International Journal of Innovation and Modern Applied Science

Vol. 11 Issue 1, December 2025

Doi :

ISSN No.:

A Review of Malware Analysis Techniques: Static, Dynamic, Forensic, and Hybrid Approaches

HAITAM AOUAYEJ [1], JAAFAR ABOUCHABAKA [2], SALMANE BOUREKKADI [3]

[1]IBN TOFEIL University , Laboratory of Computer Science Research, Kenitra, Morocco

ABSTRACT: The continuous evolution of malware represents a major challenge for modern cybersecurity. Attackers increasingly rely on obfuscation, encryption, and anti-analysis techniques to evade detection, rendering traditional signature-based defenses insufficient. In this context, malware analysis has become essential to understand threats, extract indicators of compromise, and support incident response. This manuscript provides a structured review of the main malware analysis techniques, organized into four complementary levels: triage, static analysis, dynamic analysis, and advanced reverse engineering. Each method is discussed in terms of methodology, tools, advantages, and limitations. The review highlights that static analysis provides rapid and safe insights but is ineffective against sophisticated obfuscation and packing methods. Dynamic analysis captures runtime behaviors and network communications, though it remains vulnerable to evasion strategies. Memory and network forensics enhance visibility by revealing hidden processes, fileless malware, and anomalous connections. Reverse engineering offers the most comprehensive understanding of malicious logic, yet it is highly time-consuming and requires specialized expertise. Overall, the findings confirm that no single method is sufficient; instead, hybrid approaches that integrate static, dynamic, and forensic techniques are required. The study also emphasizes the role of standardized frameworks and large-scale datasets in improving reproducibility and enabling automation in malware detection and analysis.

Keywords: malware analysis, static analysis, dynamic analysis, reverse engineering, memory forensics, network forensics, sandboxing, incident response, MITRE ATT&CK.

1. INTRODUCTION

Malware has become one of the most prevalent threats in modern cyberspace, targeting individuals, enterprises, and critical infrastructures. Its rapid evolution, driven by techniques such as obfuscation, encryption, and the use of living-off-the-land binaries (LOLBins), significantly complicates detection and defense. Traditional signature-based systems are increasingly insufficient, which underlines the importance of malware analysis as a cornerstone of cybersecurity.

The purpose of this study is to provide a structured review of malware analysis methodologies, synthesizing results from both academic research and practitioner experience. By examining multiple layers—static analysis, dynamic analysis, memory and network forensics, and advanced reverse engineering—this paper identifies strengths, weaknesses, and applicability across different contexts. Static approaches offer rapid insights but are limited against modern evasion techniques, while dynamic methods capture behavioral patterns at runtime but face challenges from VM detection (Artem Dinaburg, 2008). Memory and network analysis complement these approaches by providing visibility into runtime artifacts and malicious communications. Reverse engineering remains indispensable for advanced persistent threats (APTs), offering deep technical insights at the cost of time and expertise (Yan Shoshitaishvili).

By aligning the analysis with standardized frameworks such as MITRE ATT&CK (MITRE. (2023). MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge., n.d.) and leveraging datasets including EMBER (Hyrum S. Anderson) and SOREL-20M (Richard Harang), this review aims to contribute reproducible methodologies and comparative insights for both academic researchers and security operations teams.

2. MATERIALS AND METHODS

This review was conducted through a systematic survey of academic literature, technical reports, and practitioner blogs related to malware analysis. The methodology was designed to ensure reproducibility and completeness, and it followed four complementary stages: triage, static analysis, dynamic analysis, and advanced reverse engineering.

Literature Search Strategy. Academic articles, conference proceedings, and technical whitepapers published between 2010 and 2024 were selected from digital libraries such as IEEE Xplore, ACM Digital Library, and arXiv. Keywords including “malware static analysis,” “dynamic analysis,” “memory forensics,” and “reverse engineering malware” guided the search. The inclusion criteria prioritized papers that described methodologies, comparative studies of tools, and practical case studies (Daniele Ucci), (L. Nataraj).

