Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Robust Encoding and Secure Storage of Executables Using Image Based Encoding Techniques

Asharani R[1*],Dr.Vidyalakshmi K[2]

[1*]Research scholar, Department of CSE, Sri Siddhartha Institute of technology, Sri Siddhartha Academy of Higher Education, Tumakur 572105,Karnataka,India
[2]Assistant Professor, Department of CSE(DS) ,Sri Siddhartha Institute of Technology,Sri Siddhartha Academy of Higher Education, Tumakur 572105, Karnataka,India
* Corresponding author's **Email**: asharaniswamy@gmail.com

ARTICLE INFO

Received:04 Aug 2025
Revised:10 Sept 2025
Accepted:20 Sept 2025

ABSTRACT

The ever-increasing sophistication of malicious software poses significant hurdles to the field of cybersecurity, notably in the areas of malware detection and safe executable management.This paper discusses how malware is becoming more complicated and how that affects cybersecurity, especially when it comes to finding malware and managing executable files. It presents a hybrid dual-phase methodology that integrates a machine learning and deep learning-based malware detection system with a secure encoding framework intended to safeguard trusted executables. During the malware detection phase, static features like opcode sequences, API calls, and structural characteristics are taken out of Portable Executable (PE) files. We use feature optimization and k-fold cross-validation to make the system work better. The methodology assesses five algorithms: XGBoost, Random Forest, Gradient Boosting, Deep Learning (utilizing Keras DNN), and SVM (employing RBF Kernel). The performance metrics show that XGBoost has the highest accuracy (99.48%), F1-score (0.991), and AUC (0.9997), with Random Forest and Gradient Boosting not far behind. The Deep Learning model also does very well, with an accuracy of 99.04% and an AUC of 0.9992. This shows that it can recognize complex, non-linear patterns in malware activity. The proposed framework uses a multi-layered encoding system in the secure encoding phase. This system combines Base64 transformation, image-based mappings, and Modified Least Significant Bit (MLSB) embedding techniques. This encoding keeps trusted executables safe from tampering and unauthorized access. It has a 99.2% retrieval accuracy, which is better than traditional encryption methods when it comes to keeping data safe and private. In general, the proposed framework is a clear, scalable, and safe way to classify malware and protect executables. It has a lot of potential to be used in cybersecurity, especially for cloud infrastructures and important systems. The approach plays a big role in making AI-powered systems that can protect against a wide range of digital threats.

Keywords: Malware Detection, Executables, Machine Learning, Encoding, Cybersecurity, Text Encoding, Image Encoding, Base64, Image-based Mappings, Modified Least Significant Bit (MLSB) Embedding.

1. Introduction

The growth of malware presents a substantial risk to the integrity, confidentiality, and availability of digital systems. Malware capitalizes on flaws in executable files, frequently evading detection until it inflicts significant damage. Conventional signature-based detection approaches are inadequate for addressing complex and dynamic threats, requiring the use of advanced malware detection methodologies. Moreover, guaranteeing the security and validity of trusted executable files necessitates new encoding and analytical methods to alleviate dangers linked to tampering and illegal alterations.
This research tackles two significant challenges: (1) achieving high-accuracy malware detection in executables with machine learning approaches, and (2) encoding and analyzing trusted executables for secure storage and

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

retrieval. This work seeks to improve malware detection and executable file security through the integration of artificial intelligence and resilient encoding techniques.

The increasing complexity of malware variants and their capacity to evade traditional detection methods highlight the pressing necessity for sophisticated, feature-oriented classification systems. Furthermore, trustworthy executables necessitate safe encoding methods to guarantee their integrity, even in adversarial settings. Current methodologies frequently struggle to reconcile computing efficiency, storage demands, and resilience to manipulation.

The proliferation of malware poses a significant threat to digital systems, as it exploits vulnerabilities in executable files. Traditional signature-based detection methods fail against advanced, polymorphic, and zero-day threats, necessitating feature-driven machine learning models. To secure trusted executables against tampering and unauthorized modifications, innovative encoding techniques are needed. This study addresses two critical challenges in cybersecurity: achieving high-accuracy malware detection in executables using machine learning with extracted opcode sequences, API calls, and byte-level attributes, and encoding and analyzing trusted executables for secure storage and tamper resistance using Base64, image-based encoding, and MLSB embedding.

Existing limitations in malware detection include lack of generalization in signature-based and heuristic models, high false positives in anomaly-based detection, limited adaptability to novel malware variants, computational overhead in encryption-based methods, susceptibility to tampering in basic encoding techniques, and storage inefficiency with large-scale datasets.

The proposed solution focuses on machine learning for malware detection, extracting relevant features from executables and applying feature selection to remove noise. The model outperforms other classifiers with high detection accuracy, indicating their robustness against adversarial malware variants. The encoding method compared to Pixel Value Mapping and Modified LSB Image-Based Encoding showed better compression, faster encoding and decoding times, and stronger resistance against tampering.

This approach bridges the gap in malware detection by leveraging machine learning-driven feature selection and classification, improving detection against zero-day malware, enhancing executable security through encoding, and reducing false positives in malware classification while securing trusted executables.

Objectives of the research work as follows:

1. Construct a machine learning framework for identifying malware in executables utilizing extracted information, including opcode sequences, API calls, and byte-level attributes.
2. Encode trusted executable files into forms appropriate for secure storage and efficient retrieval utilizing Base64, image-based encoding, and sophisticated methods such as MLSB embedding encoding.
3. Evaluate and contrast encoding strategies for efficiency, retrieval precision, and resilience to tampering or corruption.

This study is significant in cybersecurity and digital forensics, where detecting and mitigating malware attacks is essential. The amalgamation of feature-based classification methods with secure encoding protocols offers a holistic strategy for executable file security. This research presents a scalable method for malware identification and the safeguarding of trustworthy executables across diverse settings, including critical infrastructure, by integrating machine learning techniques with creative encoding strategies.

The subsequent sections of this work are structured as follows. Section 2 offers a literature review and comparative studies. Section 3 delineates the Proposed Methodology. Section 4 examines the Results and findings. Ultimately, Section 5 serves as the conclusion segment.

2. Literature Review

In this section authors discussed the previous work and literature review and its advantages and disadvantages of the work.

2.1 Reviews on Steganography and Malware Detection

Teaching offensive security [1], particularly ethical hacking, is crucial in information security curricula to equip cybersecurity professionals with the knowledge to protect systems from attacks. Understanding potential vulnerabilities allows for early detection and proactive defense strategies. This method raises awareness among browser developers about potential risks associated with handling images. However, limitations include potential ethical concerns, the risk of misuse of knowledge, and the reliance on specific technology, such as image steganography, which may not cover all cybersecurity threats.

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Research Article

The article [2] explores the methods and challenges of malware distribution, particularly through file attachments in phishing emails and illegitimate downloads. It highlights the effectiveness of existing security applications using signature-based and anomaly-based machine learning techniques for detecting malware in common file formats. However, detecting malware hidden within multimedia files using steganography remains problematic; as such instances are infrequent and often serve as a preliminary step in sophisticated cyberattacks. The article aims to fill the knowledge gap in the intersection of image steganography and stegomalware detection, detailing the history, generation tools, and current advancements in image steganography techniques. However, limitations include a lack of comprehensive empirical data, limited discussion on countermeasures against stegomalware, and the focus on specific multimedia types may overlook other forms of steganographic attacks.

