Received: 20 December 2024 | Revised: 10 March 2025 | Accepted: 3 April 2025 | Published online: 12 May 2025

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2025, Vol. 4(4) 591–602
DOI: 10.47852/bonviewJCCE52025053

BON VIEW PUBLISHING

Utilizing Gray Wolf Optimization Algorithm in Malware Forensic Investigation

Mosleh Mohammd Abualhaj[1,*], Sumaya Al-Khatib[2], Nida Al Shafi[3], Iyas Qaddara[2] and Abdallah Hyassat[3]

[1]*Department of Networks and Cybersecurity, Al-Ahliyya Amman University, Jordan*

[2]*Department of Computer Science, Al-Ahliyya Amman University, Jordan*

[3]*Department of Data Science and Artificial Intelligence, Al-Ahliyya Amman University, Jordan*

Abstract: Malware forensic investigation plays a critical role in cybersecurity, aiming to unveil malicious activities, decipher their tactics, and bolster defense mechanisms. This article introduces an innovative approach to malware forensic investigation, harnessing the capabilities of the Gray Wolf Optimization (GWO) algorithm in conjunction with a range of machine learning classifiers. These classifiers include naive Bayes, random forests (RF), decision trees, support vector machines, and K-nearest neighbors. The study leverages the CIC-MalMem-2022 dataset, which comprises memory-based data, and employs the Python programming language for model development. Research findings highlight the superiority of the RF classifier, achieving an impressive 75.6% accuracy in a multiclass classification scenario involving 16 classes. Notably, our proposed approach consistently exhibits higher accuracy when compared to existing models applied to different datasets, reaching 99.2% in binary classification. Furthermore, on the same dataset, our model outperforms the competition by achieving 86.34% and 75.64% accuracy in multiclass classification scenarios involving four classes and 16 classes, respectively. These results underscore the promising potential of our proposed model in the domain of malware forensic investigation, particularly when analyzing data extracted from memory. By combining the strength of the GWO algorithm with RF, this study aids in the progression of robust and accurate malware forensic investigation methods, thereby enhancing cybersecurity efforts in an ever-evolving threat landscape.

Keywords: malware, Gray Wolf Optimization, machine learning, feature selection, random forest

1. Introduction

Cybersecurity protects internet-connected systems from attack, damage, and illegal access. It entails using various technologies, processes, and practices to protect networks, devices, and sensitive data against cyber threats like hacking, malware, phishing, and ransomware [1–3]. As we become more dependent on digital technologies and the internet, this becomes more critical. The number of cyberattacks is increasing rapidly; for instance, there were an estimated 500 million cyberattacks in 2021. In addition, cyberattacks cost organizations huge amounts of money; for instance, there were an estimated $6 trillion spent on cyberattacks in the same year [4].

Accordingly, organizations have to implement defense techniques to prevent cyberattacks. Different approaches are used for the identification and prevention of cyberattacks, including the retrospective approach. The retrospective approach consists of methods that investigate the factors that lead to attacks, as well as containing, repairing, eradicating, and recovering from the harm that an attack has caused to the target network. Cyberattack investigation (digital forensics) involves identifying the cause, preventing future attacks, and prosecuting the perpetrators [5–7]. The success of digital forensics relies on the quality and timeliness of the data collected, as well as the skills and experience of the investigators. There are several digital forensics techniques, including imaging, data recovery, file system analysis, etc. Each of these techniques uses several traditional tools (software and hardware) to support the various stages of the digital forensics process [8, 9].

Apart from digital forensics, machine learning (ML) is a subfield of artificial intelligence concentrating on developing techniques and systems that can automatically analyze data, detect patterns, and predict future behaviors [10, 11]. In digital forensics, techniques from ML can be utilized to automate a variety of operations, hence accelerating the investigation process. For example, ML techniques can be used to automate triage and categorization and recover data that has been lost or deleted from a digital device. The integration of ML and digital forensics will significantly enhance the capabilities of digital forensics. By leveraging ML techniques, digital forensics investigations can be more efficient, accurate, and cost-effective [12, 13]. However, it's vital to understand that the adoption of ML in digital forensics is not widely used, and numerous obstacles require attention, such as the risk of false positives and the need to ensure the acceptance and effectiveness of evidence in court cases [12, 13].

One critical area where ML can enhance digital forensics investigations is by reducing false positives through effective dimensionality reduction, specifically via feature selection techniques. Feature selection chooses a subset of key features (inputs) for model creation. Before developing an ML model, feature

***Corresponding author:** Mosleh Mohammd Abualhaj, Department of Networks and Cybersecurity, Al-Ahliyya Amman University, Jordan. Email: m.abualhaj@ammanu.edu.jo

selection can increase performance, prevent overfitting, and lower training costs [14, 15]. There are several methods for performing feature selection in ML. Ultimately, choosing a feature selection method varies based on the nature of the problem, the type of data, and the learning algorithm being used. One of the most recently used methods in feature selection is optimization methods. Meta-heuristic optimization algorithms could be utilized to determine the optimal set of features by treating the feature selection process as an optimization challenge. The optimization algorithm searches for the subset of features that leads to the highest performance of an ML model, such as the highest accuracy or lowest error [16, 17]. The Gray Wolf Optimization (GWO) algorithm is one of the vigorous meta-heuristic optimization algorithms [17, 18]. In this paper, a Digital Forensics Investigation (DFI) model will be proposed to improve the performance of malware digital forensics applications. The proposed DFI model uses the GWO algorithm to select the key features that help reduce the dimensions of malware data. Thus, it reduces the false positives and improves the digital forensics process. In addition, the DFI model will evaluate several ML classifiers, and the best one will be used with the DFI model.

This article is structured as follows: Section 2 highlights some of the related works regarding digital forensic applications. Section 3 discusses the suggested digital forensics investigation model. Section 4 assesses the efficiency of the suggested model. Section 5 provides the conclusion of the article.

2. Literature Review

Ali et al. [19] proposed an automatic authentication method that successfully distinguishes between genuine and forged audio. The system's design philosophy is built upon three psychoacoustic concepts of hearing, enabling it to categorize sounds recorded with the same microphone in multiple environments. By leveraging calculated features based on psychoacoustic principles, the system performs audio authentication and environment classification using a Gaussian mixture model for automatic decision-making. To test its performance, the system was evaluated with audio samples indistinguishable from human hearing, and three human judges subjectively assessed the quality of the forged audio. The results were impressive, with the suggested system achieving a remarkable categorization accuracy of 99.2% $\pm$ 2.6. Moreover, it attained a flawless 100% accuracy rate in various scenarios, including text-dependent and text-independent voice authentication. Overall, the proposed automatic authentication system exhibits extraordinary accuracy in classifying diverse recording situations and effectively differentiating between authentic and fabricated audio.

Ahmed et al. [20] propose a deep learning-based method to determine the acquisition date of digital photos, which proves valuable in temporal forensics analysis. The approach involves establishing the digital image acquisition time to create a chronological sequence for unknown pictures by comparing them with a collection of pictures from the same source with known timestamps. Convolutional neural networks (CNNs), specifically the AlexNet and GoogLeNet architectures, are employed for feature extraction and transfer learning. The researchers conducted experiments using the extensive NTIF database, specifically designed for temporal picture forensics. They utilized two existing CNN multiclass models to estimate the timeline of digital photos. The AlexNet CNN with transfer learning demonstrated impressive performance, accurately classifying over 85% of test images into the appropriate period classes. Overall, across all CNN models, the estimation accuracy of the digital image acquisition time ranged from about 80% to 88%.