Tools and Environments. Widely adopted frameworks and toolkits were examined to illustrate the implementation of analysis methods. For static analysis, tools such as IDA Pro, Ghidra, Binary Ninja, and capa were included (Hex-Rays. (2023). IDA Pro disassembler and debugger., n.d.), (National Security Agency. (2019). Ghidra Software Reverse Engineering Framework., n.d.). Dynamic analysis relied on sandbox environments such as FlareVM and REMnux, complemented with monitoring tools like Procmon, Sysmon, and Wireshark (Case Andrew). Memory forensics tools such as Volatility and Rekall were integrated for the extraction of runtime artifacts (Case Andrew).

Reference Frameworks and Datasets. To standardize classification, the MITRE ATT&CK framework was adopted as a mapping reference for tactics and techniques (MITRE. (2023). MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge., n.d.). Public malware datasets such as EMBER (Hyrum S. Anderson), SOREL-20M (Richard Harang), and BODMAS were included to support machine learning-based evaluations.

Synthesis Approach. Each method was described along four axes: methodology, tooling, advantages, and limitations. Comparative tables were used to highlight differences across approaches, while case studies provided practical context. This structure ensures that the findings are accessible to both academic

researchers and professionals in Security Operations Centers (SOC) and Digital Forensics and Incident Response (DFIR) teams.

3. RESULTS AND DISCUSSION

This section may each be divided by subheadings or may further divided into next heads as shown below.

3.1. Static Analysis

Static analysis refers to the examination of malware binaries without execution. Techniques include string extraction, header inspection, entropy measurement, and disassembly. These methods enable analysts to identify imported libraries, suspicious API calls, and embedded resources (Hex-Rays. (2023). IDA Pro disassembler and debugger., n.d.), (National Security Agency. (2019). Ghidra Software Reverse Engineering Framework., n.d.). Tools such as IDA Pro, Ghidra, Binary Ninja, and capa remain the most widely adopted for this purpose.

The advantages of static analysis lie in its speed, safety, and ability to provide initial insights. It allows the identification of potential persistence mechanisms, cryptographic functions, and obfuscation indicators (Daniele Ucci), (L. Nataraj). Moreover, the extraction of features such as control flow graphs and opcode frequencies support the application of machine learning models for malware classification (Hyrum S. Anderson). YARA rules are also commonly generated from static signatures, enabling rapid detection in SOC environments.

However, static methods face critical limitations. Packed or encrypted samples often hide their true behavior, rendering static disassembly incomplete or misleading. Furthermore, advanced obfuscation techniques such as control-flow flattening and string encryption reduce the interpretability of static results. These drawbacks emphasize the need for complementary methods. In conclusion, static analysis remains indispensable for triage and IOC extraction but cannot serve as a standalone technique in modern threat landscapes.

3.2. Dynamic Analysis

Dynamic analysis involves executing malware in a controlled environment to observe its behavior at runtime. Sandboxing platforms such as FlareVM, REMnux, and ANY.RUN, combined with tools like Procmon, Sysmon, and Wireshark, are widely used to capture changes in the file system, registry, processes, and network traffic (Case Andrew).

The results show that dynamic methods are highly effective for detecting persistence mechanisms, command-and-control (C2) communications, and decrypted payloads (MITRE. (2023). MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge., n.d.), (Case Andrew). API call monitoring and system call tracing provide detailed behavioral fingerprints that can be used for clustering malware families. These observations are often mapped to the MITRE ATT&CK framework to classify adversarial techniques (MITRE. (2023). MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge., n.d.).

Nonetheless, dynamic analysis suffers from significant challenges. Modern malware increasingly integrates anti-analysis mechanisms, such as VM detection, timing delays, and environment fingerprinting, which significantly reduce the reliability of sandbox-based results (Artem Dinaburg, 2008). In addition, certain behaviors are triggered only under specific conditions, requiring interactive or guided execution.

Critically, dynamic analysis provides a realistic view of runtime activities but requires hardened environments. Countermeasures such as stealth instrumentation, bare-metal sandboxes, and hardware-assisted monitoring are being explored to bypass evasion techniques (Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiaoyong Zhou, XiaoFeng Wang). While highly valuable, dynamic analysis must be carefully integrated with static and forensic methods to deliver comprehensive insights.

3.3 Memory and Network Forensics

Memory forensics has emerged as a powerful technique to detect hidden malware components, fileless infections, and runtime artifacts that evade traditional methods. Tools such as Volatility and Rekall allow investigators to reconstruct process lists, extract injected code, and analyze kernel modules (Case Andrew). Similarly, memory acquisition tools like WinPMem and LiME are essential for capturing evidence during live incident response.