The Robust Malicious Executable Detection (RMED) [3] system uses machine learning classifiers to identify malicious Portable Executable (PE) files on Windows operating systems. The system uses a dataset of 116,031 benign files and 179,071 malware samples, focusing on specific PE headers. The model is trained on 15 PE features, achieving an accuracy of 98.42% and a false positive rate of 1.58%. The RMED aims to improve cybersecurity measures by implementing AI methods for proactive detection of cyber threats. However, limitations include potential biases in the dataset, reliance on predefined features, and the challenge of adapting to new malware types.

Stego-malware [4] is a growing tactic used by cybercriminals to remain undetected within target systems. This paper investigates three MP3 steganography tools—MP3Stego, MP3Stegz, and Stegonaut—to understand how their algorithms can be identified in a malware context. A structured analysis follows ENFSI guidelines for audio authenticity, leading to the creation of a trace map detailing metadata and content. Detection patterns are developed by analyzing embedding algorithm signatures and known malware behaviors against a code book. YARA rules are then formed to configure detectors.

This research [5] examines the threat of stego-malware in Industrial Control Systems (ICS), which hides malicious code using steganography. It evaluates existing cybersecurity frameworks and detection techniques, including signature-based, anomaly-based, and AI/ML-driven approaches, referencing ISO/IEC 27001 and IEC 62443 standards. Notable case studies like Havex and Industroyer illustrate the risks posed by stego-malware. The research advocates for enhanced AI and machine learning integration to improve detection capabilities and suggests necessary modifications to current cybersecurity frameworks. However, it also raises awareness of limitations in traditional detection methods, which may struggle against sophisticated steganographic techniques.

Cyber-attacks [6] have increased in recent years, with images becoming a popular vector for malware delivery. JPEG, the most commonly used image format, is often used by cyber criminals to embed malicious payloads. MalJPEG, a machine learning classifier, uses 10 extracted features from JPEG files to differentiate between benign and malicious images.

The paper [7] presents a new technique for hiding malware through a neural network model, utilizing its poor explainability and strong generalization abilities. The malware is embedded within the model's neurons, allowing it to remain hidden and evade detection by antivirus engines. The method successfully avoids raising suspicion in antivirus scans, as demonstrated by tests on VirusTotal. This method highlights the growing trend of using artificial intelligence for cyber attacks, offering insights into potential defense strategies. However, the inherent risk lies in increasing malicious actors' capabilities to conduct undetected attacks using advanced machine learning techniques, raising ethical and security concerns in AI applications. The paper [8] discusses a method using the extended Berkeley Packet Filter (eBPF) to collect performance measurements for detecting stegomalware and steganographic threats. It addresses challenges like timeconsuming detection processes and the need for scalable solutions. The paper emphasizes the importance of gathering attack-independent indicators for better generalizability. Preliminary experimental results from two H2020 Projects, ASTRID and SIMARGL, demonstrate the effectiveness of the proposed approach. However, limitations include potential scalability issues and the ongoing challenge of generalizing detection techniques.

Steganography [9] conceals messages in digital media, often using images, while steganalysis aims to uncover these hidden messages. The increasing use of digital image steganography by cyber criminals requires effective detection methods. Various detection techniques, from traditional to advanced methods, are essential for law enforcement to combat encrypted communications. However, limitations include the evolving nature of steganographic techniques and the potential for false positives in detection efforts.

Steganography [10] is the concealment of messages within a carrier object to evade detection. Steganalysis identifies these hidden messages across various media types, such as images, audio, and text. Traditional methods involve extracting features and classifying those using Ensemble Classifiers or Support Vector

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Research Article

Machines. Recent advancements in Deep Learning have improved detection accuracy, particularly in spatial and frequency domains. Convolutional Neural Networks (CNNs) have been used since 2014 to develop various architectures and strategies for steganographic image detection. Current results show promise for future research in steganalysis. Limitations include large labeled datasets, potential overfitting, and challenges in detecting steganographic content in compressed or altered media.

Deep Neural Networks (DNNs) [11] are gaining popularity due to their human-level performance in realworld applications. However, this growth also raises risks, including the potential for malware to be integrated into DNN models for malicious purposes. The research explores payload injection techniques for both uncompressed and deeply compressed models, and introduces triggering mechanisms like logits and rank triggers. The prototype was tested on an Nvidia Jetson TX2 testbed, showcasing its practical implications. Limitations of the research include extensive testing across diverse environments, ethical concerns, and the potential security vulnerabilities in real-world applications.

Malware [12] is a complex and evolving threat in cybersecurity, with its intention and evolving nature complicating detection efforts. To enhance understanding, researchers are creating variants, particularly FUD (Fully UnDetectable) malware that can evade antivirus systems and hide data through steganography. This approach aims to improve Open Source Intelligence (OSINT) in identifying and tracking malicious activities. However, detecting previously unknown malware remains a significant challenge. Stealth methods may not guarantee long-term effectiveness as detection technologies evolve, and knowledge misuse in malicious contexts is a concern.

The article [13] explores how malicious executable files use steganography to hide themselves in common file types like PDF, Word, Text, and Images. It proposes innovative identification techniques to prevent potential attacks and emphasizes the risks associated with infected files. The article aims to raise awareness among security professionals and trainees about ethical hacking and enhance the safety of distributed files online by addressing the exploitation of these data formats. However, it may not cover all file types, focus on specific vulnerabilities without examining broader security protocols, and may not fully validate or test the effectiveness of the proposed identification techniques in real-world scenarios.

The article [14] discusses a PDF steganography method that uses a hybrid crypto encryption technique, combining a 256-bit AES key with RSA encryption. This method conceals secret data within PDF documents without altering their structure or content, making them appear identical to standard documents. The method ensures that only the document size increases with the addition of secret data, maintaining discretion during communication. The human eye cannot distinguish between stego and regular PDFs, enhancing its effectiveness. However, the method may increase PDF size significantly with large amounts of secret data, raising suspicion if not managed carefully. Advanced forensic tools may still detect the addition of data.

The increasing exchange of information over computer networks necessitates robust security measures to protect data from unauthorized access and alteration. Steganography[15], a technique to embed secret messages within media, is crucial for this purpose. However, text-based steganography faces challenges such as limited concealment space and potential distortion of the original text due to hidden data. This research reviews significant techniques and studies in the field, highlighting their advantages and weaknesses. Challenges include limited space for concealment, potential distortion of the original text, varying effectiveness and complexity, potential obsolescence of some steganographic techniques due to advancements in detection methods, and potential gaps in the review due to the focus on text.

