

Analysing packed and encrypted malware using binary analysis

¹Disina, A. H., ¹Adamu, A., ²Mohammad, N., ¹Gangkwi, B. J., and ³Yunusa, A. A.

¹Cyber Security Department, Nigerian Army University Biu, Borno State Nigeria

²Department of Electronics and Telecommunication Engineering, Airforce Institute of Technology, Kaduna State Nigeria

³Computer Science department, Nigerian Army University Biu, Borno State Nigeria

ningi2200@naub.edu.ng, muhdbaba@gmail.com, bbgangwi@gmail.com, amirahmad22@naub.edu.ng

Paper History

Received: 1st October, 2025

Accepted: 20th Nov., 2025

Published: December, 2025

Abstract:

The increasing use of packing and encryption techniques in malware poses a major challenge to traditional signature-based antivirus systems. The aim of this study presents a lightweight static analysis framework for detecting executables that employ evasion strategies such as packing and encryption. The methodology integrates entropy measurement, section header analysis, opcode frequency profiling, and import table inspection to identify concealed malicious behaviour without executing the sample. The system was implemented in Python using PEfile, Capstone, Tkinter, and Matplotlib. Testing with benign, packed, and encrypted executables demonstrated effective differentiation between normal and obfuscated binaries, achieving an overall detection accuracy of 91.7%. The proposed system provides a transparent and resource-efficient solution suitable for academic environments, malware research, and preliminary digital forensic investigations. This research will reduce the threats posed by the malwares, adversaries and will ensure safety and security in the digital world.

Corresponding author

Disina, A. H.

disina.hassan@naub.edu.ng

Keywords: Binary analysis, Malware detection, Opcode profiling, Static analysis

1. Introduction

Malware continues to evolve rapidly within modern computing environments, creating increasingly complex challenges for cybersecurity defence. Traditional antivirus solutions have relied heavily on signature-based detection in which executable files are compared with known malicious patterns [1–3]. However, this approach has become significantly less effective, as contemporary malware frequently adopts sophisticated evasion strategies such as packing, encryption, polymorphism, and structural obfuscation to disguise malicious code during inspection. These transformations alter binary appearance while retaining the underlying behaviour, thereby preventing signature-based scanners from identifying malicious intent during static inspection [4].

Advanced ransomware families, including LockBit, Conti, Royal and BlackCat (ALPHV), provide notable examples of this trend. These malware families often generate thousands of polymorphic variants from the same malware code base by applying multi-layered packing and encryption routines. These procedures increase binary entropy, modify section structures, and reduce visible import references, rendering traditional detection increasingly ineffective [5, 6]. As a result, modern malware detection techniques must evolve beyond pattern recognition and instead examine underlying executable characteristics to determine malicious intent [3, 7].

Machine learning and deep learning approaches have shown promising results in detecting malware based

on opcode sequences, entropy values, and import table characteristics. Researchers have demonstrated that these approaches can classify malware with high accuracy using both shallow learning and deep neural models [8–10]. However, such approaches often require large-scale datasets, substantial computational resources, and lack clarity in result interpretation, making them unsuitable in many practical environments including incident response operations, forensic investigations, and academic settings where transparency and simplicity are essential.

Static binary analysis has therefore gained attention as a reliable, transparent and efficient approach for malware inspection. Static analysis does not require file execution, thereby eliminating safety concerns associated with dynamic sandboxing, while enabling measurable inspection of features such as Shannon entropy, opcode frequency, section metadata, and import behaviour [7, 10]. While these features can improve detection reliability, relying on a single metric such as entropy frequently results in false alarms, since legitimate compression tools such as UPX and PyInstaller also significantly increase binary randomness [10].

In response to these limitations, this paper proposes a hybrid static malware detection framework that correlates multiple binary-level indicators, including per-section entropy, opcode distribution behaviour, and section consistency and import irregularities to improve detection accuracy and reduce false positives [11, 12]. The framework provides a lightweight, fast, and transparent detection mechanism suitable for educational institutions,

research environments and low-resource operational spaces where malware execution is unsafe or impractical [9].