Research demonstrates that memory analysis is particularly effective for uncovering in-memory persistence mechanisms and retrieving decrypted configurations. Fileless malware campaigns, which rely on PowerShell or WMI for execution, can often only be detected through memory snapshots.

In parallel, network forensics focuses on analyzing traffic generated by malware. Tools like Wireshark and Zeek enable analysts to detect anomalies, C2 protocols, and domain fronting techniques (Case Andrew). The correlation of captured traffic with MITRE ATT&CK techniques supports standardized threat intelligence reporting (MITRE. (2023). MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge., n.d.).

Despite their effectiveness, both memory and network analysis face limitations. Memory forensics requires specialized expertise and careful handling to avoid altering evidence (Michael Hale Ligh, Andrew Case, Jamie Levy, AAron Walters). Network analysis is constrained when malware uses encrypted channels, covert DNS tunneling, or fast flux domains. Nonetheless, these approaches play a central role in DFIR by bridging the gap between static/dynamic observations and real-world artifacts.

3.4 Reverse Engineering and Anti-analysis

Reverse engineering represents the most comprehensive yet complex approach to malware analysis. By disassembling or decompiling binaries, analysts can reconstruct malicious logic and understand its full capabilities. Tools such as IDA Pro, Ghidra, Radare2, x64dbg, and WinDbg are essential for this process (Hex-Rays. (2023). IDA Pro disassembler and debugger., n.d.), (National Security Agency. (2019). Ghidra Software Reverse Engineering Framework., n.d.).

This method enables the extraction of encryption algorithms, configuration decryption routines, and command execution logic (Daniele Ucci), (Thomas Raffetseder, Christopher Kruegel & Engin Kirda). Advanced techniques such as symbolic execution and taint analysis assist in uncovering hidden behaviors, while automated unpackers facilitate the analysis of packed binaries.

However, reverse engineering is highly time-consuming and requires significant expertise. Malware authors actively implement anti-analysis techniques such as debugger detection, control-flow obfuscation, and API redirection (Artem Dinaburg, 2008). For example, stealth breakpoints have been observed in advanced samples to mislead analysts during debugging.

From a critical perspective, reverse engineering is indispensable for APT investigations, malware family attribution, and threat intelligence. Yet, due to its complexity, it is impractical for large-scale automation. Future perspectives point toward AI-assisted reverse engineering and improved automated deobfuscation frameworks, which may help reduce manual workloads (Yan Shoshitaishvili).

3.5 Hybrid and Automated Analysis

Hybrid analysis has been proposed as a solution to overcome the limitations of standalone methods. By combining static feature extraction with dynamic behavior monitoring, hybrid approaches provide a more holistic understanding of malware. For instance, static signatures can rapidly classify known families, while dynamic execution reveals runtime payloads and network indicators.

Automated pipelines further enhance this process by integrating multiple tools into unified workflows. Cloud-based sandboxes allow scalability and collaborative analysis, while frameworks such as Cuckoo Sandbox demonstrate how hybrid models can operate in practice.

The role of artificial intelligence has also expanded significantly. Machine learning models trained on large datasets such as EMBER (Hyrum S. Anderson) and SOREL-20M (Richard Harang) achieve promising results in predicting malicious behavior. Deep learning approaches leverage opcode sequences, control flow graphs, and API call graphs to detect unknown malware variants with high accuracy.

Nevertheless, hybrid and AI-based approaches face critical challenges, including adversarial evasion, model interpretability, and the need for constant retraining with updated datasets. Despite these issues, hybrid and automated methods represent the future of malware analysis, enabling SOCs and DFIR teams to scale operations and improve detection.

3.6 Cross-method Discussion

The comparative analysis across static, dynamic, forensic, and reverse engineering approaches demonstrates that no single method can provide a complete solution to modern malware threats. Each technique offers unique insights but also exhibits inherent limitations in terms of accuracy, scalability, and practicality. To better illustrate these findings, the results of the review are organized into two complementary comparative tables.