Hina et al. [21] developed a content-based multi-label email classification technique to distinguish between spam and non-spam emails. This method was specifically tailored for forensic investigations involving large-scale email data. To achieve this goal, they compared various ML techniques. They found that logistic regression (LR) outperformed naive Bayes (NB), Stochastic Gradient Descent (SGD), random forest (RF), and support vector machine (SVM) in terms of accuracy. When utilizing bi-gram features, LR exhibited the highest performance, achieving an impressive accuracy of 91.9% in tests using benchmark datasets. On the other hand, SGD showed the lowest accuracy due to perfect parameter learning.

Kachavimath et al. [22] used two separate ML algorithms, K-nearest neighbors (KNN) and NB, to detect DDoS attacks in network forensics. They utilized the NSL-KDD and KDD Cup 99 datasets, which contain information about network traffic, as input for their model. The ML model processed the refined data to extract high-level features and identify patterns in network traffic sequences. To streamline the process, they retained eight highly correlated features essential for DDoS attack identification and removed the remaining features by evaluating the correlation between all 41 features. Experimental results demonstrated that both KNN and NB algorithms outperformed traditional learning models. The model's efficiency was assessed utilizing different measures derived from the confusion matrix. KNN exhibited superior performance over NB across almost all criteria, achieving an impressive accuracy of 98.51% compared to NB's 93.95%.

Al Banna et al. [23] have introduced a forensic investigation model for assessing image authenticity without relying on embedded security measures. The model utilizes a CNN and transfers learning approaches to extract features from an image dataset, resulting in reduced training time by retaining MobileNet feature extractor weights from ImageNet. An open image dataset of 3900 images was captured using three camera models. SVM, LR, and RF classifiers were utilized to measure the usefulness of the model. The SVM classifier achieved the highest accuracy at 98.82%, closely followed by the LR classifier at 98.54%, while the RF classifier attained a marginally lower accuracy of 97.16%. This research demonstrates a promising approach to image authenticity verification in forensic investigations, with the suggested model offering valuable contributions to the field of image forensics.

Shafin et al. [24] propose an efficient multiclass malware detection method that identifies recent malware and is well-suited for execution on embedded devices. This innovative approach integrates the feature-learning capabilities of CNNs with the temporal modeling advantages of Bidirectional Long Short-Term Memory (BiLSTM) networks, resulting in two distinct models: Compact CNN-BiLSTM (CompactCBL) and Robust CNN-BiLSTM (RobustCBL). These models stand out for their compact size and fast processing speed, making them particularly appropriate for implementation on resource-constrained embedded devices. Several tests performed on the recent CIC-Malmem-2022 dataset showcase the greater achievement of the proposed method when compared to other ML-based models proposed in the literature. In binary attack scenarios, both the RobustCBL and CompactCBL models achieved impressive accuracy rates of 99.96% and 99.92%, respectively. When dealing with malware family attacks, the RobustCBL and CompactCBL models demonstrated accuracy rates of 84.56% and 84.22%, respectively. Furthermore, in individual malware attacks within the family attack category, the RobustCBL and CompactCBL models achieved accuracy rates of 72.6% and 71.42%, respectively.

Carrier et al. [25] propose an effective framework to address the problem of advanced malware that utilizes obfuscation and other evasion techniques to avoid known detection approaches. Within this novel framework, they are extending VolMemLyzer, which is among the most cutting-edge solutions with regard to learning systems' memory feature extraction. This extension focuses on the

detection of obfuscated and concealed malware. It uses a stacked ensemble ML framework to come up with a good way to find malware. The researchers use the MalMemAnalysis-2022 malware memory dataset to mimic hidden malware settings in order to make the proposed framework work better. The results show that this suggested method of memory feature engineering works very well at finding malware that has been hidden or obfuscated in a short amount of time (99.02% accuracy).

Mezina and Burget [26] present a new approach for tackling the problem of obfuscated malware with the use of memory information. They used the CIC-MalMem-2022 dataset in their study to get a better idea of how well a dilated CNN (DCNN) finds hidden malware. As will be shown, DCNNs have become effective tools for this task due to their capacity to perform a correlation analysis of wide data fields, which is essential for the detection of concealed malware. It enlarges the dilation space and contains all-encompassing capabilities in the feature range with the same size of parameters. The research provides a thorough evaluation of the proposed scheme, including binary classification and multiclass classification. Notably, the presented results are highly effective, with 99.92% and 83.53% accuracy rates for binary and multiclass tasks, respectively. The proposed model architecture comprises four layers, with each of the two layers in each layer having 32 to 256 neurons. However, it is important to mention that the size of the model creates a problem with implementation in resource-limited devices due to the high computational overhead.

The studies above have introduced various forensic investigation models tailored to different attack types and datasets. Table 1 presents a concise overview of the accuracy attained by these models, focusing on binary classification, as reported in their respective studies. Similar to studies in References [24–26], the current study focuses on malware attacks, utilizing data extracted from memory dumps. Table 2 summarizes the accuracy achieved for the primary malware categories, involving a 4-class multiclass classification, within the CIC-MalMem-2022 dataset. Additionally, Table 3 summarizes the accuracy achieved for subcategories of malware, encompassing a 16-class multiclass classification within the CIC-MalMem-2022 dataset.

Table 1
Binary classification with different datasets

Ref	Model	Accuracy
Proposed	DFI model (RF)	99.93%
[19]	Gaussian mixture	99.20%
[20]	AlexNet CNN	85.00%
[21]	LR	91.90%
[22]	KNN	98.51%
[23]	SVM	98.82%
[25]	Stacking	99.02%

3. Digital Forensics Investigation (DFI) Model

This section discusses the suggested model used in the DFI. First, the CIC-MalMem-2022 dataset used to evaluate the proposed DFI model will be discussed. Then, the steps used to prepare the CIC-MalMem-2022 dataset for the DFI model will be described. After that, the GWO optimizer will be discussed in light of digital forensics. The ML classifiers utilized in the proposed DFI model will be discussed next. Finally, the entire DFI model processes will be shown.

Table 2
Multiclass classification (4 classes) with MalMem dataset

Ref	Model	Accuracy
	DFI model (RF)	86.34%
[24]	RobustCBL	84.56%
[24]	CompactCBL	84.22%
[25]	Stacking	N/A
[26]	DCNN	83.53%

Table 3
Multiclass classification (16 classes) with MalMem dataset

Ref	Model	Accuracy
Proposed	DFI model (RF)	75.64%
[24]	RobustCBL	72.60%
[24]	CompactCBL	71.42%
[25]	Stacking	N/A
[26]	DCNN	N/A

3.1. CIC-MalMem-2022 dataset

The CIC-MalMem-2022 dataset will be used in this work. It is built to represent as close to a real-world situation as possible. The CIC-MalMem-2022 dataset uses debug mode for the memory dump process to avoid the dumping process showing up in the memory dumps. The dataset contains a total of 58,596 records, with 29,298 benign and 29,298 malicious, which makes it a balanced dataset. The malicious records are divided into three main categories and 15 subcategories, as presented in Table 4. The number of features in