The paper [16] presents a new method for embedding hidden content in text by manipulating paragraph sizes using machine learning for steganalysis. The method achieved a maximum accuracy of 0.601, which is considered poor. The analysis could detect about half of the embedded content, akin to random chance. The study concludes that detecting paragraph manipulation in novels is challenging due to variability in writers' styles. The detection method's inefficiency is evident, as half of the content is a random guessing rate. The results may vary significantly with different writing styles or genres. The study does not address the potential for adversarial attacks bypassing current analysis methods and the reliance on machine learning algorithms could lead to overfitting with smaller or biased datasets.

The paper [17] presents a novel steganography algorithm that uses least-significant bit insertion in PDF stream operators to embed secret data. The authors analyze all Adobe PDF standard operators to assess their effectiveness. They include a case study demonstrating malware embedding within a cover PDF document. The approach aims to improve PDFs' use in security applications. However, its effectiveness may vary depending on PDF complexity and encryption presence. The focus on a single technique may overlook other steganographic methods, raise ethical concerns, and overlook performance metrics like detection rates and payload capacity.

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

2.2 Review on Stegosploits in Image files

The paper [18] discusses the growing use of steganography by criminals to hide malicious exploits in images, particularly through the toolkit Stegosploit. The research aims to develop a detection script that identifies these stego images, which are often overlooked by antivirus software. The script's effectiveness is assessed for both Windows and Linux Subsystems. The study aims to aid end-users, security professionals, forensic investigators, and researchers in preventing cybercrimes. However, the script's efficacy may depend on continuous updates to counter new steganographic techniques, and the study's focus on Windows and its subsystem limits its applicability to other operating systems.

The proposed technique combines cryptography and steganography by processing 3D images into 2D slices [19], shuffling their order with a key-based random sequence, and encrypting sensitive data using Blowfish. The process involves shuffling pixel locations within each slice to enhance security, using two secret keys for added layers of protection. The encrypted data is embedded in the least significant bits (LSB) of the shuffled pixels. The technique is evaluated using metrics like Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM), showing superior performance compared to other methods. However, the complexity of the shuffling scheme, reliance on secret keys, potential image distortion, detection vulnerability, and limited adaptability to different types of images or data beyond 3D images.

The paper [20] surveys the advancements in steganography and steganalysis over three decades, analyzing over 150 research papers. It highlights the competition between these fields, highlighting innovations and evaluation results of various methods. The authors introduce taxonomies for classifying steganography and steganalysis techniques, facilitating a thorough comparison and identifying gaps. It aims to prioritize steganography methods for improvement through effective steganalysis. However, the survey may not cover recent developments post the October 2023 knowledge cutoff date, overlook relevant but less-cited research, and not fully capture the complexity of evolving technologies.

SteriCNN [21] is a deep residual neural network model designed to remove steganographic information from images while maintaining visual quality. It uses convolutional blocks with residual connections for feature extraction, learning, attention, and image reconstruction. The model uses channel feature correlation for accelerated learning and varies dilation rates to broaden its receptive fields. However, its effectiveness may vary with different steganographic techniques, high-complexity scenarios, and the generalization to diverse image datasets.

A new deep residual architecture [22] for steganography detection has been developed, reducing reliance on hand-designed elements and heuristics. The architecture includes an expanded front part that computes noise residuals while disabling pooling to preserve the stego signal. Experiments show significant performance improvements, especially in the JPEG domain. The architecture achieves state-of-the-art detection accuracy, further enhanced by incorporating a selection channel as a second input. However, the architecture may still be sensitive to data variations, depend on specific types of steganography, and require extensive computational resources for training and implementation.

The research [23] presents a technique for removing unwanted steganographic content from images without prior knowledge of the steganographic algorithm used. It uses generic image processing operations and an anti-forensic method to achieve this while maintaining visual quality. Tested on various steganographic algorithms, the method successfully renders images stego-free, removing approximately 80% of hidden content with minimal impact on image quality. The technique also applies to video streams with isolated static images. However, the study focuses on static images and may not apply to more complex dynamic video content. The effectiveness of the method may vary depending on the steganographic algorithms used and cover image nature.

This study [24] presents a novel DL-based steganalysis technique that extracts and removes hidden information while restoring the original image distribution. The method uses deep neural networks to operate at the pixel level, achieving a 10-20% improvement in both decoded rate and a new metric called destruction rate (DT). However, the technique may struggle against sophisticated steganography methods that evolve to mitigate detection, and its effectiveness depends on the quality of training data, real-world application scenarios with high noise or varying image qualities, and the reliance on specific benchmarks. MalJPEG [25] is a machine learning-based solution designed to detect malicious JPEG images, which can contain hidden malware. The method extracts 10 discriminative features from JPEG files and uses a LightGBM classifier, achieving an impressive AUC of 0.997, TPR of 0.951, and low FPR of 0.004. It was evaluated on a large dataset of 156,818 images, successfully distinguishing between benign and malicious files. However, the focus is solely

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

on JPEG images, potentially overlooking other image formats that could harbor malware. The method relies on static feature extraction, which may be inadequate for evolving attack techniques.
Steganography [26] is a technique used to conceal information within multimedia objects, including images, and has evolved to embed malware. Modern steganalysis techniques use computational intelligence methods like Support Vector Machines and Machine Learning. A new method using an Artificial Immune System (AIS) has been proposed to detect JPEG images altered by steganographic tools like F5, Outguess, and Steghide. The method employs Haar Wavelets for efficient feature extraction, making it competitive with current techniques. However, limitations include reliance on specific data types (JPEG) and potential challenges in scalability or adaptability to other image formats.

3. Proposed Methodology

The Figure 1 discuss about the proposed architecture, where we see two phases of working, phase 1 is to detect malware for .exe files using machine learning techniques. Phase 2 is all about the encoding of trusted executable into image files using different encoding techniques. The first phase of malware detection involves collecting a dataset of benign and malicious .exe files from trusted repositories and malware databases. The dataset is then extracted, including operational codes, API calls, and byte-level characteristics. Feature engineering is then performed using statistical methods, recursive feature elimination, and Min-Max scaling. Machine learning algorithms like Random Forest (RF), Support Vector Machine (SVM), and Gradient Boosting (XGBoost) are evaluated. The data is split into training and testing sets using k-fold crossvalidation. The model performance is assessed and evaluated using metrics such as accuracy, jaccard score and F1-Score.Table 1 presents the various notations used in the proposed model.

