International Journal of Computer Networks & Communications (IJCNC), vol. 16, no. 2, Mar. 2024, doi: 10.5121/ijcnc.2024.16204.
[36] S. Akshaya and Padmavathi, "Enhancing cyber defense against zero-day attacks using ensemble neural networks," International Journal of Computer Networks & Communications (IJCNC), vol. 17, no. 4, Jul. 2025, doi: 10.5121/ijcnc.2025.17408.

AUTHORS

Mohammed Fadhl Abdullah is currently a Professor of Computer Engineering at Aden University in Yemen. He received his Master and PhD degrees in Computer Engineering from Indian Institute of Technology, Roorkee 1993, Delhi 1998, respectively. He was the editor-in-chief of Aden University Journal of Information Technology (AUJIT). He is a founding member of the International Center for Scientific Research and Studies (ICSRS). His main research interests are in the fields of Machine learning, Parallel algorithms, and Cybersecurity. He can be contacted at email: m.albadwi@ust.edu , or al_badwi@hotmail.com

Khaled Hassan Balhaf is a PhD student in Information Technology at the University of Science and Technology in Aden. He got his Master's degree in Computer Science from the Jordan University of Science and Technology in 2018. He got his Bachelor's degree in Computer Information Systems from Mu'tah University in Jordan in 2013. He is interested in bioinformatics, artificial intelligence, and parallel programming. He can be contacted at email: khaledbalhaf2021@gmail.com

Ahmed Saleh Khaled was born in May 1981 in Lahj Republic of Yemen. Completed secondary education in Taiz 2000 and earned his bachelor’s degree in computer science from the University Science of Technology- in 2004- Taiz and completed Master from Arab Academy 2023. He is currently pursuing a Ph.D. in Information Technology at the University of Science & Technology, Aden. He is contacted at: aalhurdi@gmail.com.

Mohammed Hassan Bin Shamlan is a PhD student in Information Technology at the University of Science and Technology, Aden. He received his Master’s degree in Information Technology in 2020 and his Bachelor’s degree in Computer Engineering

from Hadramout University, Yemen, in 2013. His research interests include reverse engineering, artificial intelligence, and cybersecurity. He can be contacted at: icdlmukalla@gmail.com.