Table 4
Obfuscated-MalMem-2022 malware categories

Malware Main Categories	Malware Subcategories-# of Samples	# of Records
Ransomware	MAZE-1958. Pysa-1717. Shade-2128. Conti-1988. Ako-2000.	9791
Spyware	TIBS-1410. Gator-2200. 180Solutions-2000. Transponder-2410. Coolwebsearch-2000.	10020
Trojan Horse	Refroso-2000. Reconyc-1570. Zeus-1950. Scar-2000. Emotet-1967.	9487
Total		29,298

Table 5
Obfuscated-MalMem-2022 features

#	Feature Name	#	Feature Name	#	Feature Name
1	pslist.nproc	20	handles.nmutant	39	psxview.not_in_eprocess_pool_false_avg
2	pslist.nppid	21	ldrmodules.not_in_load	40	psxview.not_in_ethread_pool_false_avg
3	pslist.avg_threads	22	ldrmodules.not_in_init	41	psxview.not_in_pspcid_list_false_avg
4	pslist.nprocs64bit	23	ldrmodules.not_in_mem	42	psxview.not_in_csrss_handles_false_avg
5	pslist.avg_handlers	24	ldrmodules.not_in_load_avg	43	psxview.not_in_session_false_avg
6	dlllist.ndlls	25	ldrmodules.not_in_init_avg	44	psxview.not_in_deskthrd_false_avg
7	dlllist.avg_dlls_per_proc	26	ldrmodules.not_in_mem_avg	45	modules.nmodules
8	handles.nhandles	27	malfind.ninjections	46	svcscan.nservices
9	handles.avg_handles_per_proc	28	malfind.commitCharge	47	svcscan.kernel_drivers
10	handles.nport	29	malfind.protection	48	svcscan.fs_drivers
11	handles.nfile	30	malfind.uniqueInjections	49	svcscan.process_services
12	handles.nevent	31	psxview.not_in_pslist	50	svcscan.shared_process_services
13	handles.ndesktop	32	psxview.not_in_eprocess_pool	51	svcscan.interactive_process_services
14	handles.nkey	33	psxview.not_in_ethread_pool	52	svcscan.nactive
15	handles.nthread	34	psxview.not_in_pspcid_list	53	callbacks.ncallbacks
16	handles.ndirectory	35	psxview.not_in_csrss_handles	54	callbacks.nanonymous
17	handles.nsemaphore	36	psxview.not_in_session	55	callbacks.ngeneric
18	handles.ntimer	37	psxview.not_in_deskthrd		
19	handles.nsection	38	psxview.not_in_pslist_false_avg		

the dataset is 55, excluding the category (output column) [24, 27]. Table 5 lists these features.

3.2. Preprocessing

When discussing ML, the term "preprocessing" refers to the processes that are carried out in order to get the input data ready for utilization in an ML model. The primary goal of preprocessing is to convert raw data into a format that aligns with the requirements of training an ML model. This may require performing a wide range of activities on data, including cleaning, transformation, normalization, reduction, and splitting. Data cleaning is performed when the dataset contains missing, inconsistent, or irrelevant data [10, 28, 29]. Accordingly, the CIC-MalMem-2022 dataset does require data cleaning. Data transformation is performed to convert the data into an appropriate format, such as converting textual data into numerical data. In the CIC-MalMem-2022 dataset, only the category (output column) feature is represented as textual data [10, 29]. Therefore, the label-encoding technique is used to convert it to a suitable format (i.e., numerical), as shown in Table 6. Data normalization is performed to rescale the large numerical values to values between 0 and 1 [10, 29]. Several features need to be normalized in the CIC-MalMem-2022 dataset. Thus, the Min-Max scaling method is used to rescale these features [10, 29]. Table 7 shows a sample of the data before and after normalization. Data reduction is performed to remove irrelevant features that could negatively impact an ML model. Feature selection is one of the approaches utilized for data reduction. Feature selection is a key pillar that impacts the performance of an ML model. Choosing the right feature selection algorithm for the intended application (i.e., forensic application) is crucial [16–18]. In this paper, the GWO optimizer will be utilized and evaluated to select suitable features for malware forensic applications. The GWO optimizer will be discussed in the next section in detail.

Table 6
Label encoding

# of Types	Before Encoding	After Encoding
2 Types	Malware	0
	Benign	1
4 Types	Trojan Horse	0
	Spyware	1
	Ransomware	2
	Benign	3
16 Types	Emotet	0
	Shade	1
	Ako	2
	Zeus	3
	Gator	4
	Pysa	5
	Transponder	6
	Conti	7
	Refroso	8
	TIBS	9
	Coolwebsearch	10
	180Solutions	11
	Reconyc	12
	MAZE	13
	scar	14
	Benign	15

Table 7
MalMem dataset normalization process

Before normalization	After normalization
42, 16, 10.73809524, 0, 209.2142857, 1621, 38.5952381, 8787, 209.2142857	0.091743119, 0.125, 0.602767848, 0, 0.232847424, 0.307725139, 0.682017433, 0.168570919, 0.076448356
40, 16, 9.525, 0, 204.175, 1504, 37.6, 8167, 204.175	0.082568807, 0.125, 0.522309316, 0, 0.226113578, 0.257789159, 0.660305073, 0.140899759, 0.068341683
42, 16, 10.02380952, 0, 206.2619048, 1610, 38.33333333, 8663, 206.2619048	0.091743119, 0.125, 0.555392853, 0, 0.228902246, 0.303030303, 0.676303654, 0.163036687, 0.071698876
44, 17, 9.590909091, 0, 200.7954545, 1674, 38.04545455, 8835, 200.7954545	0.526680736, 0, 0.221597593, 0.330345711, 0.670023219, 0.170713202, 0.062905026

3.3. GWO optimizer for digital forensic applications

There are several purposes for DFI, including identifying the source of the cyberattack, investigating allegations of fraud, and recovering lost data. A key pillar for accurate DFI is the collected data from the scene. The more relevant the collected data, the more accurate the DFI. For this, the GWO optimizer will be used, which is efficient in selecting the most relevant features among a large set of features from the collected data, thus improving the accuracy of the DFI model investigation [17, 18].

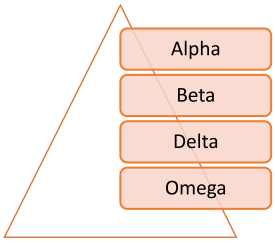
GWO is a meta-heuristic optimization algorithm that was influenced by how gray wolves hunt. Numerous optimization issues, such as feature selection in ML, have been solved using GWO. GWO can be used in feature selection to look for the ideal subset of features that gives an ML model (e.g., DFI model) the greatest performance. GWO can manage a high number of features and nonlinear correlations between features and the target variable, making it a flexible and efficient method for feature selection [17, 18]. GWO was chosen for its superior exploration-exploitation balance, computational efficiency, and strong convergence behavior compared to other algorithms, such as Particle Swarm Optimization (PSO) or Genetic Algorithm (GA). Unlike PSO, which may converge prematurely, and GA, which requires complex operations, GWO's hierarchical hunting mechanism ensures efficient feature selection. This helps identify the most discriminative memory-based features for malware detection while reducing redundancy [30, 31].