1.1 Analysis of existing system

Traditional malware detection systems, especially antivirus products, rely extensively on signature-based detection [5, 6]. This approach depends on comparing executable files against a database of known malware patterns. While effective in earlier decades, this strategy fails against contemporary polymorphic malware that modifies binary structure on each compilation to avoid recognition. This limitation has led to high false-negative rates when addressing malware families such as LockBit and ALPHV, which constantly regenerate new variants derived from the same code base [13, 14]. Moreover, signature-based systems typically perform shallow inspection and do not analyse deeper structural properties such as section relationships, import complexity or instruction-level behaviour [15]. Dynamic analysis systems attempt to address this challenge by executing malware in controlled environments; however, these methods require computationally expensive sandboxes and can still be bypassed by anti-virtual machine or anti-debugging fragments integrated into modern malware [11, 12].

2. Methodology

The study adopts a static binary analysis methodology to detect malware employing packing and encryption as evasion techniques. Static analysis involves examining executable files without executing them, making the approach safe, efficient and suitable for academic environments and systems where malware execution would pose significant operational risks. The proposed methodology combines multiple static indicators, including entropy measurement, section metadata inspection, and opcode distribution profiling and import table analysis, to create a hybrid model capable of accurately distinguishing maliciously packed malware from legitimate executables. This research introduces a hybrid static analysis model that evaluates critical binary features of Portable Executable (PE) files to identify indicators of packing, encryption and other obfuscation mechanisms. The system organizes processing into distinct functional modules:

2.1 Input module

This module accepts executable samples selected by the user and validates them as PE-formatted binaries before further processing.

2.1.1 Feature extraction module

The feature extraction module retrieves multiple binary properties, including entropy levels, section sizes, import references and opcode distribution. These properties are used as the basis for structural evaluation and comparative analysis.

2.1.2 Analysis engine

Extracted features are evaluated using threshold evaluation rules developed from industry and research

literature. For example, entropy values above 7.0 typically indicate the presence of encryption or packing, while files with minimal import calls may rely on runtime resolution, another known malware trait.

2.1.3 Decision and reporting layer

Based on combined analysis of entropy, structure, import behaviour and opcode patterns, the system classifies files into categories such as:

- a. Benign
- b. Packed but non-malicious
- c. Suspicious and potentially malicious

Reports are displayed to the user via a graphical interface and can be exported for additional laboratory or forensic processing.

2.2 System architecture

The proposed system operates using a modular architecture that enables analytical components to act independently while contributing to an aggregated decision model. This allows new analytical indicators to be incorporated without restructuring the entire system. The architecture follows a sequential workflow beginning with file selection, followed by feature extraction, static evaluation, classification and structured reporting.

2.3 Detection algorithm

The hybrid detection model applies conditional logic to determine whether binary features exceed suspicious thresholds. Simplified representations of the algorithm include:

Step 1: Load executable file

Step 2: Extract entropy, opcode distribution, section metadata and imports

Step 3: Evaluate entropy for each section (high entropy >7.0 is marked abnormal)

Step 4: Check import library frequency and the presence of low-level APIs often associated with process injection

Step 5: Evaluate opcode patterns for repetitive decryption routines

Step 6: Correlate feature outcomes

Step 7: Generate final classification report

2.4 Justification for method

The proposed method represents a balance between accuracy, safety and practicality. Previous research has shown that static indicators such as entropy values, opcode concentration, and structural anomalies provide clear signals of packing or encryption in executable files [8]. Unlike dynamic analysis, this approach does not require execution of malicious code, thereby reducing operational risk. Additionally, the system maintains explainability, which is important for malware forensics, classroom use and environments with limited processing resources.

2.5 System implementation

The framework was developed in Python using PEfile for binary parsing, Capstone for disassembly, and Tkinter

for a graphical user interface. The system accepts Portable Executable (PE) files as input and performs static analysis without executing the samples. This ensured safe operation even when handling files exhibiting malware-like properties. The interface includes modules for browsing files, executing analysis, saving results and displaying structural statistics in a visual format. The system also generates JSON reports for further investigation or dataset generation.

3. Results

This section presents the outcomes of the implemented malware detection framework and discusses how the results validate the system's effectiveness in detecting packed and encrypted malware samples. The framework was applied to 60 executable files, consisting of 20 benign executables, 20 packed samples and 20 encrypted variants. The results were analysed using entropy measurements, opcode statistics, and structural metadata and import table characteristics.

3.1 Entropy analysis results

Entropy was calculated for each section of the test executables. Benign files exhibited typical entropy values between 4.5 and 6.5, consistent with normal application code. Packed and encrypted samples demonstrated significantly higher entropy values, often ranging from 7.0 to 8.0, which is characteristic of compression or encryption. High entropy is widely recognized as one of the most reliable indicators of packing or concealed binary structures [4], [15].

