Received: March 1, 2024. Revised: May 8, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Enhancing Malware Detection through Self-Union Feature Selection Using Firefly Algorithm with Random Forest Classification

Mosleh M. Abualhaj[1]* Mahran Al-Zyoud[1] Adeeb Alsaaidah[1] Ahmad Abu-Shareha[2] Sumaya Al-Khatib[3]

[1]*Department of Networks and Cybersecurity, Al-Ahliyya Amman University, Amman, Jordan*
[2]*Department of Data Science and Artificial Intelligence, Al-Ahliyya Amman University, Amman, Jordan*
[3]*Department of Computer Science, Al-Ahliyya Amman University, Amman, Jordan*
* Corresponding author's Email: m.abualhaj@ammanu.edu.jo

Abstract: The proliferation of malware gravely threatens the security of computer systems and sensitive data. This work aims to improve malware detection by using advanced feature selection techniques. The study utilizes the Firefly Algorithm (FA) for feature selection in binary and multiclass classifications to enhance the discrimination capabilities of selected features. The selected features from the binary and multiclass classifications are combined to generate a comprehensive feature set. The Obfuscated-MalMem2022 dataset is employed in the experimental evaluation. The Random Forest (RF) method completes the classification problem. Remarkably, the results demonstrate that RF performs better with the combined feature set than with features chosen separately from the binary and multiclass classifications by the FA method. RF attains a remarkable 99.983% accuracy in binary classification, demonstrating the potency of the selected features in differentiating between malicious and benign data. Moreover, RF demonstrates an impressive accuracy of 87.304% in multiclass classification, highlighting the strength of the proposed methodology.

Keywords: Machine learning, Cybersecurity, Malware, Firefly algorithm, Random forest.

1. Introduction

Malware is software designed to damage or take advantage of users, networks, or computer systems. Its manifestations conceal distinct malicious intents and goals, from stealing confidential data to interfering with regular computer functions. Because malware is becoming more sophisticated, it can take more control over its targets and cause more serious harm, which has increased its significance [1,2]. According to Symantec's 2019 Internet Security report, one in ten URLs are found malicious, a significant increase from one in sixteen URLs found malicious the year before. This emphasizes how dangerous and common malicious links are on the Internet. Furthermore, the total ransomware infection rate decreased three years after the Wannacry ransomware attack 2017. In contrast to the general declining trend, there has been a 12% increase in the infection rate, especially within organizations. This disparity suggests that malware, which targets corporations especially, is a persistent and expanding danger [3].

The dynamic nature of malware emphasizes the continued need for strong cybersecurity defenses. Conventional methods of mitigating malware use proactive and reactive tactics to reduce the likelihood of infections and lessen their effects. These traditional techniques include, but are not limited to, email filtering, firewalls, and antivirus software [2]. Nevertheless, conventional detection and prevention techniques are challenged by the increasing sophistication of malware, which makes them inadequate in offering strong protection mechanisms [4]. Due to this deficiency, researchers and cybersecurity professionals have been exploring and developing more sophisticated methods and tools. These cutting-edge methods seek to efficiently

detect, neutralize, and lessen the effects of malware invasions [5]. With cutting-edge technologies like machine learning, the field of malware detection has seen a notable change [6-8].

Machine learning aims to develop models and algorithms that allow computers to learn from data and improve with experience [9]. The training data's variety, representativeness, and volume heavily influence how effective these models are [9,10]. In many cases, datasets exhibit high dimensionality, characterized by many features. Machine learning employs techniques like feature selection to manage and extract meaningful information from such high-dimensional datasets. Feature selection is a critical step in the machine learning pipeline, involving the choice of a subset of relevant features from the original set. The primary objectives are to enhance the model's performance, mitigate overfitting, and reduce computational requirements [10,11].

Metaheuristic algorithms are optimization techniques applied to tackle intricate problems, and they find utility in various domains, including feature selection within machine learning [12,13]. Among the plethora of metaheuristic optimization algorithms, the Firefly Algorithm (FA) stands out as it draws inspiration from the flashing behavior of fireflies. FA has gained widespread use in feature selection across diverse applications [14,15]. In this paper, we focus on employing FA to select optimal features to enhance accuracy in malware detection strategically. By leveraging the unique attributes of the Firefly Algorithm, we aim to improve the efficiency and effectiveness of feature selection specifically tailored for the challenges posed by malware detection scenarios.

This paper will specifically examine the malware that conceals itself in computer memory and is utilized to take advantage of weaknesses in the MS Windows operating system. Hence, this study suggested framework utilizes the RF classifier and the FA optimizer to identify malware. The suggested framework is referred to as RFFA-Mal. The RFFA-Mal framework suggests employing a union feature selection. The union feature selection strategy is employed to choose the most relevant features by merging the features from several subsets of features. The utilization of the union feature selection strategy may yield greater success compared to the conventional single feature selection approach. The reason for this is that union feature selection enhances the accuracy and efficiency of the feature selection process. This enhances the overall efficiency of the machine learning framework and reduces overfitting.

This paper is organized as follows. Section 2 discusses some works that have employed ML to enhance malware detection. Section 3 discusses the main components and operations of the proposed malware detection framework. Section 4 discusses the implementation environment and the proposed framework's results. Finally, Section 5 presents the conclusion.

2. Related works

This section discusses several previous studies that have been proposed to handle the issue of malware detection. The discussed studies are divided into two groups. The first group presents the achievement of various techniques on different datasets. The second group discusses the studies that have been evaluated using the MalMem-2022 dataset, which has been used to examine the method proposed in this work.

The first group presents the achievement of various techniques on different datasets. Abijah et al. [16] have introduced a diverse deep forest model to enhance malware detection and classification systems. The proposed system addresses three key aspects to improve existing malware detection approaches. Firstly, it involves the conversion of PE binary files into 2D grayscale images. Secondly, the images undergo processing in two distinct phases: the sliding window scanning phase and the cascade layering phase. Notably, the sliding window scanning phase considers critical features for improved predictions. Thirdly, the decision to continue or stop the layering process is determined based on cross-validation performance. The results of the proposed model demonstrate a high detection rate of 98.65%, 97.2%, and 97.43% for the Malimg, BIG 2015, and MaleVis [17] malware datasets, respectively. Priya et al. [18] introduce a novel approach to address the problem of malware identification on Android platforms. The suggested approach extracts features using static Android APKs (Android Package Kits) analysis. After extracting features from the Android APKs, the authors generated two combinations of permissions and conducted several experiments considering the combined extracted features. Using the Drebin dataset [19], the proposed approach attained 98.19% classification accuracy. Moreover, the suggested approach showed a 98.84% accuracy rate when with the Malgenome dataset [20]. Urooj et al. [21] have developed a framework to detect malicious Android applications, by determining and choosing functions to record and examine Android app behavior. This is achieved using AndroGuard to extract features from

binary vectors and reverse application engineering. During testing, the proposed framework achieves a 96% accuracy in the given context with a low false positive rate of 0.3, especially when larger and improved feature and sample sets are used. The study shows ensemble and strong learner algorithms perform better than other approaches when handling classifications and high-dimensional data.