3.3.1. Gray wolf's key operations

The GWO algorithm optimizes a specified objective function by simulating the social structure and hunting behavior of the gray wolf pack. A pack of wolves is seen as a population of candidate solutions in GWO, with each wolf standing in for a prospective solution. Each iteration updates the positions of the wolves based on those of the other wolves. The alpha, beta, and delta wolves are the top three wolves in the pack, while the omega is considered the scapegoat in the pack. The second and third-best solutions are denoted by the beta and delta wolves, respectively, while the alpha wolf represents the best option thus far. Figure 1 shows the social hierarchy of gray wolves. The gray wolves follow certain behaviors to hunt for prey. The GWO has modeled this behavior mathematically for optimization problems [17, 18, 30, 31]. The procedures for this behavior and the issues that should be considered while finding the optimal solution are discussed below.

At first, the GWO algorithm involves exploring the solution space to find the optimal solution. This stage can be modeled by generating a population of wolves randomly within the decision space. Each wolf represents a candidate solution to the optimization issue. The initial population can be generated randomly or utilizing some other heuristic method. Then, the encircling behavior of gray wolves is used to simulate the balance between exploration and exploitation. This stage can be modeled by selecting the three best wolves in the population: alpha, beta, and delta wolves. The other wolves in the pack then update their positions toward the optimal solution. After that, the attacking behavior of gray wolves is used to simulate the exploitation of the optimal solution found so far. This stage can be modeled by updating the position of the alpha wolf toward the optimal solution found by the pack so far. Finally, the hunting behavior of gray wolves is used to simulate the overall search for the optimal solution. This stage involves iterating through the encircling and attacking stages for a fixed number of repetitions or until a stopping condition is encountered. Overall, the GWO algorithm uses a combination of exploration and exploitation, communication and coordination among the wolves in the pack, and simulations of the hunting behavior of gray wolves to effectively explore the solution space for the optimal solution [17, 18, 30, 31].

Figure 1
Social hierarchy



3.3.2. Selected features by GWO

As mentioned earlier, GWO has been successfully applied in various optimization problems, including feature selection. The GWO algorithm has been implemented to select the most relevant and influential feature for the DFI model. GWO has several hyperparameters that need to be set to appropriate values to obtain good performance in forensic applications. The hyperparameters and their recommended values for forensic applications are population size between 10 and 50, maximum number of iterations between 100 and 500, search range of [−1, 1], and crossover rate of 0.8. It is important to note that the optimal values of these hyperparameters may depend on the specific forensic application and the size and complexity of the dataset. Figure 2 displays the pseudocode of the GWO algorithm that has been used in the proposed DFI model [30–33]. Lines 2–4 initialize a population of gray wolves (solutions) with random positions. Lines 7–11 update the locations of the wolves. Line 12 finds the new position, which is a weighted sum of the calculated positions. Line 13 checks if the new position is within the

Figure 2
Pseudocode of GWO algorithm

```
WOA Algorithm
1.  psn is position, a is alpha wolf,
    b is beta wolf, d is delta wolf
2.  For each wolf i:
3.     Initialize position Xi
4.     Evaluate fitness f(Xi)

5.  While # of iteration is not met:
6.     For each wolf i:
7.        A = 2 * (rand() * C1) - 1
8.        C2 = 2 * rand()

9.        X1 = a.psn - A * (abs(C2 * a.psn - Xi.psn))
10.       X2 = b.psn - A * (abs(C2 * b.psn - Xi.psn))
11.       X3 = d.psn - A * (abs(C2 * d.psn - Xi.psn))

12.       Xi_new = (X1 + X2 + X3) / 3.0

13.       Clip Xi_new to the search space bounds

14.       Evaluate f(Xi_new)

15.       If f(Xi_new) < f(alpha_wolf):
16.          d = b
17.          b = a
18.          a = Xi_new
19.       Else if f(Xi_new) < f(b) and f(Xi_new) > f(a):
20.          d = d
21.          d = Xi_new
22.       Else if f(Xi_new) < f(d) and f(Xi_new) > f(b):
23.          d = Xi_new

24.       Xi.psn = Xi_new

25. Return the best solution found
```

search space bounds. Line 14 evaluates the fitness of the new position. Lines 15–23 update the alpha, beta, and delta wolves. Line 24 updates the position of wolf *i* to the new position. Line 25 returns the best solution found. Based on the implementation of the GWO and the values of the hyperparameters, the features have been reduced from 55 to 4 for binary classification, 8 for 4-class multiclass classification, and 16-class multiclass classification. The selected features are shown in Table 8. The GWO effectively selects the features that minimize noise while retaining critical discriminative properties. The selected features significantly influence the performance of our model by enhancing its ability to distinguish between different malware types. These features provide essential behavioral patterns that help classifiers achieve superior performance [30, 31].

Table 8
Selected features with different types of classification

Type of classification	Selected features
binary classification	2, 20, 23, and 46
4-class multiclass classification	13, 14, 16, 18, 20, 24, 27, and 54
16-class multiclass classification	6, 14, 18, 21, 28, and 29

3.4. Utilized classifiers

This section discusses the classifiers that are utilized in the DFI model. Classifiers are an integral part of every model that is used for ML. Every classification strategy has a unique set of advantages and disadvantages that are unique to it. The nature of the issue that needs to be handled is a major factor in determining which classifier is going to be the most effective for a certain endeavor. Because of this, it is essential to test multiple models and evaluate how well they perform before selecting the most effective one. In the DFI model, the decision tree (DT), SVMs, RF, NB, and KNN classifiers will be evaluated, and the best among them will be used for the DFI model.

3.4.1. DT classifier

DTs can address several classification and regression problems. DTs have root, internal, and leaf nodes. Internal nodes represent dataset features, while the root node represents the dataset. Each internal node has branches representing feature values. The model's output is the leaf nodes' choice. The DT technique iteratively partitions the dataset into subgroups based on the feature that best classifies the data or lowers target variable variation. The algorithm ends when the tree reaches its maximum depth, or the minimum number of samples needed to split a node is not available. The steps of building a DT are as follows: First, split the dataset based on the most informative feature and threshold. Second, create a decision node based on the chosen feature. Third, split the dataset into smaller subsets based on the selected feature. Repeat the first three steps for each subset. Then, form a leaf node when no further splitting is needed. Finally, use the resulting DT for prediction [10, 29].

3.4.2. RF classifier

Several DTs are integrated into the RF ensemble learning method to improve the classifier's performance and robustness. Both classification and regression issues can be solved with this well-liked ML approach. Each DT in RF is built utilizing a random subset of the training data along with a subset of the features. This lessens overfitting and improves tree decorrelation. The majority vote or the average of the different DTs' predictions serves as the foundation for the RF classifier's final prediction. The steps of building an RF are as follows: First, determine the number of trees in the forest. Second, a subset of the training data will be randomly selected to train each tree. Third, randomly select a subset of the features to reduce the correlation between trees. Fourth, build a DT based on the selected data and features. Repeat the first four to build the predefined number of trees. Finally, the resulting RF will be used for prediction [10, 29].

3.4.3. SVMs classifier

SVMs are a form of supervised ML algorithms used in regression and classification. The objective of SVMs is to determine a hyperplane that efficiently separates the different classes within the training data. The hyperplane is chosen to optimize the margin, which indicates the distance to the closest data points from each class. SVMs have numerous merits, including their strong performance in high-dimensional spaces, and they do not burden memory because only a subset of the training data is utilized in the decision function. However, training SVMs can be computationally costly, particularly when dealing with large datasets. The steps performed by the SVM classifier are as follows: First, select a kernel function that transforms the input data into a higher-dimensional space. Second, find the optimal hyperplane that maximizes the margin between different classes. Third, optimize the hyperparameters to balance

complexity and generalization. Finally, the resulting SVM model is used for predictions [10, 29].