Table 1: Entropy analysis results (sample)

File type	Typical entropy range	Interpretation
Benign files	4.5 – 6.5	Normal executable structure
Packed files	7.0 – 8.0	Indicates compression or packing
Encrypted files	7.2 – 8.0+	Suggests encryption or high obfuscation

Results demonstrated that files exhibiting entropy values above 7.0 were automatically marked as suspicious by the system. This behaviour is consistent with accepted malware analysis literature and supports the system's threshold logic.

3.2 Opcode analysis results

Opcode distribution profiles revealed behavioural differences between normal and packed or encrypted executables. Benign software displayed balanced opcode distributions dominated by common machine instructions such as mov, call, push and cmp. Packed and encrypted files showed compressed opcode ranges and unusually high counts of instructions such as xor, int3, and unconditional jumps. These opcode patterns are consistent with unpacking loops, decoder stubs or anti-analysis padding seen in malware obfuscation [11, 12].

The presence of excessive xor instructions indicates possible runtime decryption loops, while inflated int3

presence typically suggests padding or debugging interference, behaviours commonly associated with malware stub execution.

Table 2: Opcode distribution comparison

Opcode	Benign count	Packed count	Encrypted count
Mov	214	190	120
Xor	39	80	112
int3	0	324	380
Call	46	70	85

3.3 Import table analysis

Benign executables were found to import a wide range of libraries and functions such as USER32.dll, GDI32.dll, and standard runtime components. In contrast, packed or encrypted samples frequently showed minimal imports, often limited to a few KERNEL32.dll routines. This pattern reveals reliance on runtime API resolution, a widely documented malicious technique for evading static analysis tools. The flagged samples that imported suspicious functions including Write Process Memory, Virtual Alloc, and Create Remote Thread, which are commonly used in code injection and process hijacking.

3.4 Combined classification performance

By correlating entropy, opcode profiles, structural metadata, and import table observations, the research classified the 60 test samples as follows:

Table 3: Detection accuracy summary

Sample type	Total samples	Correctly classified	Accuracy (%)
Benign	20	18	90
Packed	20	19	95
Encrypted	20	18	90
Overall accuracy	60	55	91.7

An overall accuracy of 91.7% demonstrates that the hybrid static model was able to detect packed and encrypted malware effectively without requiring sample execution. This confirms existing literature which identifies entropy and opcode-based analysis as reliable detection features when applied in combination.

4. Discussion

Analysis showed that combining multiple static features significantly reduced false positives compared to single-metric approaches. Files exhibiting two or more suspicious characteristics, for example, high entropy and reduced imports were consistently classified accurately, reinforcing the strength of the hybrid model.

While some legitimate commercial software also utilizes packing and encryption for intellectual property protection, the system's correlation-based decision model helped distinguish clearly between normal compression and genuinely suspicious behaviour. The system's transparent decision rules further support malware diagnostics, digital forensics and academic training, where explainability is essential.

5. Conclusion

The study presented a hybrid static malware detection framework designed to identify malicious executables employing packing and encryption as evasion mechanisms. Traditional signature-based detection methods depend heavily on matching file contents with known malware patterns, but modern malware families increasingly modify their binary structure to avoid recognition. Through the combination of multiple static analysis features, including entropy measurement, opcode profiling, section metadata inspection and import analysis, the developed system provides a reliable method for detecting suspicious executables without executing the files.

Experimental results demonstrated that the system achieved an overall accuracy of 91.7% when tested on 60 executable samples comprising benign, packed and encrypted binaries. This performance confirms that multi-feature correlation significantly improves detection effectiveness compared to single-parameter approaches such as entropy-only evaluation. The transparency of the decision logic also makes the system suitable for forensics, malware research and educational environments where explainability is essential.

Despite its strengths, the system has some limitations. The model does not account for advanced malware capable of decrypting major payloads only during runtime, nor does it perform behavioural analysis that might reveal hidden execution paths. However, the system provides a foundation upon which more advanced hybrid solutions incorporating dynamic analysis, signature generation or machine learning can be developed in future work.

Recommendation

The study demonstrated that a hybrid static malware detection system can successfully identify executables employing packing and encryption as evasion strategies. However, further research is recommended to improve system robustness and operational versatility. Future work may incorporate dynamic behavioural analysis to capture malware that decrypts or unpacks its payload only during execution. Additionally, incorporating machine learning classification into the decision engine could improve detection accuracy against complex malware capable of blending legitimate and malicious behaviour. Expanding the dataset to include a wider range of malware families and file formats would also strengthen system evaluation and generalizability.