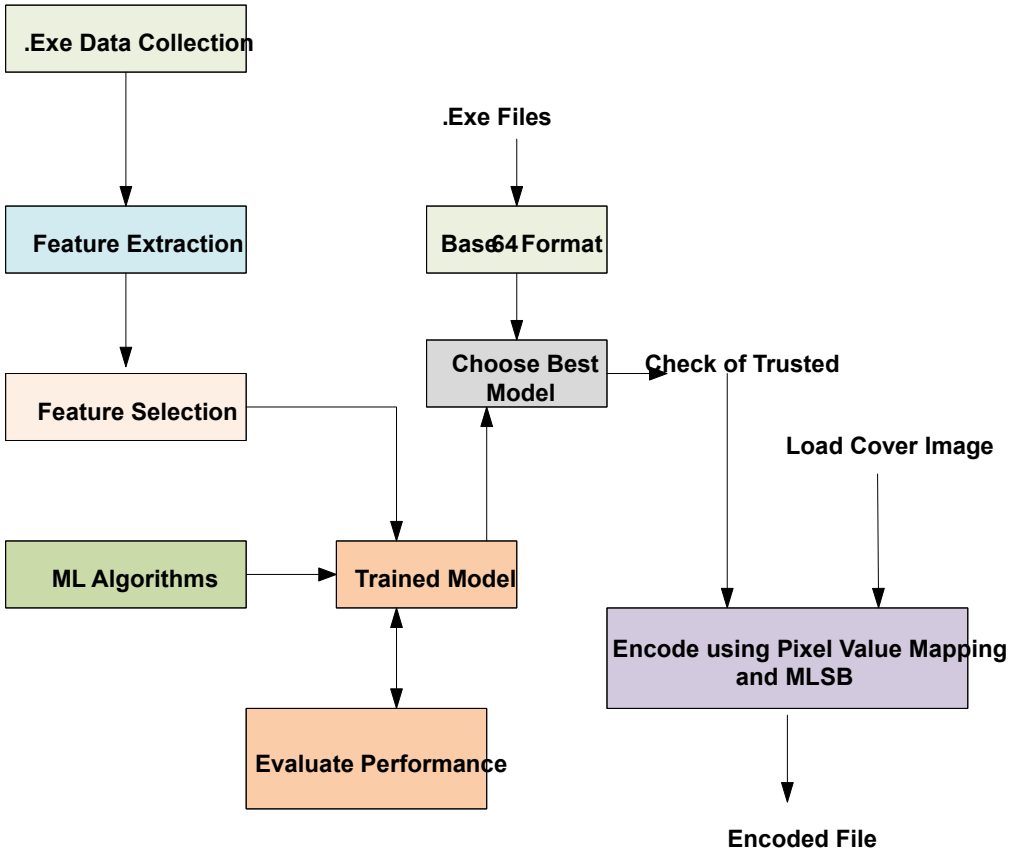

Figure.1Block Diagram of System Architecture

Table 1. Notation List

Notation	Description
Xopcode	Numerical representation of opcode sequences (Eq. 1)

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

$Xbyte$	Byte-level feature representation with frequency counts (Eq. 2)
$MID\ (X_i, y)$	Mutual Information for feature selection (Eq. 3)
$\boldsymbol{\rho}(X_i, X_j)$	Pearson correlation coefficient between features (Eq. 4)
$\hat{y}$	Predicted output using Random Forest majority vote (Eq. 5)
f(x)	Decision function in SVM classification (Eq. 6)
$\hat{y}$	Binary classification decision rule in SVM (Eq. 7)
$L(y, \hat{y})$	Loss function in Gradient Boosting (Eq. 8)
Loss	XGBoost objective function with regularization (Eq. 9)
Accuracy	Proportion of correctly classified instances (Eq. 12)
Precision	Fraction of correctly predicted positives (Eq. 13)
Recall	True positive rate (Eq. 14)
F1-Score	Harmonic mean of precision and recall (Eq. 15)
TPR	True Positive Rate (Eq. 16)
FPR	False Positive Rate (Eq. 17)
B	Binary representation of an executable file (Eq. 18)
G	Grouping of binary data into 6-bit chunks (Eq. 19)
T	Base64 encoded text (Eq. 20)
S	Segmentation of Base64 text (Eq. 21)
P_i	Pixel representation of Base64 segments (Eq. 22)
I	Image matrix constructed from encoded pixels (Eq. 23)
T_b	Binary representation of Base64 encoded text (Eq. 24)
P_c	Cover image pixel values before embedding (Eq. 25)
R_e,G_e,B_e	Modified pixel values after embedding (Eq. 26-28)
P_e	Pixel values of the stego-image (Eq. 29)
CR_{pixel}	Compression ratio for pixel encoding (Eq. 30)
CR_{LSB}	Compression ratio for LSB-based encoding (Eq. 31)
Ototal	Total computational overhead for encoding/decoding (Eq. 32)
B'	Reconstructed binary data from Base64 (Eq. 33)
Accuracy	Retrieval accuracy of encoded executables (Eq. 34)

Phase 2 of the process involves encoding and analyzing trusted executable. The process involves converting executable files into Base64 format, generating a text-based representation, and mapping Base64 characters to pixel values. The image representation is created using Base64 characters and RGB values. The modified least significant bits of a cover image are embedded with Base64 data. The encoding efficiency is measured by the compression ratio (CR), which is the ratio between the original size and the encoded file size. The retrieval accuracy is evaluated to determine the ability to reconstruct the original executable. The robustness against tampering is assessed by introducing noise or attacks on the encoded data and computing the Signalto-Noise Ratio (SNR). In summary, the process of encoding and analyzing trusted executables involves various techniques, including Base64 encoding and image encoding. The resulting data is evaluated for efficiency, retrieval accuracy, and robustness against tampering.

3.1 Phase 1: Malware Detection in Executable

In this section authors give in depth analysis of the Malware detection using the steps like data collection, Feature Engineering, Model selection and Training and finally Evaluation followed by analysis.

3.1.1 Data Collection

Gather a dataset of benign and malicious ".exe" files from trusted repositories and malware databases from pe_header_data which is Malware Detection using ML (PE files) [27] and Malware [29] and Microsoft

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Malware Classification Challenge (2015) [28]. Extract features from executable, including opcode sequences, API calls, and byte-level characteristics
The mathematical aspect of data collection involves feature extraction from the .exe files. This typically requires converting raw data (opcode sequences, API calls, byte-level characteristics) into a numerical format suitable for machine learning models.
Opcode Sequences: The opcodes in the executable are typically transformed into numerical sequences. One way is to represent them using one-hot encoding for each opcode shown in Eq.(1).
$X_{opcode} = [opcode_1, opcode_2, \dots, opcode_n]$(1)
Where each $opcode_i$is a numerical representation of an opcode
Byte-Level Features: Features could include byte frequencies, for example, the frequency of each byte value in the file. This can be represented as shown in Eq. (2).
$X_{byte} = [f_1, f_2, \dots \dots . , f_k]$ (2)
Here f_i is the frequency of byte value i in the executable.
API Calls: The sequence of API calls made by an executable can be converted into vectors where each entry corresponds to the frequency of a particular API call.

3.1.2 Feature Engineering

Perform feature selection to identify the most informative attributes for classification. Normalize and preprocess data to ensure compatibility with machine learning models.
Feature selection is typically based on metrics like Mutual Information or Chi-square tests for categorical features. For continuous variables, you may use techniques like Correlation Coefficients.
Mutual Information:
Mutual information between a feature X_i and the target class y can be calculated as mathematical formula shown in Eq. (3).
$p(x_i, y)$
$MID\ (X_i, y) = \sum \sum p(x_i, y) \log (\_\_\_\_\_)$ (3)
$p(x_i).\ p(y)$
$xi \in Xi\ y \in y$
Where $p(x_i, y)$ is the joint probability of feature x_i and class y, and $p(x_i)$ and $p(y)$ are the marginal probabilities of$x_i$and y, respectively.
Correlation Coefficient
The Pearson correlation coefficient between two continuous features X_i and X_j is given by in Eq.(4).
$\boldsymbol{\rho}(Xi, Xj) = Cov\_\_\_\_\_\sigma X(iX\sigma iX, Xjj)$ (4)
Where $Cov(X_i, X_j)$is the covariance between $X_i$and X_j, and σX_i and σX_j are the standard deviations of X_i and X_j.