3.4.4. NB classifier

The NB classifier is a probabilistic classification algorithm that makes use of Bayes' theorem to classify data into different classes. It works by assuming that the features in the data are independent of each other, given the class label. Because of its strong assumption of feature independence, which may not hold true in many real-world situations, the technique is known as "naive." Overall, the NB is a straightforward yet effective classifier that works well for high-dimensional datasets and has many features. However, its performance may be limited by the independence assumption, and it may not work well for datasets with correlated features. The steps performed by the NB classifier are as follows: First, separate the data by class. Second, compute the prior probabilities of each class. Third, calculate feature and class statistics. Fourth, find the posterior probability of a data point based on the observed features. Finally, use the resulting NB for predictions [10, 29].

3.4.5. KNN classifier

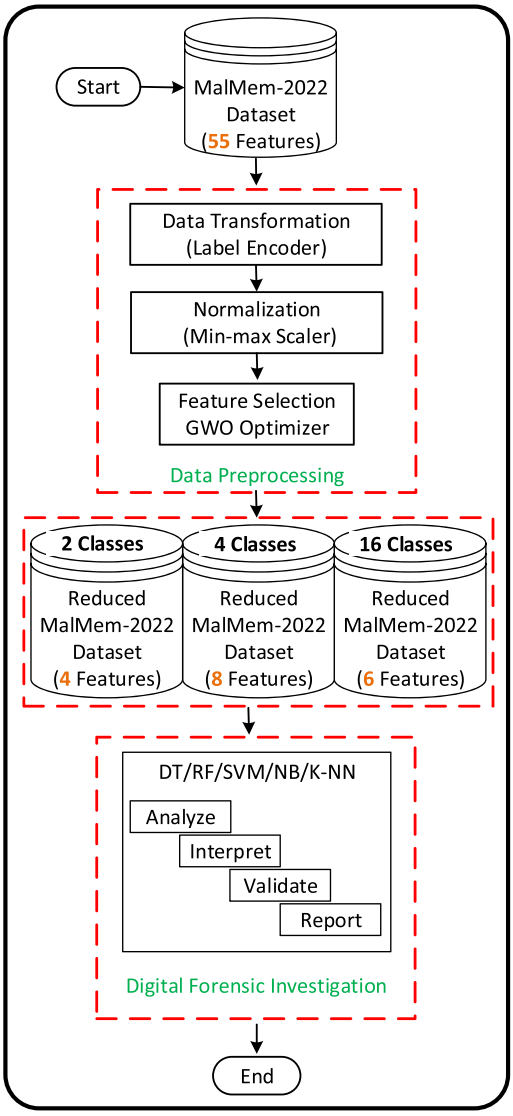
KNN is a supervised learning algorithm employed for classification and regression purposes. In the context of classification, KNN is a form of instance-based learning, where the model learns to categorize new instances based on how similar they are to instances that have already been seen in the training set. Each occurrence in the training set is denoted as a vector in a multidimensional feature space by the KNN algorithm. The method locates the k instances in the feature space closest to the new instance. The class of the new instance is decided by a majority vote among its k nearest neighbors after the new instance's k nearest neighbors have been found. KNN is a straightforward and understandable algorithm that can be applied to numerous classification jobs. But when working with big datasets or high-dimensional feature spaces, it can be computationally expensive. The steps performed by the KNN classifier are as follows: First, determine the number of nearest neighbors (K) that will be considered when making a classification decision. Second, select a distance metric to quantify how close data points are to one another. Third, the distance with all samples will be calculated. Fourth, select the K samples with the shortest distance. Fifth, assign the predicted class of the new data point. Finally, the resulting KNN model is used for predictions [10, 29, 34].

3.5. Investigation processes using DFI model

Digital forensic investigations involve the process of analyzing digital evidence to identify and recover information that can be used in legal proceedings. ML can be used to assist in this process by automating some of the repetitive tasks, reducing the workload on the forensic investigator, and potentially uncovering hidden patterns and insights in large volumes of data. There are several ways in which ML can be applied in digital forensic investigation, including malware analysis, log analysis, and file carving. For instance, ML algorithms can be trained on large datasets of malware samples to classify new samples as malicious or benign automatically. This can help forensic investigators quickly identify malware and understand its behavior. Though it is a general model, the effectiveness of the suggested DFI model uses the CIC-MalMem-2022 dataset (discussed in Section 3.1). In order to prepare it for the DFI model, the data within CIC-MalMem-2022 has been transformed using label-encoding and normalized using Min-Max scaling (discussed in Section 3.2). The final step before running the ML algorithms is to reduce the size of the data for better performance of the DFI model. For this, the GWO optimizer will be utilized, as discussed in Section 3.3.

The DFI model will use ML classifiers to examine the data and determine its relevance to the investigation. ML algorithms will classify and categorize the evidence, identify patterns, and flag anomalies that may indicate suspicious activity. After analysis, the evidence must be interpreted to determine its meaning and relevance to the investigation. ML classifiers can be used to identify relationships between different pieces of evidence and help investigators conclude. Then, the findings of the investigation will be validated to make sure they are reliable and accurate. ML algorithms can be used to cross-check the results obtained from different analysis techniques to ensure their consistency. Finally, the DFI model will report the findings of the investigation and provide recommendations for further action. Numerous ML classifiers will be tested in the DFI model, and the one that gives the best result will be put to use in the DFI model. Figure 3 demonstrates the flow of the DFI model.

Figure 3
The DFI model



4. Performance Evaluation

This section evaluates the proposed DFI model. First, the tool and environment used to implement the DFI model will be discussed. Then, the criteria used to evaluate the DFI model will be described. Finally, the obtained results will be pretested and analyzed.

4.1. Implementation tools and environment

The proposed DFI model was implemented using Python. Python is a popular language for ML because it has a rich set of libraries and frameworks that facilitate the implementation of ML algorithms and techniques. The DFI model implementation has been executed on a Core i7 PC. The specification of the PC is summarized in Table 9. The K-fold cross-validation method will be employed to evaluate the DFI model's performance. The advantage of K-fold cross-validation is that it ensures that all the available data can be used for both training and evaluation, which can result in a more precise assessment of the model's performance.

Table 9
PC specification

Item	Description
Processor	Intel Core i7-12700 with speed of 2.1GHz up to 4.9GHz and 25MB cache
Memory	32GB DDR5
Storage	1TB SSD M.2
Graphic	NVIDIA GeForce RTX 4060 8GB
Operating system	Windows 10 Professional

4.2. The evaluation criteria

The proposed DFI model has been assessed based on the elements of the error matrix: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). TP is the proportion of instances that were properly categorized as positive. FP is the proportion of instances that were wrongly categorized as positive. TN is the proportion of instances that were properly categorized as negative. FN is the proportion of instances that were wrongly categorized as negative. Using these four elements, several metrics are derived to evaluate the efficiency of the DFI model. These metrics are the accuracy (Equation 1), the precision (Equation 2), the recall (Equation 3), and the $F1$-score (Equation 4). As noticed from the equations, accuracy measures the percentage of correctly classified instances, precision tells how precise the positive predictions are, recall indicates how many of the actual positives the model successfully identified, while the $F1$-score is the harmonic mean of precision and recall [35–37].