The second group describes the research examined using the MalMem-2022 dataset, which has been used to examine the method proposed in this work. The researchers Luhr J. et al. [28] have proposed and compared two different methods for detecting malicious software hidden in a computer's memory. The first approach uses a deep learning model known as a Multi-Layer Perceptron (MLP). In contrast, the second approach uses a machine learning ensemble-based model that includes RF, LR, KNN, and SVM classifiers. A comparative analysis of the two approaches was carried out using the MalMem-2022 dataset, focusing on binary and multiclass classification. In binary classification, the MLP approach obtains an accuracy of 99.93%, while in multiclass classification, it achieves an accuracy of 75.91%. On the other hand, the Ensemble technique obtains a higher accuracy level than the MLP method in binary and multiclass classification, with a 99.95% and 79.11% accuracy rate, respectively. Carrier et al. [29] developed an

Table 1. Summary of the related works

Ref#	Authors (Year)	Algorithm	Dataset	Results (Accuracy)
Ref [16]	Abijah et al. (2020)	Proposed layered ensemble approach that mimics the key characteristics of deep learning.	Malimg.	98.65% binary classification
			BIG2015.	97.2% binary classification
			MaleVis.	97.43% binary classification
Ref [18]	Priya et al. (2022)	RF	Drebin	98.19% binary classification
		SVM with PCA	Malgenome	98.84% binary classification
Ref [21]	Urooj et al. (2022)	Ensemble learning	Self-created malware dataset	96% binary classification
Ref [22]	Luhr J. et al. (2022)	MLP	CLC-MalMem- 2022 dataset	99.93 % binary classification
				75.91 % multiclass classification
		Ensemble learning		99.95 % binary classification
				79.11% multiclass classification
Ref [23]	Carrier et al. (2022)	Ensemble learning		99% binary classification
Ref [24]	Mezina et al. (2022)	DCNN		99 % binary classification
				83% multiclass classification
Ref [25]	Jerbi et al. (2023)	Proposed method based on Memetic Algorithm and an Artificial Immune system		97.67 binary classification

additional approach for detecting malware and then analyzed it using the MalMem-2022 dataset. The proposed method takes the form of ensemble learning with two layers. Layer 1 of the suggested technique comprises NB, RF, and DT base learners, while layer 2 comprises LR meta-learners. Regarding accuracy, the proposed combination of base learners and meta-learners has attained the highest possible score of 99%. Mezina et al. [30] proposed a different method for detecting malware, and they then tested it using the Malmem-2022 dataset. Dilated convolutional neural networks (DCNNs) are utilized in the proposed technique, which consists of four layers. Two convolutional layers are comprised in each of the four layers, and the number of neurons in each of these layers might range from 32 to 256. With binary classification, the proposed technique achieved an accuracy of 99%, while with multiclass classification, it achieved an accuracy of 83%. Jerbi et al. [31] proposed a different approach that uses the Malmem-2022 dataset for evaluation. The suggested method uses a memetic algorithm to generate new samples of malware. After that, resilient detectors produced by an algorithm based on artificial immune systems are utilized to identify the newly acquired samples. By using binary classification, the proposed method reached an accuracy of 97.67%.

The first group of the previous studies [16, 18, 21] introduces various solutions to address the malware detection problem. However, these solutions have yet to employ feature selection techniques or the RF ML classifier to detect the malware. Furthermore, while these solutions have demonstrated high accuracy in detecting malware, the findings indicate an opportunity to develop a new approach to improve malware detection. The proposed approaches have been assessed using several datasets, excluding the CIC-MalMem-2022 dataset. This dataset specifically focuses on samples that exploit malware hidden in memory and target vulnerabilities in MS Windows. The primary objective of analyzing the research in the first group is to demonstrate the efficacy of several alternative techniques for detecting malware. The second group of previous studies [22-25] introduces other solutions for feature selection, excluding any metaheuristic optimization techniques. Furthermore, they have yet to employ the RF classifier to detect the malware. Moreover, while these solutions have demonstrated high accuracy in detecting malware, the findings indicate an opportunity to develop a new approach to improve malware detection. Table 1 summarizes the previous studies. The studies in the second group are the primary focus of this study, as they address the same problem that this work tries to address. This study suggested a framework combining an RF classifier and an FA optimizer for malware detection. Hence, the suggested framework is referred to as RFFA-Mal. The study employs FA to select features for binary and multiclass classifications, aiming to improve the discriminatory power of the selected features. The selected features from binary and multiclass classifications are combined to provide a comprehensive dataset. The proposed framework has undergone testing using the combined comprehensive dataset.

3. Proposed malware detection framework

The proposed RFFA-Mal framework uses a combination between RF classifier and FA optimizer to detect malware. This section provides an overview of the Obfuscated-MalMem2022 dataset, outlining its role in evaluating the proposed framework. Subsequently, it details the operations undertaken to preprocess the dataset for classification purposes. The feature selection process, featuring the FA, is then expounded upon. Finally, the section delves into the classification process of malware, elucidating the role of the RF classifier.

3.1 Data preparation

The Obfuscated-MalMem2022 dataset is a widely used benchmark dataset in malware detection and network security. The Obfuscated-MalMem2022 dataset is designed to facilitate the evaluation of malware systems by providing a more realistic and challenging environment for testing. It comprises 58,596 network traffic records collected in a simulated environment, with instances categorized equally into normal and malware data types. The malware types are further divided into three main types: Trojan Horse (9487 records), Spyware (10020 records), and Ransomware (9791 records). The Obfuscated-MalMem2022 dataset consists of a total of 55 features, beside the output column [2,26], as shown in Table 2.

The Obfuscated-MalMem2022 dataset contains both numerical and categorical data. The output (label) column contains categorical data that must be transformed into numbers for machine learning classifiers to work. Label encoding technique will be used to transform these labels into numbers [9]. For binary classification, the Label encoding technique transforms the normal and malware labels into 0 and 1, respectively. For multiclass classification, the Label encoding technique transforms the normal, Trojan Horse, Spyware, and Ransomware labels into

Table 2. Obfuscated-MalMem-2022 features

Feature name	#	Feature name	#	Feature name	#	Feature name
pslist.nproc	15	handles.nthread	29	malfind. protection	43	psxview.not_ in_session_ false_avg
pslist.nppid	16	handles.ndirectory	30	malfind. uniqueInjections	44	psxview.not_ in_deskthrd_ false_avg
pslist.avg_threads	17	handles.nsemaphore	31	psxview.not_ in_pslist	45	modules.nmodules
pslist.nprocs64bit	18	handles.ntimer	32	psxview.not_ in_eprocess_pool	46	svcscan.nservices
pslist.avg_handlers	19	handles.nsection	33	psxview.not_ in_ethread_pool	47	svcscan.kernel_ drivers
dlllist.ndlls	20	handles.nmutant	34	psxview.not_ in_pspcid_list	48	svcscan.fs_ drivers
dlllist.avg_dlls _per_proc	21	ldrmodules. not_in_load	35	psxview.not_ in_csrss_handles	49	svcscan.process_ services
handles.nhandles	22	ldrmodules. not_in_init	36	psxview.not _in_session	50	svcscan.shared_ process_ services
handles.avg_ha ndles_per_proc	23	ldrmodules. not_in_mem	37	psxview.not_ in_deskthrd	51	svcscan. interactive_ process_ services
handles.nport	24	ldrmodules. not_in_load_avg	38	psxview.not_ in_pslist_false_avg	52	svcscan.nactive
handles.nfile	25	ldrmodules. not_in_init_avg	39	psxview.not_ in_eprocess_ pool_false_avg	53	callbacks. ncallbacks
handles.nevent	26	ldrmodules. not_in_mem_avg	40	psxview.not_ in_ethread_ pool_false_avg	54	callbacks. nanonymous
handles.ndesktop	27	malfind. ninjections	41	psxview.not_ in_pspcid_ list_false_avg	55	callbacks. ngeneric
handles.nkey	28	malfind. commitCharge	42	psxview.not_ in_csrss_ handles_false_avg		