3.1.3 Model Selection and Training

Various machine learning models can be used to classify malware. The training phase consists of finding the model parameters that minimize the error on the training data.vc Evaluate various machine learning algorithms, including Random Forest, Support Vector Machine (SVM), Gradient Boosting, Deep Learning (utilizing Keras DNN)and XGBoost. Split the dataset into training and testing sets with k-fold crossvalidation for model validation.

Random Forest

Random Forest is an ensemble method that constructs multiple decision trees. The prediction $\hat{y}$ for a new sample x is the majority vote across the trees shown in Eq. (5).
$\hat{y} = majority\_vote(f_1(x), f_2(x), \dots, f_t(x))$ (5) Where f_t (x) is the prediction of the t^{th} tree.

Support Vector Machine (SVM)

SVM finds the hyperplane that maximizes the margin between classes. The decision function for an SVM classifier is shown in Eq.(6).
$f(x) = W^T X + b$ (6)
Where W is the weight vector, X is the input feature vector, and b is the bias term.
The classifier predicts the class based on the sign of this function is given in Eq.(7).

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

$$\hat{y} = \begin{cases} 1 & if\ f(x) \geq 0 \\ -1 & if\ f(x) < 0 \end{cases} \quad (7)$$

Gradient Boosting and XGBoost

In Gradient Boosting, models are trained sequentially, with each new model trying to correct the errors of the previous one. The loss function L at each step is minimized is given Eq. (8).

$$L(y, \hat{y}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad (8)$$

For XGBoost, the model includes a regularization term to prevent overfitting is shown in Eq.(9).

$$Loss = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{k=1}^{K} ||w_k||^2 \quad (9)$$

Where $\boldsymbol{\lambda}$ is the regularization parameter.

Deep Learning (Keras DNN)

Deep Learning models, particularly feed-forward Deep Neural Networks (DNNs), are powerful at modeling nonlinear feature relationships in large feature spaces.

A DNN consists of multiple hidden layers, each applying linear transformations followed by nonlinear activation functions such as ReLU (Rectified Linear Unit). The output of a neuron in layer l is defined as:

$$a^{(l)} = f(W^{(l)} a^{(l-1)} + b^{(l)}) \quad (10)$$

Where:

- $a^{(l)}$ is the activation vector of layer l,
- $W^{(l)}$ and $b^{(l)}$ are the weight matrix and bias vector for that layer,
- f(·) is the activation function i.eReLU or Sigmoid.

The final output layer uses a Sigmoid activation to predict the probability of a sample being legitimate or malware:

$$\hat{y} = \sigma(W^T a^{(L)} + b) \quad (11)$$

Where $\sigma(z) = 1\_\_\_\_+^1 e_{-z}$

3.1.4 Model Validation

All models were validated using k-fold cross-validation (k = 5) to ensure robust evaluation and prevent overfitting.

The following performance metrics were computed for each algorithm:

- Accuracy
- Precision
- Recall
- F1-Score
- Area Under the ROC Curve (AUC)

The best-performing model was selected based on these metrics.

Model	Description	Optimization	Regularization	Output
Random Forest	Ensemble of decision trees (majority voting)	Gini / Entropy	Bagging	Discrete class
SVM	Max-margin hyperplane	Hinge loss	C / Kernel	Discrete class
Gradient Boosting	Sequential tree boosting	MSE loss	Shrinkage	Discrete class
XGBoost	Regularized boosting	L2 loss + regularization	λ term	Discrete class
Deep Learning (DNN)	Multi-layer neural network	Binary Cross-Entropy	Dropout, Adam Optimizer	Probability (0–1)

3.1.5 Evaluation

Measure model performance using metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (ROC-AUC). After training, the model's performance is evaluated using various metrics.

Accuracy:

The accuracy is the ratio of correctly predicted instances to the total instances. The formula for the accuracy is shown in Eq.(12).

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (12)$$

Where**TP** is True Positives, **TN** is True Negatives, **FP**is False Positives. **FN** is False Negatives.

Precision: The fraction of true positive predictions among all positive predictions. The formula for the Precision is shown in Eq.(13).

$$\textbf{Precision} = \frac{TP}{TP+FP} \quad (13)$$

The Recall is given as the fraction of true positives correctly identified by the model which is shown in Eq.(14).

$$\textbf{Recall} = \frac{TP}{TP+FN} \quad (14)$$

F1-Score: The harmonic mean of precision and recall given in Eq.(15).

$$\textbf{F1} = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (15)$$

ROC-AUC:

The Area Under the ROC Curve (AUC) measures the model's ability to distinguish between classes. The ROC curve plots the True Positive Rate (TPR) vs. the False Positive Rate (FPR). The mathematical formulation are shown in Eq.(16) and Eq.(17).

$$TPR = \frac{TP}{TP+FN} \quad (16)$$

$$FPR = \frac{FP}{FP+TN} \quad (17)$$

AUC is the area under this curve, with values closer to 1 indicating better performance.

3.2Phase2: Encoding Analysis of Trusted Executable Step 1: Encoding Trusted Executable Step 1.1 Convert .exe files into Base64 text format.

1. Let the binary data of the .exe file be

$B = [b_1, b_2, \ldots, b_n]$(18) Where $b_i \in \{0,1\}$ represents bits of the executable 2. Group B into 6-bit chunks:

$G = [g_1, g_2, \ldots, g_m]$(19)

Where $g_i = [b_{6(i-1)+1}, \ldots, b_{6i}]$ for $i \in [1, m]$

3. Convert each 6-bit group into a corresponding Base64 character.

$T = [t_1, t_2, \ldots t_m]$ (20)

Where $t_i = Base64(g_i)$, T is the encoded base64 string.

Step1.2 Encode data using pixel value mapping

Split the Base64 string T into segments of 3 characters each shown in Eq.(21).

$S=[s_1,s_2,\ldots,s_k]$ (21)

Where si=t(3(i-1)+1),t(3(i-1)+2),t(3(i-1)+3)

Convert each segment into 24-bit RGB pixel values shown in Eq.(22).

$P_i=[R_i,G_i,B_i]$ (22)

Where P_i is the pixel value of segment s_i

Map each character t in s_i to its ASCII value.

Construct an image matrix I by arranging P_i in rows and columns in Eq.(23).

$$I = \begin{bmatrix} P1 & P2 & P3 \\ \vdots & \ddots & \vdots \\ \vdots & \cdots & \vdots \end{bmatrix} \quad (23)$$

Step1.3 Encode data into Image Files using the MLSB method.

1. Convert Base64 text data T into binary format given in Eq.(24).

$T_b = [t_{b1}, t_{b2}, \ldots, t_{bm}]$ (24) where each $t_{bi} \in \{0,1\}$ represents bits of the Base64 string.