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{1}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{2}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{3}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

4.3. Results analysis

This section first evaluates the proposed DFI model using DT, RF, SVM, NB, and KNN classifiers. Then, it compares the DFI model results against other forensic investigation models on different datasets and data. Finally, the DFI model results are compared against other models on the same dataset.

Figures 4–7 show the accuracy, precision, recall, and $F1$-score of the DFI model, respectively. The RF classifier has achieved the highest results among the five tested classifiers with all four metrics. Meanwhile, the RF algorithm has attained the same accuracy, precision, and recall of 75.6% and the highest $F1$-score with 75.4%. This result is achieved with the 16 types of classes (multiclass classification) in the CIC-MalMem-2022 dataset. It is important to mention that certain malware categories had lower detection rates due to imbalanced data distribution within the CIC-MalMem-2022 dataset.

Figure 4
Accuracy of the DFI model with 16 classes

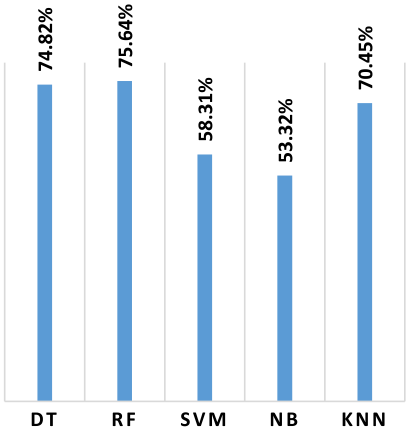


Figure 5
Precision of the DFI model with 16 classes

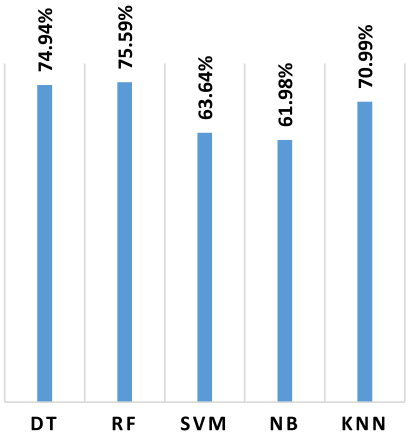


Figure 6
Recall of the DFI model with 16 classes

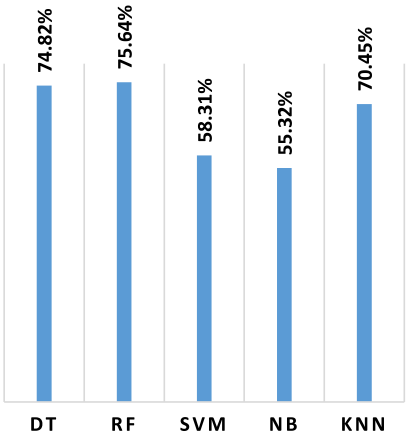
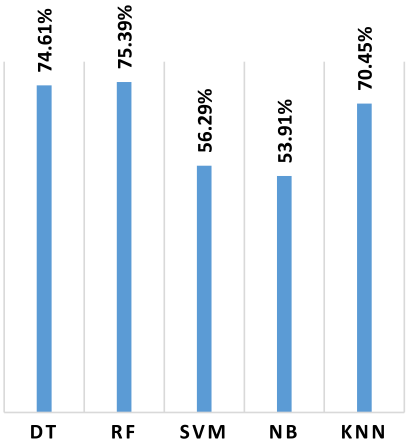


Figure 7
***F*1-score of the DFI model with 16 classes**



In addition, some classes had significantly fewer samples, which may have hindered the model's ability to learn their distinct patterns.

Figure 8 shows the accuracy of the DFI model with the RF classifier (DFI-RF) compared to other forensic investigation models across several datasets. The DFI-RF model demonstrated a maximum accuracy of 99.93%, surpassing the accuracy of the nearest model [19], which obtained 99.2%, resulting in an improvement of 0.71%. This outcome is attained using binary classification, relying on the outcomes presented by similar systems.

Figure 9 displays the accuracy of the DFI-RF compared to other forensic investigation models on the identical dataset (CIC-MalMem-2022). The DFI-RF model demonstrated the best accuracy rate of 86.34%, surpassing the nearest model, RobustCBL (R-CBL) [24], which attained an accuracy rate of 84.56%. This indicates an improvement of 1.87%. The four types of classes (multiclass classification) in the CIC-MalMem-2022 dataset are responsible for achieving this outcome.

Figure 9
The DFI-RF model accuracy versus other forensic models using CIC-MalMem-2022 dataset (four classes)

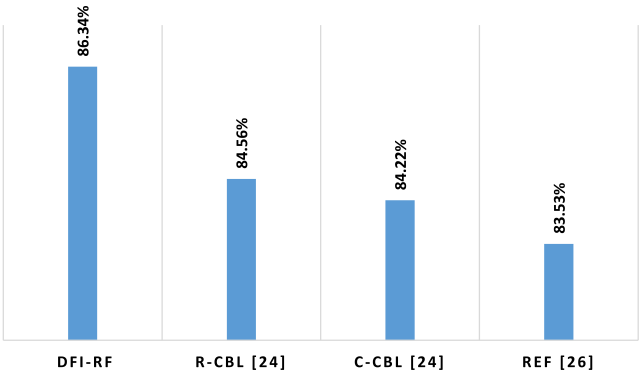


Figure 10 illustrates the comparative accuracy of the DFI-RF model in relation to various forensic investigation models using the identical dataset (CIC-MalMem-2022). The DFI-RF model demonstrated a maximum accuracy of 75.64%, surpassing the accuracy of the R-CBL model [24] by 3.06%, which reached an accuracy of 72.60%. This result is obtained by utilizing the 16 distinct types of classes (multiclass classification) present in the CIC-MalMem-2022 dataset.

Figure 8
The DFI-RF model accuracy versus other forensic models across several datasets (binary classification)

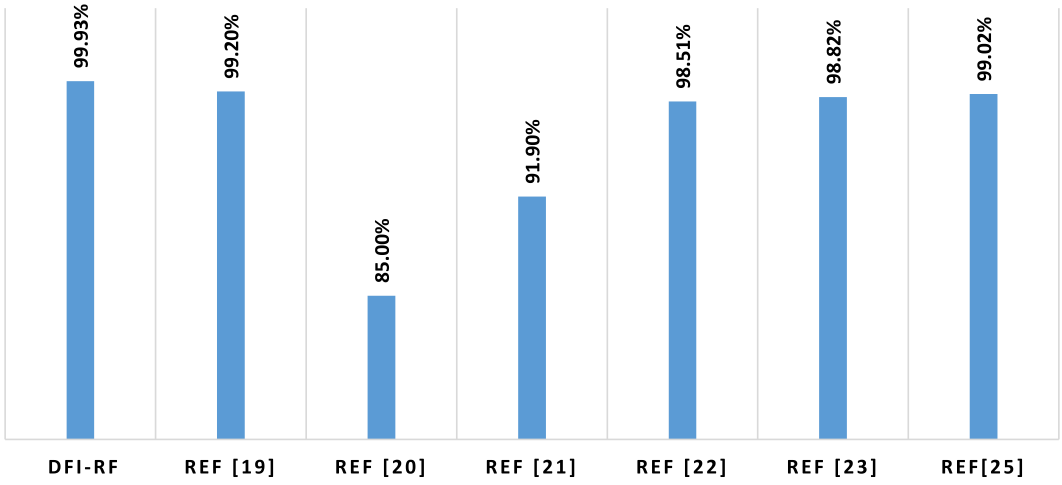
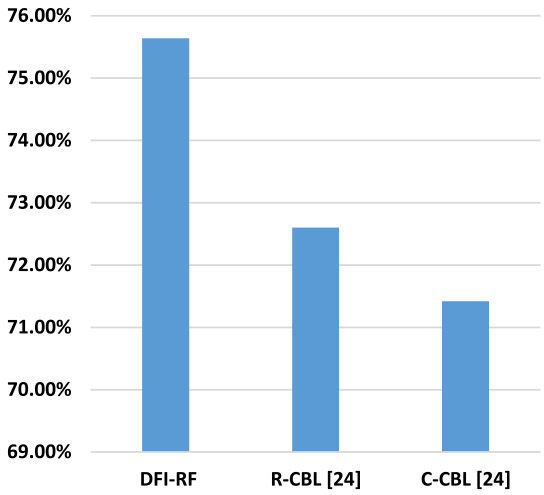