0, 1, 2, and 3, respectively. Besides categorical data, the Obfuscated-MalMem2022 dataset contains numerical features with varying magnitudes. Accordingly, in the normalization step, numerical features undergo Min-Max scaling to confine their values within a specified range, between 0 and 1 [9]. This scaling ensures that features with varying magnitudes contribute proportionally to model training, preventing bias towards variables with larger scales. Table 3 and Table 4 show a sample of the original data within the Obfuscated-MalMem2022 dataset and a sample of the dataset after being processed (Transformation and Normalization), respectively. By executing these transformation and normalization steps, the Obfuscated-MalMem2022 dataset is refined, laying a solid foundation for developing robust malware detection models.

Table 3. Sample of the Obfuscated-MalMem2022 dataset before preprocessing

Data Samples	Output
46, 18, 10.555556, 0, 202.84444, 1695, 39.5	Benign
48, 18, 11.54191589, 0, 243.2350436, 2174, 44.13766958	Benign
41, 140, 140.725, 0, 289.235, 1942, 49.3	Benign
38, 25, 12.41125742, 0, 320.7792418, 2574, 41.17793318	Malware
39, 15, 12.14614625, 0, 315.9389389, 2466, 59.09218218	Malware
37, 16, 11.91795892, 0, 237.7411356, 1562, 49.79587289	Malware

Table 4. Sample of the Obfuscated-MalMem2022 dataset after preprocessing

Data Samples	Output
0.145762576, 0.156172515, 0.578521534, 0, 0.25172938, 0.379375254, 0.780540758	0
0.157685633, 0.199646173, 0.64439608, 0, 0.289354507, 0.606410865, 0.896942921	0
0.123742456, 0.203348475, 0.768784019, 0, 0.223382124, 0.355202787, 0.790635083	0
0.104875535, 0.110798665, 0.645187936, 0, 0.158748754, 0.320234798, 0.604971432	1
0.107884531, 0.130679655, 0.645187945, 0, 0.178759754, 0.342043788, 0.704971512	1
0.107884531, 0.130679655, 0.598485539, 0, 0.155663757, 0.327067065, 0.679638787	1

3.2 Feature selection

Feature selection is a critical stage in machine learning, which entails selecting a subset of pertinent features from the initial feature set. Effectively selecting features can enhance interpretability, create quicker training times, and improve model performance. This paper proposes a novel self-union feature selection mechanism using FA optimization algorithm.

3.2.1. FA optimization algorithm

The selection of features for the MalMem2022 dataset represents a big step forward in the preprocessing of the dataset. Feature selection is the process of selecting subsets of relevant features from an existing set of features to improve the model's performance, reduce the probability of overfitting, and raise the readability of the model. In the process of choosing a feature selection method, the specific features of the dataset should be taken into consideration. The FA method is a well-known metaheuristic optimization method used for feature selection. FA is based on the core principle that fireflies interact with one another and attract one another through flashing. Those fireflies with a greater variety of colors are considered more appealing and more likely to attract other fireflies. The algorithm duplicates this behavior to achieve the goal of identifying optimal or nearly optimal solutions [14,15,27,28].

```
1.  for each firefly i:
2.      X_i = random_position()
3.  for each firefly i:
4.      brightness[i] = evaluate_fitness(X_i)
5.  calculate_initial_attractiveness()
6.  while not convergence_criteria_met():
7.      for each firefly i:
8.          for each firefly j:
9.              if brightness[j] > brightness[i]:
10.                 move_firefly_towards(i, j)
11.     update_brightest_firefly_positions()
12.     update_attractiveness()
13. distance = calculate_distance(X_i, X_j)
14. attractiveness = calculate_attractiveness(distance)
15. randomization_factor = generate_random_number()
16. X_i = X_i + attractiveness * (X_j - X_i) + alpha *
    (randomization_factor - 0.5)
17. brightest_firefly = find_brightest_firefly()
18. update_global_best_position(brightest_firefly)
19. for each firefly i:
20.     brightness[i] = evaluate_fitness(X_i)
21. return convergence_criteria_satisfied
```

Figure. 1 Pseudocode of FA algorithm

The FA optimization algorithm performs several operations to find the relevant features for malware detection. The pseudocode in Fig. 1 shows for the operations performed by the FA algorithm [14,15,27,28]. Line 1 and 2 initialize positions randomly in the search space. Line 3 and 4 calculate attractiveness based on the objective function. Line 5 to 12 calculate and update the brightest firefly positions. Line 13 to 15 move firefly i towards brighter firefly j. Line 16 updates the position of firefly i. Line 17 and 18 identify and update the position of the brightest firefly. Line 19 and 20 update attractiveness based on the new positions. Line 21 check convergence criteria (e.g., maximum iterations or target fitness reached).

3.2.2. Self-Union feature selection using FA

The FA optimization algorithm is used for feature selection in the proposed RFFA-Mal framework. The FA algorithm has selected 14 out of

Table 5. Selected feature by different methods

Method	Selected features (feature number)
Binary	1, 16, 19, 23, 25, 27, 29, 30, 37, 41, 42, 46, 49, 53
Multiclass	1, 4, 6, 8, 14, 16, 17, 18, 19, 20, 21, 25, 27, 28, 29, 31, 34, 35, 37, 39, 42, 44, 46, 47, 49, 52
Union of Binryclass & Multiclass	1, 4, 6, 8, 14, 16, 17, 18, 19, 20, 21, 23, 25, 27, 28, 29, 30, 31, 34, 35, 37, 39, 41, 42, 44, 46, 47, 49, 52, 53

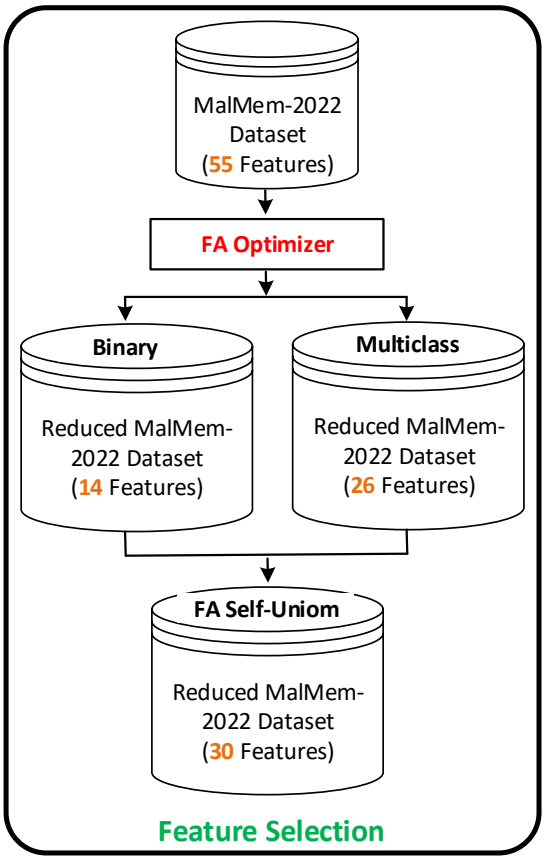

Figure. 2 Feature selection process

55 features for binary classification. The FA algorithm has selected 26 out of 55 features for multiclass classification. Table 5 lists the selected features by FA for both binary and multiclass classification. Besides typical feature selection, the RFFA-Mal framework proposes to use a union feature selection. The union feature selection approach is used to select the most pertinent features by combining the features from different subsets of features. The union feature selection approach could be more successful than the typical single feature selection approach. This is because union feature selection increases the accuracy and efficiency of the feature selection process. This improves the overall performance of the machine learning framework and decreases overfitting [29].