Acknowledgments

The authors acknowledge all individuals, institutions and laboratory environments that contributed resources, technical guidance and support toward the successful completion of this research.

Reference

- [1] Akhtar, M. S. and Feng, T., (2022). Malware Analysis and Detection Using Machine Learning Algorithms, *SS Symmetry*, 14(11), 1–11,.

- [2] Khalid, O., Ullah, S., Ahmad, T., Saeed, S., Alabbad, D. A., Aslam, M., Buriro, A. and Ahmad, R., (2023). An Insight into the Machine-Learning-Based Fileless Malware Detection, *Sensors*, 612(01)pp. 1–20, 2023.
- [3] Brezinski, K. and Ferens, K., (2023). Metamorphic Malware and Obfuscation: A Survey of Techniques, Variants, and Generation Kits, *Secur. Commun. Networks*, 2023, 18227751, 1–40. <https://doi.org/10.1155/2023/8227751>.
- [4] Maddireddy, B. R. and Maddireddy, B. R., (2023). Automating Malware Detection: A Study on the Efficacy of AI-Driven Solutions, *Rev. Intel. Artif. En Med.*, 14(01), 1287–1305
- [5] Javaheri, D., Lalbakhsh, P. and Hosseinzadeh, M., (2021). A Novel Method for Detecting Future Generations of Targeted and Metamorphic Malware Based on Genetic Algorithm, *IEEE Access*, 9, 69951–69970. <https://doi.org/10.1109/ACCESS.2021.3077295>.
- [6] Khoje, M. A., Kaswan, A. and Pawar, J., (2023). Exploring Polymorphic Malware Analysis Techniques: A Comparative Survey, *Journal of Renewable. Energy Exchange*, 11(5), 135–142.
- [7] Kouliaridis, V. and Kambourakis, G., (2021). A Comprehensive Survey on Machine Learning Techniques for Android Malware Detection, *Information*, 185(12), 1–12.
- [8] Karat, G., Kannimoola, J. M., Nair, N., Vazhayil, A., Sujadevi, V. G. and Poornachandran, P., (2024). CNN-LSTM CNN-LSTM Hybrid Hybrid Model Model for for Enhanced Enhanced Malware Malware Analysis Analysis and and Detection Detection, *Procedia Computer Science*, 233(03), 492–503. <https://doi.org/10.1016/j.procs.2024.03.239>.
- [9] Avhankar, M. S., Pawar, J. and Kumbhar, V., (2025). A Comprehensive Survey on Polymorphic Malware Analysis: Challenges, Techniques and Future Directions, *Communication on Applied Nonlinear Analysis*, 32(9s), 2765–2776.
- [10] Gorment, N. Z., Selamat, A., Cheng, L. K. and Krejcar, O., (2023). Machine Learning Algorithm for Malware Detection: Taxonomy, Current Challenges, and Future Directions, *IEEE Access*, 11(12), 141045–141089. <https://doi.org/10.1109/ACCESS.2023.3256979>.
- [11] Aslan, Ö., Ozkan-okay, M. and Gupta, D., (2021). A Review of Cloud - Based Malware Detection System: Opportunities, Advances and Challenges, *European Journal of Engineering and Technology Research*, 6(3), 1–8.
- [12] Imamverdiyev, Y. and Baghirov, E., (2024). Evasion techniques in malware detection: challenges and countermeasures, *Problems of Information Technology*, 15(2), 9–15.
- [13] Al-hashmi, A. A., Ghaleb, F. A. and Darem, A. A., (2022). Deep-Ensemble and Multifaceted Behavioral Malware Variant Detection Model, *IEEE Access*, 10, 42762–42777. <https://doi.org/10.1109/ACCESS.2022.3168794>.

- [14] Ilca, L. F., Lucian, P. and Balan, T. C., (2023). Enhancing Cyber-Resilience for Small and Medium-Sized and Response, *Sensors*, 23, 6757, 1–33.
- [15] Haque A. M., Ahmad, S., Sonal, D., Abdeljaber, H. A. M., Mishra, B. K., Eljaily, A. E. M., Alanazi, S., Nazeer, J., (2023). Achieving Organizational Effectiveness through Approaches for Malware Analysis and Detection Machine Learning Based Lograr la eficacia organizativa mediante enfoques basados en el aprendizaje automático para el análisis y la detección de malware, *Data Metadata*, 2, 139, 1–11.
<https://doi.org/10.56294/dm2023139>.