Figure 10
The DFI-RF model accuracy versus other forensic models using CIC-MalMem-2022 dataset (16 classes)



4.4. The DFI complexity theoretical analysis

The complexity of the GWO feature selection is O (T × N × d), where T is the number of iterations, N is the population size, and d is the feature dimension. Compared to other meta-heuristic algorithms, GWO offers a balanced trade-off between search efficiency and computational cost. In addition, the RF classifier, which achieved the best performance, has a worst-case complexity of O(m × n log n), where m is the number of trees and n is the dataset size. Despite these computational demands, the proposed method remains feasible due to GWO's efficient search strategy and RF's parallel processing capabilities, which optimize training time.

5. Conclusion

This article introduces an innovative malware forensic investigation (DFI) model that leverages the GWO algorithm and a comprehensive array of ML classifiers. Utilizing the memory-centric CIC-MalMem-2022 dataset and implementing Python for rigorous computational implementation, our research underscores the exceptional prowess of the RF classifier, achieving an impressive 75.6% accuracy in multiclass classification across 16 classes. Importantly, our model DFI-RF consistently outperforms competing approaches, achieving a remarkable 86.34% accuracy in multiclass classification involving four classes and maintaining its superiority on the same dataset with an accuracy of 75.64% in multiclass classification across 16 classes. These results underline the transformative perspective of our DFI model, particularly concerning memory-resident data and its capacity to advance the field of malware forensic investigation, fortifying cybersecurity endeavors in an ever-evolving threat landscape. In future work, we plan to explore hybrid optimization techniques to enhance feature selection and classification accuracy further. In addition, we plan to include a more thorough evaluation by incorporating comparisons with baseline algorithms such as PSO, GA, and traditional feature selection methods. We will also evaluate our approach on diverse datasets to ensure its robustness across various malware types and forensic scenarios.

Acknowledgment

The authors would like to acknowledge Al-Ahliyya Amman University for its support and resources provided throughout the course of this research.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data are available on request from the corresponding author upon reasonable request.

Author Contribution Statement

Mosleh Mohammd Abualhaj: Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing, Supervision. **Sumaya Al-Khatib:** Conceptualization, Software, Validation, Resources, Data curation, Writing – original draft, Visualization. **Nida Al Shafi:** Software, Validation, Resources, Data curation, Writing – review & editing. **Iyas Qaddara:** Methodology, Formal analysis, Writing – original draft, Visualization. **Abdallah Hyassat:** Methodology, Formal analysis.

References

[1] Bokan, B., & Santos, J. (2022). Threat modeling for enterprise cybersecurity architecture. In *Systems and Information Engineering Design Symposium*, 25–30. https://doi.org/10.1109/SIEDS55548.2022.9799322

[2] Abualhaj, M. M., Al-Shamayleh, A. S., Munther, A., Alkhatib, S. N., Hiari, M. O., & Anbar, M. (2024). Enhancing spyware detection by utilizing decision trees with hyperparameter optimization. *Bulletin of Electrical Engineering and Informatics*, *13*(5), 3653–3662. https://doi.org/10.11591/eei.v13i5.7939

[3] Won, D.-O., Jang, Y.-N., & Lee, S.-W. (2023). PlausMal-GAN: Plausible malware training based on generative adversarial networks for analogous zero-day malware detection. *IEEE Transactions on Emerging Topics in Computing*, *11*(1), 82–94. https://doi.org/10.1109/TETC.2022.3170544

[4] SonicWall. (2022). *2022 SonicWall cyber threat report* [White Paper]. https://www.sonicwall.com/resources/white-papers/2022-sonicwall-cyber-threat-report

[5] Rao, A. R. (2020). A three-year retrospective on offering an embedded systems course with a focus on cybersecurity. In *IEEE Integrated STEM Education Conference*, 1–8. https://doi.org/10.1109/ISEC49744.2020.9280671

[6] Tiwari, A., Mehrotra, V., Goel, S., Naman, K., Maurya, S., & Agarwal, R. (2021). Developing trends and challenges of digital forensics. In *5th International Conference on Information Systems and Computer Networks*, 1–5. https://doi.org/10.1109/ISCON52037.2021.9702301

[7] Cook, M., Marnerides, A., Johnson, C., & Pezaros, D. (2023). A survey on industrial control system digital forensics: Challenges, advances, and future directions. *IEEE Communications Surveys & Tutorials*, *25*(3), 1705–1747. https://doi.org/10.1109/COMST.2023.3264680

[8] Pallavi, & Bharti, V. (2022). A comprehensive review of cloud forensics and blockchain-based solutions. In *6th International Conference on Electronics, Communication and Aerospace*

Technology, 749–754. https://doi.org/10.1109/ICECA55336.2022.10009188

[9] Ding, F., Zhu, G., Alazab, M., Li, X., & Yu, K. (2022). Deep-learning-empowered digital forensics for edge consumer electronics in 5G HetNets. *IEEE Consumer Electronics Magazine*, *11*(2), 42–50. https://doi.org/10.1109/MCE.2020.3047606

[10] Abualhaj, M. M., Abu-Shareha, A. A., Hiari, M. O., Alrabanah, Y., Al-Zyoud, M., & Alsharaiah, M. A. (2022). A paradigm for DoS attack disclosure using machine learning techniques. *International Journal of Advanced Computer Science and Applications*, *13*(3), 192–200.

[11] Yang, Y., Du, H., Xiong, Z., Niyato, D., Jamalipour, A., & Han, Z. (2025). Enhancing wireless networks with attention mechanisms: Insights from mobile crowdsensing. *IEEE Wireless Communications, 32*(5), 152–161.