Union features selection typically combines features from different selection algorithms [29]. The RFFA-Mal framework proposes to use a novel self-union feature selection algorithm that combines features from one features selection algorithm, the FA algorithm. The proposed self-union feature selection algorithm works as follows. First, the FA optimizer performs feature selection for binary classification. Then, the FA optimizer performs feature selection for multiclass classification. Finally, the union of these features is combined together in one subset. This will improve the performance of the RFFA-Mal framework and decrease overfitting. The proposed self-union feature selection mechanism is illustrated in Fig. 2. Table 5 lists the union of features from binary and multiclass classification.

3.3 Classification

In the previous steps (Section 3.1 and 3.2), the data has been processed and prepared for the classification, distinguish the malware and benign data. The RF classifier will be used for the classification purpose. RF is a popular machine learning algorithm that can be used for classification problems in ML, such as distinguishing between the malware and benign data. RF is based on the

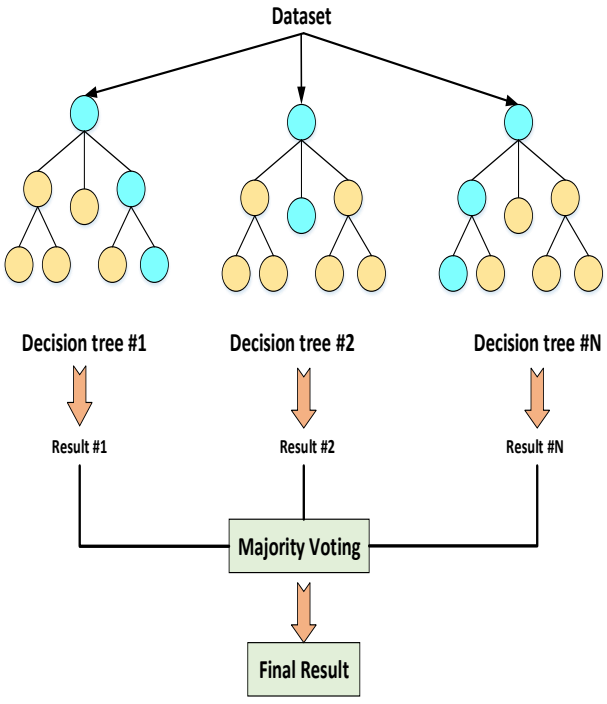

Figure. 3 RF classifier

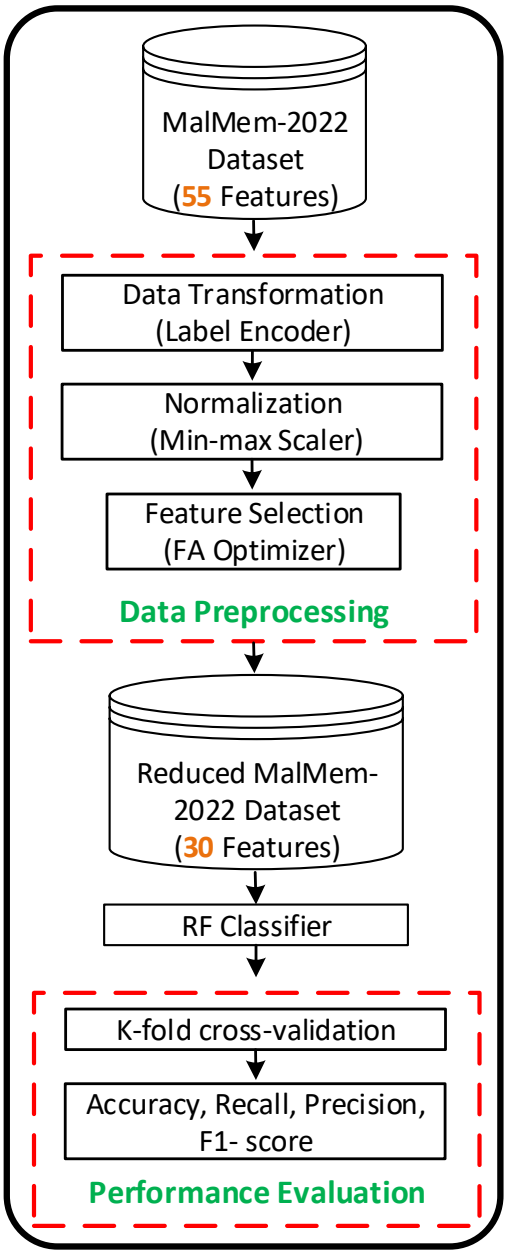

Figure. 4 RFFA-Mal framework

concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. Whereas, RF contains a number of decision trees on various subsets of the given dataset, as shown in Fig. 3. The ensemble learning nature of the RF makes it preferable over most of the other classifiers. Leveraging the power of ensemble learning b helps mitigate overfitting, reduce variance, and improve overall model accuracy compared to individual models [30-32].

At this stage, the proposed RFFA-Mal framework is ready to detect malware. Fig. 4 demonstrates the RFFA-Mal framework. The performance of the RFFA-Mal framework will be evaluated in the following section.

4. Implementation, result and discussion

The section includes the implementation environment, performance evaluation criteria, results, and discussion of the proposed RFFA-Mal framework.

4.1 Implementation environment

The proposed framework was conducted on a desktop with Intel Core i7-2600 Processor (3.4GHz and 8M Cache), 32 GB DDR5-3200 memory, SSD M.2 256GB, NVIDIA GeForce 12GB GDDR6 graphics card, and Ubuntu 20.01.3 LTS O.S. Python was used to test and evaluate the proposed framework. To implement an RF classifier for detecting malware in Python, you'll primarily use the 'scikit-learn' library (sklearn). 'Scikit-learn' provides a comprehensive set of classes and methods for building and evaluating RF classifiers for malware detection. 'RandomForestClassifier', 'train_test_split', 'accuracy_score', and 'classification_report', besides other libraries, have been used in this work [33].

K-fold cross-validation is used to ensure consistent performance of the framework across different subsets of the data, reducing the risk of overfitting to a particular train-test split. The value of K is set to five. In 5-fold cross-validation, the dataset is divided into five equally sized folds. The cross-validation process is then repeated five times, with each fold used once as a validation set while the remaining four folds form the training set [31,32].