Table 1. Malware Analysis Methods

Method	Example Tools	Advantages	Limitations	References
Static Analysis	IDA Pro, Ghidra, Binary Ninja, capa	Fast, safe, allows initial classification, signature/IOC extraction	Ineffective against obfuscation, packing, encryption	(Daniele Ucci), (L. Nataraj), (Hex-Rays. (2023). IDA Pro disassembler and debugger., n.d.), (National Security Agency. (2019). Ghidra Software Reverse Engineering Framework., n.d.), (Hyrum S. Anderson), (Thomas Raffetseder, Christopher Kruegel & Engin Kirda)

Dynamic Analysis	FlareVM, REMnux, Procmon, Sysmon, Wireshark, ANY.RUN	Observes real behavior, detects persistence, network traffic, decrypted payloads	Vulnerable to anti-VM/sandbox detection, requires strong isolation	(MITRE. (2023). MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge., n.d.), (Artem Dinaburg, 2008), (Case Andrew), (Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiaoyong Zhou, XiaoFeng Wang)
Memory Forensics	Volatility, Rekall	Detects fileless malware, recovers runtime artifacts and hidden processes	Complex acquisition, requires expertise and advanced tooling	(Case Andrew), (Michael Hale Ligh, Andrew Case, Jamie Levy, AAron Walters)
Network Forensics	Wireshark, Zeek	Identifies C2 protocols, anomalies, encrypted tunnels	Limited visibility if malware uses covert channels or strong encryption	(MITRE. (2023). MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge., n.d.)
Reverse Engineering	x64dbg, WinDbg, Radare2, Ghidra	Deep understanding of malware logic, enables unpacking and decryption	Time-consuming, high expertise required, frequent anti-analysis mechanisms	(Daniele Ucci), (Hex-Rays. (2023). IDA Pro disassembler and debugger., n.d.), (National Security Agency. (2019). Ghidra Software Reverse Engineering Framework., n.d.), (Thomas Raffetseder, Christopher Kruegel & Engin Kirda), (Artem Dinaburg, 2008), (Yan Shoshitaishvili)

Table 1 presents the technical perspective by summarizing the main tools, strengths, and weaknesses of each method. This offers a structured view of how different techniques operate at the analytical level. Table 2 extends this discussion to the operational domain by evaluating scalability, automation potential, expertise requirements, and applicability within SOC and DFIR environments. Together, these tables bridge the gap between theoretical capabilities and real-world usability, providing both researchers and practitioners with a dual framework for understanding and applying malware analysis techniques.

Table 2. Comparative Evaluation of Malware Analysis Methods

Method	Scalability	Automation	Expertise Required	SOC/DFIR Relevance	References
Static Analysis	High	High	Low to Moderate	Strong for triage, IOC extraction	(Daniele Ucci), (L. Nataraj), (Hex-Rays. (2023). IDA Pro disassembler and debugger., n.d.), (National Security Agency. (2019). Ghidra Software Reverse Engineering Framework., n.d.), (Hyrum S. Anderson), (Thomas Raffetseder, Christopher Kruegel & Engin Kirda)
Dynamic Analysis	Moderate	Moderate	Moderate	Strong for behavior profiling, C2 detection	(MITRE. (2023). MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge., n.d.), (Artem Dinaburg, 2008), (Case Andrew), (Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiaoyong Zhou, XiaoFeng Wang)
Memory Forensics	Low	Low	High	Essential for fileless malware and live response	(Case Andrew), (Michael Hale Ligh, Andrew Case, Jamie Levy, AAron Walters)
Network Forensics	Moderate	Moderate	Moderate	Valuable for C2 detection, lateral movement analysis	(MITRE. (2023). MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge., n.d.)
Reverse Engineering	Very Low	Very Low	Very High	Indispensable for APT attribution, threat intelligence	(Daniele Ucci), (Hex-Rays. (2023). IDA Pro disassembler and debugger., n.d.), (National Security Agency. (2019). Ghidra Software Reverse Engineering Framework., n.d.), (Thomas Raffetseder, Christopher Kruegel & Engin Kirda), (Artem Dinaburg, 2008), (Yan Shoshitaishvili)
Hybrid/Automated	High	High	Moderate	Crucial for modern SOC	(Hyrum S. Anderson), (Richard Harang), (Clemens

				pipelines and scalable detection	Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiaoyong Zhou, XiaoFeng Wang), (Yan Shoshitaishvili)