[12] Liu, X., Fu, X., Du, X., Luo, B., & Guizani, M. (2023). Machine learning-based non-intrusive digital forensic service for smart homes. *IEEE Transactions on Network and Service Management*, *20*(2), 945–960. https://doi.org/10.1109/TNSM.2022.3224863

[13] Rizvi, S., Scanlon, M., Mcgibney, J., & Sheppard, J. (2022). Application of artificial intelligence to network forensics: Survey, challenges, and future directions. *IEEE Access*, *10*, 110362–110384. https://doi.org/10.1109/ACCESS.2022.3214506

[14] Wang, Z., Xiao, X., & Rajasekaran, S. (2020). Novel and efficient randomized algorithms for feature selection. *Big Data Mining and Analytics*, *3*(3), 208–224. https://doi.org/10.26599/BDMA.2020.9020005

[15] Almomani, O. (2021). A hybrid model using bio-inspired metaheuristic algorithms for network intrusion detection system. *Computers, Materials & Continua*, *68*(1), 409–429.

[16] Abualhaj, M. M., Al-Khatib, S. N., Al-Allawee, A., Munther, A., & Anbar, M. (2024). Enhancing network intrusion detection systems through dimensionality reduction. In *Recent Advances on Soft Computing and Data Mining: Proceedings of the Sixth International Conference on Soft Computing and Data Mining*, 244–253. https://doi.org/10.1007/978-3-031-66965-1_24

[17] Xu, L., Zhou, X., Fu, Y., Jiang, G., Yu, M., ..., & Guizani, M. (2022). Accurate and efficient performance prediction for mobile IoV networks using GWO-GR neural network. *IEEE Internet of Things Journal*, *9*(17), 16463–16471. https://doi.org/10.1109/JIOT.2022.3152739

[18] Sun, X., Zhang, Y., Tian, X., Cao, J., & Zhu, J. (2022). Speed sensorless control for IPMSMs using a modified MRAS with gray wolf optimization algorithm. *IEEE Transactions on Transportation Electrification*, *8*(1), 1326–1337. https://doi.org/10.1109/TTE.2021.3093580

[19] Ali, Z., Imran, M., & Alsulaiman, M. (2017). An automatic digital audio authentication/forensics system. *IEEE Access*, *5*, 2994–3007. https://doi.org/10.1109/ACCESS.2017.2672681

[20] Ahmed, F., Khelifi, F., Lawgaly, A., & Bouridane, A. (2020). Temporal image forensic analysis for picture dating with deep learning. In *International Conference on Computing, Electronics & Communications Engineering*, 109–114. https://doi.org/10.1109/iCCECE49321.2020.9231160

[21] Hina, M., Ali, M., Javed, A. R., Srivastava, G., Gadekallu, T. R., & Jalil, Z. (2021). Email classification and forensics analysis using machine learning. In *IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation*, 630–635. https://doi.org/10.1109/SWC50871.2021.00093

[22] Kachavimath, A. V., Nazare, S. V., & Akki, S. S. (2020). Distributed denial of service attack detection using naïve Bayes and k-nearest neighbor for network forensics. In *2nd International Conference on Innovative Mechanisms for Industry Applications*, 711–717. https://doi.org/10.1109/ICIMIA48430.2020.9074929

[23] Al Banna, M. H., Haider, M. A., Al Nahian, M. J., Islam, M. M., Taher, K. A., & Kaiser, M. S. (2019). Camera model identification using deep CNN and transfer learning approach. In *International Conference on Robotics, Electrical and Signal Processing Techniques*, 626–630. https://doi.org/10.1109/ICREST.2019.8644194

[24] Shafin, S. S., Karmakar, G., & Mareels, I. (2023). Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications. *Sensors*, *23*(11), 5348. https://doi.org/10.3390/s23115348

[25] Carrier, T., Victor, P., Tekeoglu, A., & Lashkari, A. H. (2022). Detecting obfuscated malware using memory feature engineering. In *Proceedings of the 8th International Conference on Information Systems Security and Privacy*, 177–188. https://doi.org/10.5220/0010908200003120

[26] Mezina, A., & Burget, R. (2022). Obfuscated malware detection using dilated convolutional network. In *14th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops*, 110–115. https://doi.org/10.1109/ICUMT57764.2022.9943443

[27] Fakhouri, H. N., Al-Shamayleh, A. S., Ishtaiwi, A., Makhadmeh, S. N., Fakhouri, S. N., & Hamad, F. (2024). Hybrid four vector intelligent metaheuristic with differential evolution for structural single-objective engineering optimization. *Algorithms*, *17*(9), 417. https://doi.org/10.3390/a17090417

[28] Bijoy, M. B., Pebbeti, B. P., Manoj, A. S., Fathaah, S. A., Raut, A., Pournami, P. N., & Jayaraj, P. B. (2023). Deep cleaner—A few shot image dataset cleaner using supervised contrastive learning. *IEEE Access*, *11*, 18727–18738. https://doi.org/10.1109/ACCESS.2023.3247500

[29] Al Hwaitat, A. K., Manaseer, S., Al-Sayyed, R. M. H., Almaiah, A., & Almomani, O. (2020). An investigation of digital forensics for Shamoon attack behaviour in FOG computing and threat intelligence for incident response. *Journal of Theoretical and Applied Information Technology*, *98*(7), 977–990.

[30] Zhang, C., Zhou, Y., Bi, Y., Wang, J., Liang, F., & Song, Z. (2022). Aerial target motion feature selection based on improved grey wolf optimization algorithm. In *2nd International Conference on Algorithms, High Performance Computing and Artificial Intelligence*, 154–159. https://doi.org/10.1109/AHPCAI57455.2022.10087564

[31] Purushothaman, R., Rajagopalan, S. P., & Dhandapani, G. (2020). Hybridizing Gray Wolf Optimization (GWO) with Grasshopper Optimization Algorithm (GOA) for text feature selection and clustering. *Applied Soft Computing*, *96*, 106651. https://doi.org/10.1016/j.asoc.2020.106651

[32] Almomani, O. (2020). A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms. *Symmetry*, *12*(6), 1046. https://doi.org/10.3390/sym12061046

[33] Hou, Y., Gao, H., Wang, Z., & Du, C. (2022). Improved grey wolf optimization algorithm and application. *Sensors*, *22*(10), 3810. https://doi.org/10.3390/s22103810

[34] Yang, Y., Du, H., Sun, G., Xiong, Z., Niyato, D., & Han, Z. (2025). Exploring equilibrium strategies in network games with generative AI. *IEEE Network*, *39*(5), 191–200. https://doi.org/10.1109/MNET.2024.3521887

[35] Shambour, Q., Qandeel, N., Alrabanah, Y., Abumariam, A., & Shambour, M. K. (2024). Artificial intelligence techniques for early autism detection in toddlers: A comparative analysis. *Journal of Applied Data Sciences*, *5*(4), 1754–1764. https://doi.org/10.47738/jads.v5i4.353

[36] Alraba'nah, Y., & Toghuj, W. (2024). A deep learning based architecture for malaria parasite detection. *Bulletin of Electrical Engineering and Informatics*, *13*(1), 292–299. https://doi.org/10.11591/eei.v13i1.5485

[37] Abu-Shareha, A. A., Qutaishat, H., & Al-Khayat, A. (2024). A framework for diabetes detection using machine learning and data preprocessing. *Journal of Applied Data Sciences*, *5*(4), 1654–1667. https://doi.org/10.47738/jads.v5i4.363

How to Cite: Abualhaj, M. M., Al-Khatib, S., Al Shafi, N., Qaddara, I., & Hyassat, A. (2025). Utilizing Gray Wolf Optimization Algorithm in Malware Forensic Investigation. *Journal of Computational and Cognitive Engineering*, *4*(4), 591–602. https://doi.org/10.47852/bonviewJCCE52025053