4.2 Performance evaluation criteria

The proposed RFFA-Mal framework is evaluated the confusion matrix tool. Evaluating the performance of the RFFA-Mal framework for detecting malware involves using a confusion matrix and related metrics to assess how well the framework is performing in terms of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). TP is the instances where the framework correctly identifies a malware sample. TN is the instances where the framework correctly identifies a non-malicious sample. FP is the instances where the model incorrectly identifies a non-malicious sample as malicious. FN is the instances where the model incorrectly identifies a malicious sample as non-malicious. Confusion matrix provides a comprehensive breakdown of the predicted and actual classes, allowing for the calculation of various performance metrics. These metrics are Accuracy, Recall, Precision, F1- score.

Accuracy represents the overall correctness of the model. Accuracy is calculated using Eq. (1). Recall measures the ability of the model to capture all positive instances, indicating how many of the actual positive instances were correctly predicted. Recall is calculated using Eq. (2). Precision measures the accuracy of positive predictions, indicating how many of the predicted positive instances were actually positive. Precision is calculated using Eq. (3). The F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics. F1-score is calculated using Eq. (4) [31-33].

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

$$Recall = \frac{TP}{(TP+FN)} \quad (2)$$

$$Precision = \frac{TP}{(TP+FP)} \quad (3)$$

$$F1-Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (4)$$

4.3 Results and discussion

The proposed RFFA-Mal framework was tested with binary and multiclass classification types. For each type, the subset of features selected by the FA algorithm was tested using two distinct approaches: Typical Feature Selection (FA-T-FS) (refer to Section 3.2.2) and Union Feature Selection (FA-U-FS) (refer to Section 3.2.2). The RF classifier will be used to evaluate the RFFA-Mal framework (refer to Section 3.3). Besides, the proposed FA-U-FS have been compared with state-of-the-art (SOTA) methods.

4.3.1. Binary classification

Figs. 5, 6, 7, and 8 illustrate Accuracy, Recall, Precision, and F1-score performance metrics for binary classification, respectively. Across all four metrics, the typical FA-T-FS method demonstrated an impressive value of 99.974%. In contrast, the proposed union FA-U-FS method surpassed this, achieving a higher value of 99.983%. Consequently, the results obtained through the proposed union FA-U-FS method surpass those achieved by the typical FA-T-FS method, indicating its superior performance in binary classification based on the evaluated metrics.

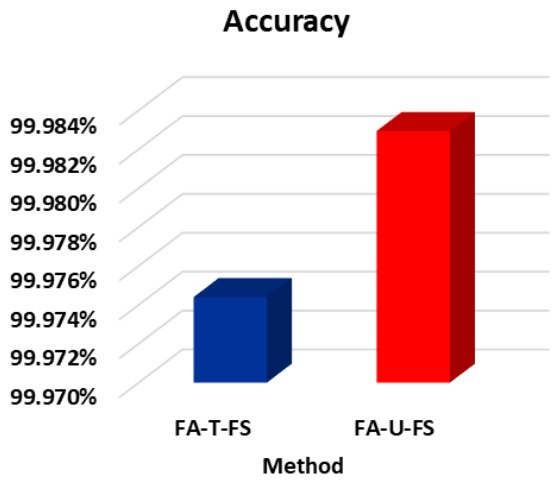

Figure. 5 Accuracy of the RFFA-Mal framework with binary classification

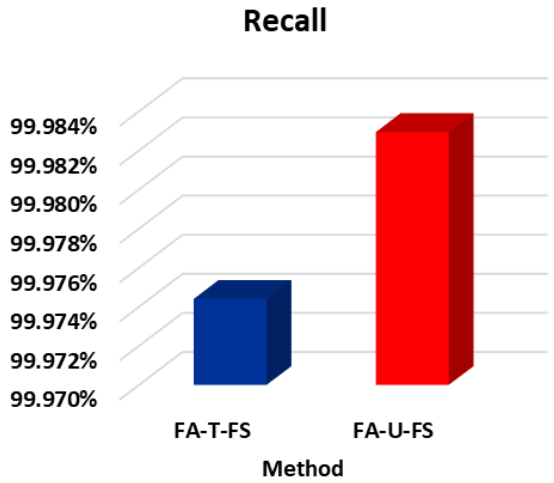

Figure. 6 Recall of the RFFA-Mal framework with binary classification

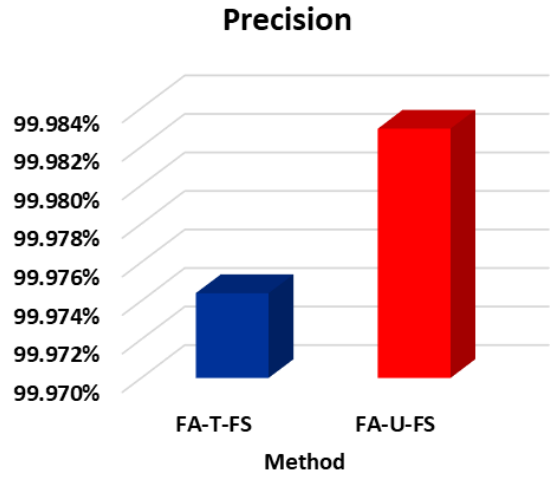

Figure. 7 Precision of the RFFA-Mal framework with binary classification

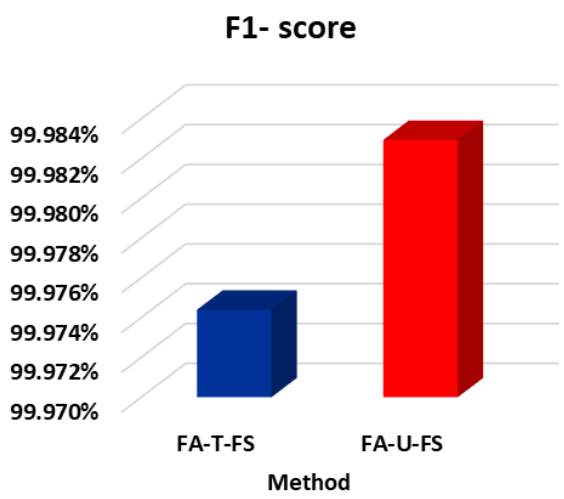

Figure. 8 F1-score of the RFFA-Mal framework with binary classification

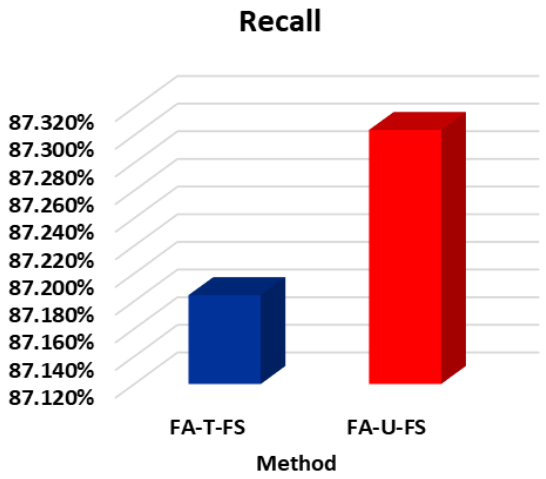

Figure. 10 Recall of the RFFA-Mal framework with multiclass classification

4.3.2. Multiclass classification

Figs. 9, 10, 11, and 12 present the performance metrics of Accuracy, Recall, Precision, and F1-score for multiclass classification, respectively. In all four metrics, the typical FA-T-FS method demonstrated a performance of 87.184%. In contrast, the proposed union FA-U-FS method achieved a higher performance of 87.304%, signifying an improvement of 0.120% over the typical FA-T-FS method. Consequently, the proposed union FA-U-FS method outperforms the typical FA-T-FS method in multiclass classification based on the evaluated metrics.