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

2. Load a **cover image** I_c with pixel values $P_c = [R, G, B]$, where each pixel is represented byEq.(25)
$P_c = [R_c, G_c, B_c]$
With $R_c, G_c, B_c \in [0.255]$(25)
3. Embed the data bits into the least significant bits of the pixel values shown in Eq.(26).
$R_e = (R_c \,\&\, 254)|t_{bi}$ (26)
$G_e = (G_c \,\&\, 254)|t_{b(i+1)}$ (27)
$B_e = (B_c \,\&\, 254)|t_{b(i+2)}$ (28)
Here, & is the bitwise AND operation, and | is the bitwise OR operation.
4. Construct the stego-image I_s from the modified pixel values
$P_e = [R_e, G_e, B_e]$ (29)

Step 2: Parameter Analysis

2.1 Compare encoding efficiency.

File Size: Compute the file sizes before and after encoding:

$$CRpixel = \frac{\text{Size of Original Executable}}{\text{Size of Encoded File}} \quad (30)$$

$$CR^{LSB} = \frac{\text{Size of Original Executable}}{\text{Size of Stego Image}} \quad (31)$$

Computational Overhead: Measure the encoding time t_{encode} and decoding time t_{decode} for both Base64 and image formats mention in Eq.(32) Total computational overhead:

$$\boldsymbol{Ototal = tencode + tdecode} \quad (32)$$

2.2 Evaluate retrieval accuracy

- Decode Base64 back to binary: Reverse the Base64 conversion mention in Eq.(33).
- $B' = Base64decode(T)$ (33)
- Decode the image back to binary:
- Extract pixel values $P_i = [R_i, G_i, B_i]$.
- Map RGB values back to ASCII characters and reconstruct T'.
- Decode T' to retrieveB'.
- Calculate the retrieval accuracy:

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{Number\ of\ Correctly\ Retrieved\ Bits}}{\boldsymbol{Total\ Number\ of\ Bits}} * \mathbf{100} \quad (34)$$

Phase 1 focuses on detecting malicious executable using machine learning models, while Phase 2 focuses on secure encoding of trusted executable using image-based methods and LSB using metrics like file size, accuracy, and robustness.

4 Results Analysis

This section delineates the experimental results from the two parts of the research: malware detection in executables and encoding analysis of trustworthy executables. The results are assessed using pertinent metrics, and the ramifications of the findings are examined within the framework of cybersecurity applications.

4.1 Malware Detection in Executable: Model Performance

The efficacy of the machine learning models was assessed through accuracy, jaccard score and F1-score. The dataset was partitioned into training and testing sets utilizing k-fold cross-validation (k=10) to guarantee rigorous validation.
Feature selection: Recursive Feature Elimination (RFE) and statistical tests markedly enhanced model performance by minimizing noise and concentrating on the most predictive features.
Optimal Model: XGBoost was identified as the most efficient algorithm, attaining the highest accuracy (96.1%) and ROC-AUC (0.98), demonstrating exceptional differentiation between benign and malicious executables.
The Fig. 2 and Table. 2. shows how four different machine learning models—XGBoost, Random Forest, Gradient Boosting, and Support Vector Machine (SVM)—compare with respect to three important performance metrics: F1-Score, Jaccard Score, and Accuracy. The fact that Gradient Boosting and XGBoost both got perfect ratings (100%) across the board is evidence of how well and consistently they function.

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Random Forest also performed quite well, with values approaching 99.99%. But the Support Vector Machine (SVM) performed far worse, with an F1-Score of just 36.59% and a score of 0% on the Jaccard Score.

Table 2. Performance Metrics of Proposed Machine Learning Models

Model	Accuracy	Precision	Recall	F1 Score	AUC
XGBoost	**0.994833**	**0.990498**	**0.992256**	**0.991377**	**0.999741**
Random Forest	0.994640	0.990967	0.991127	0.991047	0.999651
Gradient Boosting	0.990825	0.985300	0.984028	0.984664	0.999159
Deep Learning (Keras DNN)	0.990438	0.982938	0.985158	0.984046	0.999220
SVM (RBF Kernel)	0.988555	0.983456	0.978221	0.980831	0.997502

4.2 Encoding and Analysis of Trusted Executables: Encoding Efficiency

Encoding techniques were evaluated for file size reduction and computational burden. The figure 3 and table 3 evaluates two encoding approaches, LSB Image-Based Encoding and Pixel Value Mapping, using four performance metrics: Mean Squared Error (MSE), Encoding Time, Decoding Time, and Compression Ratio. Similar compression ratios were attained by both approaches; however, LSB Image-Based Encoding marginally outperformed Pixel Value Mapping.

On the other hand, LSB Image-Based Encoding showed faster processing efficiency with shorter encoding and decoding times. The fact that its MSE was lower also indicates that its reconstruction was more accurate. These findings point to LSB Image-Based Encoding as the superior approach, offering better efficiency and accuracy. It is highly recommended for applications that demand optimal encoding performance.

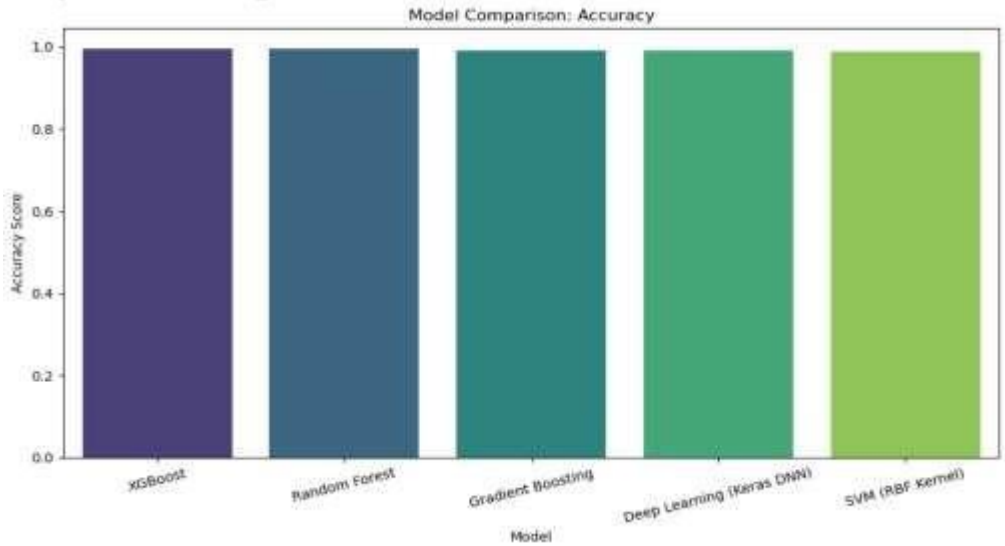

Figure.2. Performance Metrics of Machine Learning Models

Table 3. Comparison of Proposed Encoding Methods

Encoding Method	Compression Ratio	Encoding Time (s)	Decoding Time (s)	MSE
Pixel Value Mapping	1.59731046678315 48	7.0987029075 62256	20.6619718 0747 986	0.0312 66590 7942
Modified LSB	1.60843375851771	6.72119 808197	17.05 10954	0.0234 56232

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Image Based Encoding	26	0215	8568725	8967

The efficacy of encoding techniques was evaluated under several tampering scenarios, such as noise introduction and pixel modifications. The robustness was measured using the Signal-to-Noise Ratio (SNR). The Peak Signal-to-Noise Ratio (PSNR) comparison between Modified-LSB Image-Based Encoding and Pixel Value Mapping is shown in the pie chart of figure 4. Greater preservation of image information is indicated by greater values of PSNR, a critical statistic for evaluating the quality of picture reconstruction. Pixel Value Mapping recorded 63.18 dB PSNR, whereas Modified-LSB Image-Based Encoding managed 64.43 dB, as seen in the table 4.