Table 2 provides a comparative evaluation of the different malware analysis methods, considering practical criteria such as scalability, automation potential, expertise requirements, and relevance to SOC and DFIR operations. The results show that static analysis is highly scalable and easily automated, making it an effective first step in large-scale triage workflows. Dynamic and network forensics offer stronger behavioral insights but require moderate expertise and are only partially scalable due to the need for controlled environments. Memory forensics, while indispensable for fileless malware detection, remains resource-intensive and requires advanced expertise, limiting its scalability in operational contexts. Reverse engineering delivers unparalleled insights but is the least scalable and most expertise-dependent method, making it suitable primarily for high-profile investigations such as APT campaigns. Hybrid and automated approaches emerge as the most promising direction, balancing scalability and accuracy by combining static, dynamic, and forensic techniques. Their integration into SOC pipelines provides both operational efficiency and resilience against evolving threats.

4. CONCLUSION

This review provided a comprehensive synthesis of malware analysis techniques, structured around static analysis, dynamic execution, memory and network forensics, reverse engineering, and hybrid approaches. The findings confirm that each method has specific advantages and inherent limitations. Static analysis is rapid and scalable but easily bypassed by obfuscation, while dynamic methods reveal runtime behavior but remain vulnerable to evasion. Memory and network forensics uncover hidden artifacts and communications but require advanced expertise. Reverse engineering delivers the most detailed understanding yet is highly resource-intensive.

The comparative evaluation, supported by two complementary tables, highlights the duality between technical capabilities and operational applicability. Static and dynamic methods are essential for large-scale triage, whereas memory forensics and reverse engineering are best suited for in-depth incident response and threat intelligence. Hybrid and automated analysis emerge as the most promising direction, combining the strengths of each method while mitigating their individual weaknesses.

In conclusion, the study underscores the importance of layered, automated, and standardized malware analysis pipelines that integrate multiple techniques and leverage frameworks such as MITRE ATT&CK and large-scale datasets. Such integration is vital to strengthen the resilience of Security Operations Centers and DFIR teams against increasingly sophisticated threats.

ACKNOWLEDGEMENTS

This work acknowledges the contributions of the research community in the field of malware analysis, whose tools, frameworks, and datasets provided the foundation for this review. In particular, appreciation is extended to the developers of IDA Pro, Ghidra, FlareVM, REMnux, Volatility, and MITRE ATT&CK, whose efforts have advanced the practice of static, dynamic, forensic, and reverse engineering methodologies. The availability of open-access datasets such as EMBER and SOREL-20M has also been instrumental in supporting the development and evaluation of machine learning–based approaches. The synthesis presented in this paper benefits greatly from the collective knowledge shared by academics, practitioners, and organizations dedicated to improving cybersecurity.

REFERENCES

Artem Dinaburg, P. R. (2008). Ether: malware analysis via hardware virtualization extensions.

Case Andrew, G. G. (n.d.). Memory forensics: The path forward.

Clemens Kolbitsch, Paolo Milani Comparetti, Christopher Kruegel, Engin Kirda, Xiaoyong Zhou, XiaoFeng Wang. (n.d.). Effective and Efficient Malware Detection at the End Host.

Daniele Ucci, L. A. (n.d.). Survey of machine learning techniques for malware analysis.

Hex-Rays. (2023). IDA Pro disassembler and debugger. (n.d.). Retrieved from https://hex-rays.com/

Hyrum S. Anderson, P. R. (n.d.). EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models.

L. Nataraj, S. K. (n.d.). Malware images: visualization and automatic classification.

Michael Hale Ligh, Andrew Case, Jamie Levy, AAron Walters . (n.d.). *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory.*

MITRE. (2023). MITRE ATT&CK: Adversarial tactics, techniques, and common knowledge. (n.d.). Retrieved from https://attack.mitre.org/

National Security Agency. (2019). Ghidra Software Reverse Engineering Framework. (n.d.). Retrieved from https://github.com/NationalSecurityAgency/ghidra

Richard Harang, E. M. (n.d.). SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection.

Thomas Raffetseder, Christopher Kruegel & Engin Kirda . (n.d.). Detecting System Emulators.

Yan Shoshitaishvili, R. W. (n.d.). SOK: (State of) the Art of War: Offensive Techniques in Binary Analysis.