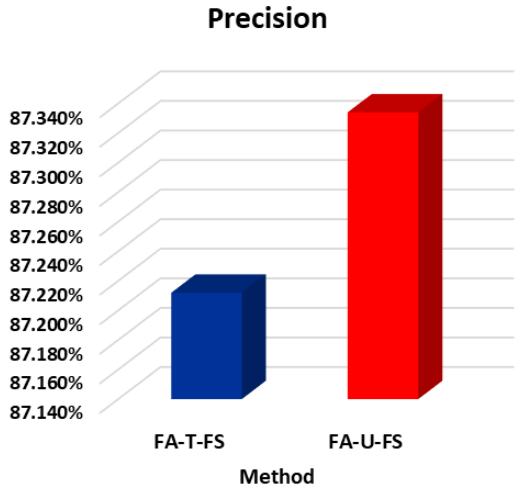

Figure. 11 Precision of the RFFA-Mal framework with multiclass classification

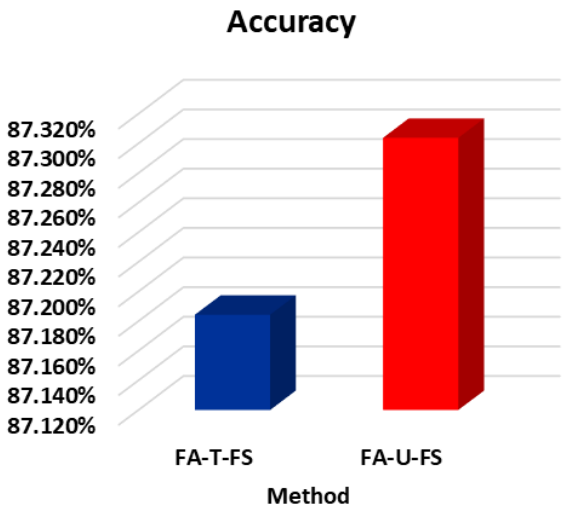

Figure. 9 Accuracy of the RFFA-Mal framework with multiclass classification

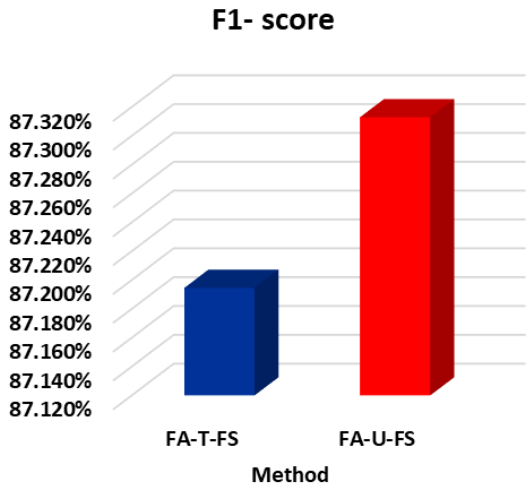

Figure. 12 F1-score of the RFFA-Mal framework with multiclass classification

In summary, these metrics serve as evaluative benchmarks for the RFFA-Mal framework, employing the RF classifier and featuring the FA-T-FS and FA-U-FS methods for feature selection. With binary classification, the RFFA-Mal framework obtains an excellent result of 99.983%, while with multiclass classification, it achieves 87.304%. The thorough display of these indicators provides a sophisticated evaluation of the framework's effectiveness across various performance metrics.

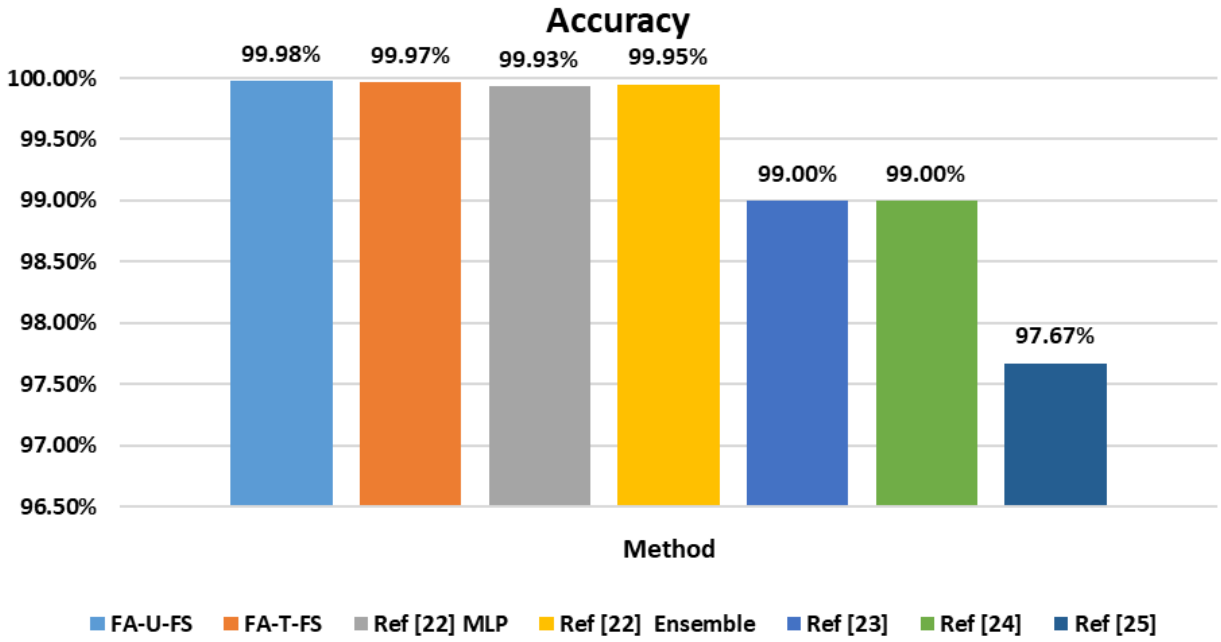

Figure. 13 Accuracy of the FA-U-FS technique in comparison to the SOTA techniques on CIC-MalMem-2022 dataset (binary classification)