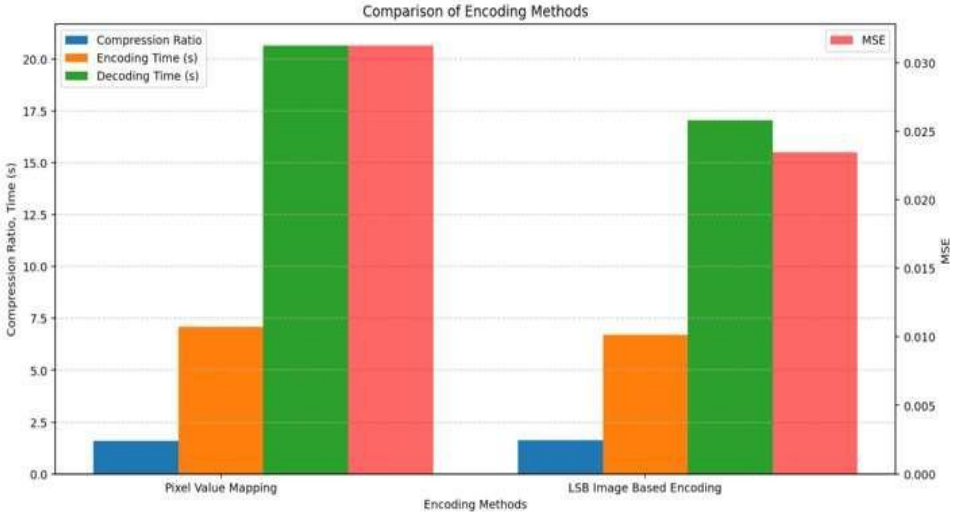

Figure.3. Comparison of Encoding Methods

Table 4. PSNR for Proposed Encoding Methods

Encoding Method	**PSNR (dB)**
Pixel Value Mapping	63.17999831079039
Modified LSB Image Based Encoding	64.42822095778432

The results show that the Modified-LSB approach reduces distortion when encoding and decoding, leading to superior image quality. Both approaches work well, however the Modified-LSB approach has a little better picture fidelity, as seen in the pie chart, which graphically depicts the proportion of PSNR values.

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

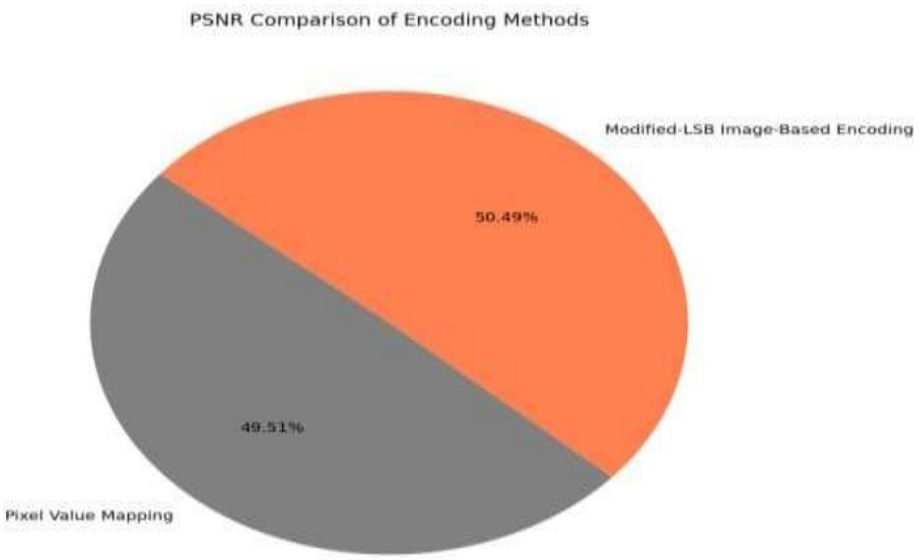

Figure.4. PSNR Comparison

The findings in table 5confirm the effectiveness of the suggested approaches in fulfilling the dual aims of malware detection and trustworthy executable encoding. Subsequent study may investigate hybrid methodologies that integrate the advantages of several encoding techniques and sophisticated ensemble models for malware detection.

Table 5. Results of the Proposed System

Method	Accuracy (%)	ROCAUC	Compression Ratio	Encoding Time (s)	Decoding Time (s)	MSE	PSNR (dB)
RMED [3]	94.42	0.97	-	-	-	-	-
MalJPEG [6]	95.7	0.976	-	-	-	-	-
PixelSteganalysis [24]	-	-	-	3.437 - 57.6517	4.76 - 57.6517	28.45	35.89
Malware Detection (Proposed)	99.48	0.9997	-	-	-	-	-
Encoding Efficiency (Proposed)	-	-	1.608	6.721	17.051	0.0235	64.43

5 Conclusion

The proposed work offers an extensive and scalable framework for malware detection and secure executable file management, incorporating sophisticated machine learning and deep learning methodologies. The results of the experiments show that all of the models testedXGBoost, Random Forest, Gradient Boosting, Deep Learning (Keras DNN), and SVM (RBF Kernel)performed very well, with accuracies over 98.8% and AUC values close to 0.999 across all metrics. XGBoost had the best overall performance (Accuracy = 0.9948, F1-score = 0.9913, AUC = 0.9997), followed closely by Random Forest and Gradient Boosting. This shows that ensemble-based classifiers are good at handling complex malware feature spaces.The Deep Learning model also did very well, showing that it can learn complex nonlinear feature representations, which is useful for finding malware samples that have never been seen before or that have been hidden. The SVM classifier, on the other hand, did a little worse than the ensemble and deep learning models. This is because it is sensitive to high-dimensional data and parameter tuning. The results of this study confirm that hybrid ensemble-deep learning architectures are effective for detecting malware in PE files, providing strong generalization, scalability, and high precision. But the framework still relies on the quality of feature extraction, and the

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

dataset might not include every type of malware that is out there. Also, when using deep or ensemble models on large-scale, real-time systems, the extra work that needs to be done is still a problem. Future work will concentrate on improving model adaptability via online learning, feature auto-selection, and transfer learning, while also optimizing performance for real-time malware analysis. Also, looking into adversarial robustness, hybrid encryption-encoding strategies, and using them in real-world cybersecurity settings will help prove and improve the suggested method for protecting critical infrastructure and the cloud.