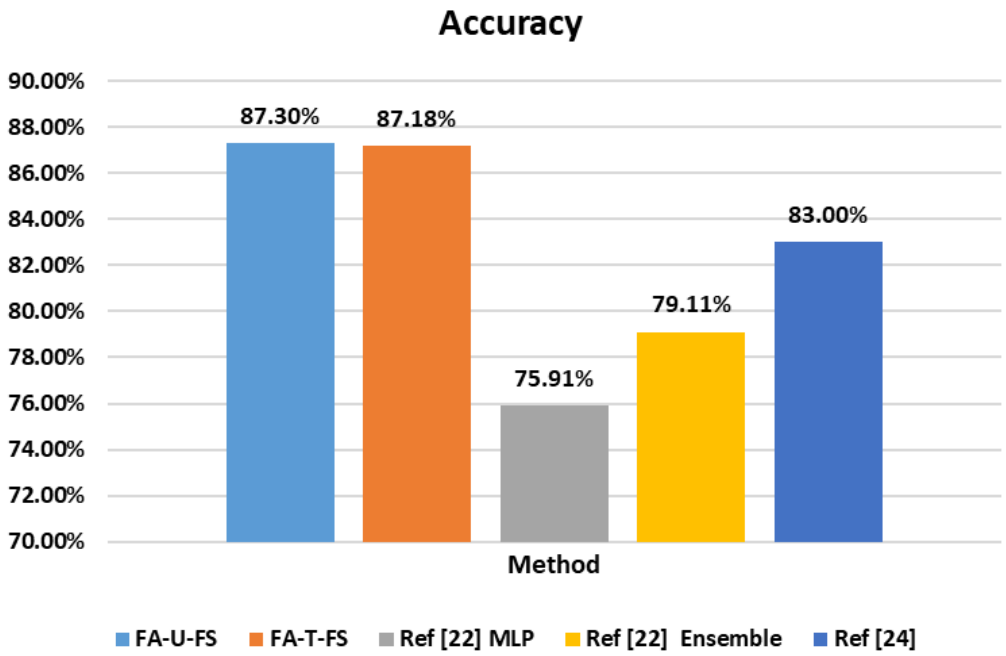

Figure. 14 Accuracy of the FA-U-FS technique in comparison to the SOTA techniques on CIC-MalMem-2022 dataset (multiclass classification)

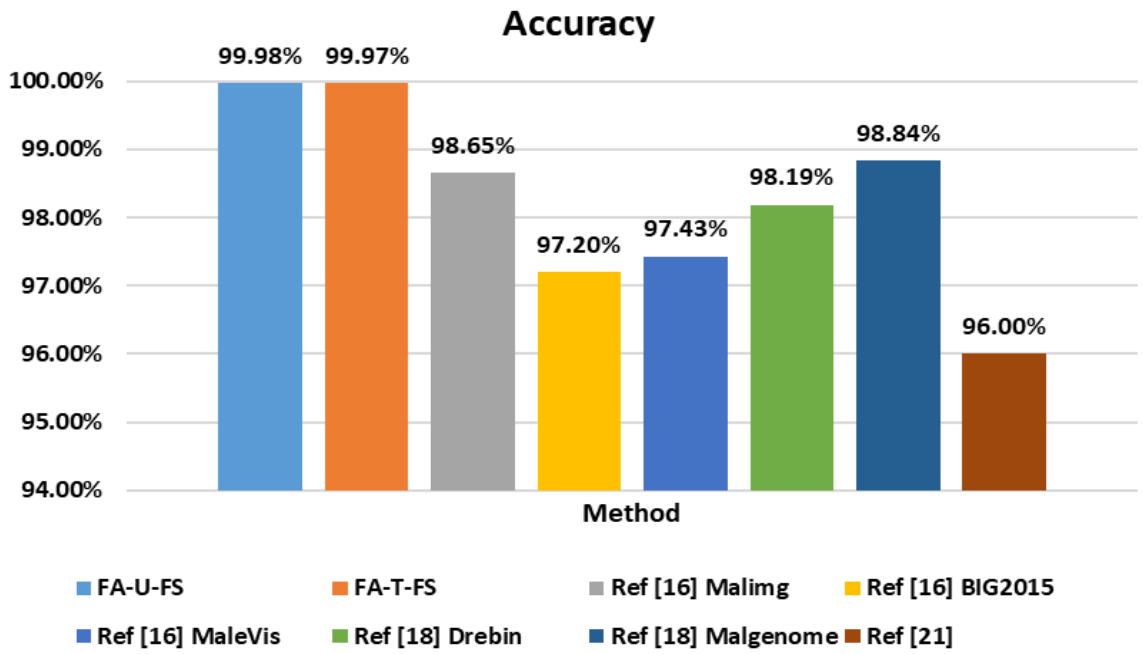

Figure. 15 Accuracy of the FA-U-FS technique in comparison to the SOTA techniques on various dataset (binary classification)

4.3.3. Performance assessment of the RFFA-Mal framework against the SOTA techniques

Figs. 13 and 14 show the accuracy of the suggested FA-U-FS technique in comparison to the SOTA techniques on binary and multiclass classifications utilizing the CIC-MalMem-2022 dataset, respectively. Binary and multiclass classifications on the CIC-MalMem-2022 dataset demonstrate that the FA-U-FS technique outperforms the SOTA techniques. Fig. 15 shows the accuracy of the suggested FA-U-FS technique in comparison to the SOTA techniques across various datasets, focusing on binary classification. Notably, both the FA-U-FS technique exhibit superior performance in binary classification compared to other approaches.

5. Conclusion

This study emphasizes the value of novel strategies for tackling the growing problems caused by malware in cybersecurity. This study developed a new RFFA-Mal framework for disclosing malware using the RF classifier and FA algorithm. The RFFA-Mal framework has significantly improved malware detection accuracy by leveraging the FA algorithm for feature selection and employing a unified feature set from binary and multiclass classifications in conjunction with the RF classification algorithm. Among the noteworthy accomplishments is an exceptional binary classification accuracy of 99.983%, demonstrating the strong discriminatory capacity of the chosen features to discern between malicious and benign data. Moreover, the impressive multiclass classification accuracy of 87.304% highlights the flexibility and dependability of the suggested methodology in managing various malware classifications. Combining characteristics from binary and multiclass classifications has been crucial since it produces better outcomes than using the features independently. This improves the feature set's comprehensiveness and makes the malware detection system more all-encompassing and efficient. The evaluation environment provided by the Obfuscated-MalMem2022 dataset is both realistic and demanding, serving as a close replica of real-world situations to validate the suggested technique.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Following are the contribution of authors: conceptualization, Mosleh M. Abualhaj and Ahmad Abu-Shareha; methodology, Mosleh M. Abualhaj and Ahmad Abu-Shareha; software, Sumaya Al-Khatib; validation, Mahran Al-Zyoud, and Adeeb Alsaaidah; formal analysis, Mosleh M. Abualhaj and Adeeb Alsaaidah; investigation, Mosleh M.

Abualhaj and Adeeb Alsaaidah; resources, Sumaya Al-Khatib; data curation, Sumaya Al-Khatib; writing—original draft preparation, Mosleh M. Abualhaj and Ahmad Abu-Shareha; writing—review and editing, Mosleh M. Abualhaj and Mahran Al-Zyoud; visualization, Mosleh M. Abualhaj and Mahran Al-Zyoud; supervision, Mosleh M. Abualhaj and Mahran Al-Zyoud; project administration, N/A; funding acquisition, N/A".

References

[1] J. H. Park, "Symmetry-adapted machine learning for information security", *Symmetry*, Vol. 12, No. 6, p. 1044, 2020.

[2] M. Abualhaj, A. Abu-Shareha, Q. Shambour, A. Alsaaidah, S. Al-Khatib, and M. Anbar, "Customized K-nearest neighbors' algorithm for malware detection", *International Journal of Data and Network Science*, Vol. 8, No. 1, pp. 431-438, 2024.

[3] T. Li, Y. Liu, Q. Liu, W. Xu, Y. Xiao, and H. Liu, "A malware propagation prediction model based on representation learning and graph convolutional networks", *Digital Communications and Networks*, Vol. 9, No.5, pp. 1090-1100, 2023.