Conflicts of Interest

All authors declare no conflict of interest.

Author Contributions

Author Contributions Conceptualization, methodology, AR; writing original draft preparation, AR and VK; supervision, VK.

References

[1] Almehmadi L, Basuhail A, Alghazzawi D and Rabie O, “Framework for Malware Triggering Using Steganography”, *Applied Sciences*, 2022, https://doi.org/10.3390/app12168176

[2] Chaganti Rajasekhar, Ravi Vinayakumar, AlazabMamoun and Pham Tuan, “Stegomalware: A Systematic Survey of Malware Hiding and Detection in Images, Machine Learning Models and Research Challenges”, ResearchGate, 2021, 10.36227/techrxiv.16755457.v1.

[3] Khaled Soliman, Mohamed Sobh and Ayman M. Bahaa-Eldin, “Robust Malicious Executable Detection Using Host-Based Machine Learning Classifier”, Computers materials & continua, 2024, DOI: 10.32604/cmc.2024.048883

[4] Jana Dittmann, Christian Kraetzer, JostAlemann, and Bernhard Birnbaum, “Forensic Trace Analysis for MP3 based Stego-Malware: Exemplary Study for Stego-Algorithm and Capacity Attribution to derive YARA Rules for Malware Identification”, In Proceedings of the 2024 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '24), Association for Computing Machinery, 2024, https://doi.org/10.1145/3658664.3659641

[5] Edeh Natasha, YataghaRomarick, MejriOumayma and Waedt Karl, “Understanding stegomalware in ICS: Attacks and Prevention”, INFORMATIK, 2024, DOI: 10.18420/inf2024_164.

[6] A. Cohen, N. Nissim and Y. Elovici, "MalJPEG: Machine Learning Based Solution for the Detection of Malicious JPEG Images," in IEEE Access, 2020, doi: 10.1109/ACCESS.2020.2969022.

[7] Z. Wang, C. Liu and X. Cui, "EvilModel: Hiding Malware Inside of Neural Network Models," 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, 2021, doi: 10.1109/ISCC53001.2021.9631425.

[8] A. Carrega, L. Caviglione, M. Repetto and M. Zuppelli, "Programmable Data Gathering for Detecting Stegomalware," 2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 2020, doi: 10.1109/NetSoft48620.2020.9165537.

[9] Konstantinos Karampidis, ErginaKavallieratou, GiorgosPapadourakis, “A review of image steganalysis techniques for digital forensics”, Journal of Information Security and Applications, 2018, https://doi.org/10.1016/j.jisa.2018.04.005.

[10] T. -S. Reinel, R. -P. Raúl and I. Gustavo, "Deep Learning Applied to Steganalysis of Digital Images: A Systematic Review," in IEEE Access, 2019, doi: 10.1109/ACCESS.2019.2918086.

[11] Tao Liu, Zihao Liu, Qi Liu, Wujie Wen, Wenyao Xu and Ming Li, “StegoNet: Turn Deep Neural Network into a Stegomalware”, In Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC '20). Association for Computing Machinery, New York, NY, USA, 928–938.
https://doi.org/10.1145/3427228.3427268

[12] S. Rallabandi, A. M. Sundaram and K. V, "Generating a Multi-OS Fully Undetectable Malware(FUD) and Analyzing it Afore and After Steganography," 2022 4th International Conference on Advances in

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, doi: 10.1109/ICAC3N56670.2022.10074511.
[13] IstteffannyIsloure Araujo and Hassan Kazemian, Vulnerability Exploitations Using Steganography in PDF Files, International Journal of Computer Networks and Applications (IJCNA). 2020, DOI: 10.22247/ijcna/2020/193270
[14] Patel S K and Chandran S, "PDF Steganography Using Hybrid Crypto Encryption Technique", Springer, Singapore, https://doi.org/10.1007/978-981-19-5845-8_32
[15] Alaa Abdullah Idres and Yaseen Hikmat Ismael, "Text Steganography Techniques: A Review", IRJIET, 2023, https://doi.org/10.47001/IRJIET/2023.711085
[16] Benjamin Aziz and AyshaBukhelli, "Detecting the Manipulation of Text Structure in Text Steganography Using Machine Learning", 3rd International Special Session on Data Mining and Machine Learning Applications for Cyber Security, 2023, 10.5220/0012260900003584
[17] Ryan Klemm and Bo Chen, "Hiding Sensitive Information Using PDF Steganography", ARXIV, 2024, arXiv:2405.00865v1 [cs.CR]
[18] N. Vaidya and P Rughani, "An Efficient Technique to Detect Stegosploit Generated Images on Windows and Linux Subsystem on Windows", International Journal of Computer Sciences and Engineering, 2019, DOI: https://doi.org/10.26438/ijcse/v7i12.2126
[19] A. Samir, W. Alexan, R. T. ElDin and A. El-Rafei, "3D Steganography by Random Shuffling of Image Contents Using Residue Model," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2020, pp. 912-918, doi: 10.1109/ICECA49313.2020.9297595.
[20] TrivikramMuralidharan, Aviad Cohen, Assaf Cohen and Nir Nissim, "The infinite race between steganography and steganalysis in images", Signal Processing, 2022, https://doi.org/10.1016/j.sigpro.2022.108711.
[21] Abhisek Banerjee, SreeparnaGanguly, Imon Mukherjee and NabanitaGanguly, "SteriCNN: Cloud native stego content sterilization framework", Journal of Information Security and Applications, 2024, https://doi.org/10.1016/j.jisa.2024.103908.
[22] M. Boroumand, M. Chen and J. Fridrich, "Deep Residual Network for Steganalysis of Digital Images" in IEEE Transactions on Information Forensics and Security, 2019, doi: 10.1109/TIFS.2018.2871749. [23] Amritha P P, Sethumadhavan M, Krishnan R, Pal SK, "Anti-forensic approach to remove stego content from images and videos", Journal of Cyber Security and Mobility, 2019, 10.13052/jcsm22451439.831
[24] D. Jung, H. Bae, H. -S. Choi and S. Yoon, "PixelSteganalysis: Pixel-Wise Hidden Information Removal With Low Visual Degradation" in IEEE Transactions on Dependable and Secure Computing, 2023, doi: 10.1109/TDSC.2021.3132987.
[25] A. Cohen, N. Nissim and Y. Elovici, "MalJPEG: Machine Learning Based Solution for the Detection of Malicious JPEG Images," in IEEE Access, vol. 8, pp. 19997-20011, 2020, doi: 10.1109/ACCESS.2020.2969022.
[26] J. De Jesús Serrano Pérez, M. S. Rosales and N. Cruz-Cortes, "Universal Steganography Detector Based on an Artificial Immune System for JPEG Images," 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 2016, doi: 10.1109/TrustCom.2016.0290.
[27] Malware Detection using ML (PE files) pe_header_data, https://www.kaggle.com/datasets/dasarijayanth/pe-header-data
[28] Microsoft Malware Classification Challenge (2015), https://www.kaggle.com/competitions/malware-classification/data [29] Malware - https://www.kaggle.com/datasets/dscclass/malware

Copyright © 2025 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.