[4] M. Belaoued, A. Derhab, S. Mazouzi, and F. Khan, "MACoMal: A multi-agent based collaborative mechanism for anti-malware assistance", *IEEE Access*, Vol. 8, pp. 14329-14343, 2020.

[5] A. Ross, and D. Morgan, "Malware Mitigation Using Host Intrusion Prevention in the Enterprise", In: *Proc. of Security and Management*, pp. 46-54, 2004.

[6] E. Kidmose, M. Stevanovic, and J. Pedersen, "Correlating intrusion detection alerts on bot malware infections using neural network", In: *Proc. of 2016 International Conf. On Cyber Security And Protection Of Digital Services (Cyber Security)*, pp. 1-8, 2016.

[7] V. Vasani, A. Bairwa, S. Joshi, A. Pljonkin, M. Kaur, and M. Amoon, "Comprehensive analysis of advanced techniques and vital tools for detecting malware intrusion", *Electronics*, Vol. 12, No. 20, p. 4299, 2023.

[8] M. Kolhar, F. Al-Turjman, A. Alameen, and M. Abualhaj, "A three layered decentralized IoT biometric architecture for city lockdown during COVID-19 outbreak", *IEEE Access*, Vol. 8, pp. 163608-163617, 2020.

[9] M. Abualhaj, A. Abu-Shareha, M. Hiari, Y. Alrabanah, M. Al-Zyoud, and M. Alsharaiah, "A Paradigm for DoS Attack Disclosure using Machine Learning Techniques", *International Journal of Advanced Computer Science and Applications*. Vol. 13, No. 3, 2022.

[10] H. Zhao, Q. Hu, P. Zhu, Y. Wang, and P. Wang, "A recursive regularization based feature selection framework for hierarchical classification", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 33, No. 7, pp. 2833-2846, 2019.

[11] T. Zhang, T. Zhu, P. Xiong, H. Huo, Z. Tari, and W. Zhou, "Correlated differential privacy: Feature selection in machine learning", *IEEE Transactions on Industrial Informatics*, Vol. 16, No. 13, pp. 2115-2124, 2019.

[12] L. Jovanovic, D. Jovanovic, M. Antonijevic, B. Nikolic, N. Bacanin, M. Zivkovic, and I. Strumberger, "Improving phishing website detection using a hybrid two-level framework for feature selection and xgboost tuning", *Journal of Web Engineering*, Vol. 22, No. 3, pp. 543-574, 2023.

[13] O. Alyasiri, Y. Cheah, A. Abasi, and O. Al-Janabi, "Wrapper and hybrid feature selection methods using metaheuristic algorithms for English text classification: A systematic review", *IEEE Access*, Vol. 10, pp. 39833-39852, 2022.

[14] W. Liu, P. Li, Z. Ye, and S. Yang, "September. A node deployment optimization method of wireless sensor network based on firefly algorithm", In: *Proc. of 2021 IEEE 4th International Conf. on Advanced Information and Communication Technologies (AICT)*, pp. 167-170, 2021.

[15] S. Bazi, R. Benzid, Y. Bazi, and M. Rahhal, "A fast firefly algorithm for function optimization: application to the control of BLDC motor", *Sensors*, Vol. 21, No. 16, p. 5267, 2021.

[16] S. Roseline, S. Geetha, S. Kadry, and Y. Nam, "Intelligent vision-based malware detection and classification using deep random forest paradigm", *IEEE Access*, Vol. 8, pp. 206303-206324, 2020.

[17] A. Bozkir, A. Cankaya, and M. Aydos, "Utilization and comparision of convolutional neural networks in malware recognition", In: *Proc. of 2019 27th Signal Processing and Communications Applications Conference (SIU)*, pp. 1-4, 2019.

[18] P. Raghuvanshi, and J. Singh, "Android Malware Detection Using Machine Learning Techniques", In: *Proc. Of 2022 International Conf. on Computational Science and Computational Intelligence (CSCI)*, pp. 1117-1121, 2022.

[19] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket", *In Ndss*, Vol. 14, pp. 23-26, 2014.

[20] S. Yerima, and S. Sezer, "Droidfusion: A novel multilevel classifier fusion approach for android malware detection", *IEEE transactions on cybernetics*, Vol. 49, No. 2, pp. 453-466, 2018.

[21] B. Urooj, M. Shah, C. Maple, M. Abbasi, and S. Riasat, "Malware detection: a framework for reverse engineered android applications through machine learning algorithms", *IEEE Access*, Vol. 10, pp. 89031-89050, 2022.

[22] J. Luhr, and H. Hallqvist, "Fast Classification of Obfuscated Malware with an Artificial Neural Network", 2022.

[23] T. Carrier, P. Victor, A. Tekeoglu, and A. H. Lashkari, "Detecting Obfuscated Malware using Memory Feature Engineering", In: *Proc. of Icissp*, pp. 177-188, 2022.

[24] A. Mezina, and R. Burget, "Obfuscated malware detection using dilated convolutional network", In: *Proc. of 2022 14th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 110-115, 2022.

[25] M. Jerbi, Z. C. Dagdia, S. Bechikh and L. B. Said, "Immune-Based System to Enhance Malware Detection", In: *Proc. of IEEE 2023 Congress on Evolutionary Computation*, pp. 1-8, 2023.

[26] T. Carrier, P. Victor, A. Tekeoglu, and A. Lashkari, "Detecting Obfuscated Malware using Memory Feature Engineering", In: *Proc. of the 8th International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 177-188, 2022.

[27] J. Pye, B. Issac, N. Aslam, and H. Rafiq, "Android malware classification using machine learning and bio-inspired optimisation algorithms", In: *Proc. of 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 1777-1882, 2020.

[28] A. Gupta, and P. Padhy, "Modified Firefly Algorithm based controller design for integrating and unstable delay processes", *Engineering Science and Technology, an International Journal*, Vol. 19, No. 1, pp. 548-558, 2016.

[29] Ü. Çavuşoğlu, "A new hybrid approach for intrusion detection using machine learning methods", *Applied Intelligence*, Vol. 49, pp. 2735-2761, 2019.

[30] Y. Ren, X. Zhu, K. Bai, and R. Zhang, "A New Random Forest Ensemble of Intuitionistic Fuzzy Decision Trees", *IEEE Transactions on Fuzzy Systems*, Vol. 31, No. 5, pp. 1729-1741, 2023.

[31] H. Al-Mimi, N. Hamad, and M. Abualhaj, "A Model for the Disclosure of Probe Attacks Based on the Utilization of Machine Learning Algorithms", In: *Proc. of 2023 10th International Conf. on Electrical and Electronics Engineering (ICEEE)*, pp. 241-247, 2023.

[32] H. Al-Mimi, N. Hamad, M. Abualhaj, S. Al-Khatib, and M. Hiari, "Improved Intrusion Detection System to Alleviate Attacks on DNS Service", *Journal of Computer Science*, Vol. 19, No. 12, pp. 1549-1560, 2023.

[33] H. Al-Mimi, N. Hamad, M. Abualhaj, M. Daoud, A. Al-dahoud, and M. Rasmi, "An Enhanced Intrusion Detection System for Protecting HTTP Services from Attacks", *International Journal of Advances in Soft Computing & Its Applications*, Vol. 15, No. 3, 2023.