Contents lists available at GrowingScience

International Journal of Data and Network Science

homepage: www.GrowingScience.com/ijds

Optimal feature selection based on OCS for improved malware detection in IoT networks using an ensemble classifier

Mangayarkarasi Ramaiah[a], Vanmathi Chandrasekaran[a], Padma Adla[a], Asokan Vasudevan[b], Mohammad Faleh Ahmmad Hunitie[b] and Suleiman Ibrahim Shelash Mohammad[c,d*]

[a]*School of Computer Science Engineering and Information Systems, Vellore Institute of Technology, Vellore. Tamil Nadu, India*
[b]*Faculty of Business and Communications, INTI International University, 71800 Negeri Sembilan, Malaysia*
[c]*Research follower, INTI International University, 71800 Negeri Sembilan, Malaysia*
[d]*Department of Business Administration, Business School, Al al-Bayt University, Jordan*

C H R O N I C L E

Article history:
Received: April 3, 2024
Received in revised format: May 28, 2024
Accepted: June 24, 2024
Available online: June 24, 2024

Keywords:
Feature selection
K-fold cross-validation
Machine learning
Ensemble learning
Malware attack
IoT

A B S T R A C T

The increasing amount of IoT devices increases the size of network traffic data, causing an increase in the incidence of security breaches in IoT networks. Cybercriminals have developed malware to compromise the security of sensitive data, among other cyber threats. In the presence of inadequate and robust security mechanisms, sensitive data is prone to vulnerability. Hence, protecting data in the IoT environment is becoming a mandatory task. Various approaches have addressed malware detection using network data features. However, there is still room for improvement in developing superior techniques and utilizing more comprehensive datasets. This paper presents a novel lightweight ensemble voting classifier to detect malware traffic by deploying the best possible network data. The merits of the correlation coefficient and Opposition-Based Crow Search Algorithm (OCS) have been leveraged to compute the best possible features. Another advantage of this proposed experiment is its focus on a dataset tailored to malware traffic features. This focus enables highly accurate malware detection. After feature selection using OCS, the proposed malware classifier is trained and validated with both 5-fold and 10-fold cross-validation techniques. The tested results confirm that the presented malware classifier performs best using a minimal feature set, which is highly advantageous for IoT networks due to resource constraints.

1. Introduction

The deployment of IoT devices is becoming increasingly essential in today's smart environments to implement a wide range of services that improve our lives. The inclusion of IoT devices has surged in almost all sectors, like energy, transportation systems, education, government, healthcare, and industry. Data produced by IoT devices creates vulnerabilities that cyber attackers can exploit. The protocols and standards used by many IoT devices and networks are inherently susceptible to cyberattacks due to weaknesses in security design. While cyberattacks come in many forms, malware acts as a silent but significant weapon, compromising the security of vast networks of interconnected devices within the IoT. The malware classification techniques were broadly categorized into static, dynamic, and hybrid approaches (Abusitta et al., 2021). Static procedures are more mature compared to other systems. A static approach inspects the new application without executing it, and the methods consider the permissions associated with the application and API calls (Almin & Chatterjee, 2015). Most static malware classification relies on domain insight for portable executables, which is good enough in the context of known attacks (Rao & Hande 2017). IDS works to analyze the output through bytes, n-grams, and strings. Collecting domain

* Corresponding author.
E-mail address dr_sliman73@aabu.edu.jo (S.I.S. Mohammad)

ISSN 2561-8156 (Online) - ISSN 2561-8148 (Print)

doi: 10.5267/j.ijdns.2024.6.018

knowledge for such a technique is practically infeasible; due to this fact, this category of methods fails to gain much attention (Raff et al., 2018). The system extracts the features from portable executable files to train machine-learning models to complement the detection process (Saxe & Berlin 2015; Raff et al., June 2018; Mohy-eddine et al., 2023). Dynamic behavioral approaches check the application code at runtime to track the history of various system calls and combine the merits of neural network structure to enhance the involved procedures (Canfora et al., 2015; Huang et al., 2016; Pascanu et al., 2015; Shibahara et al., 2016). Techniques include a combination of static and dynamic hybrid methods. For example, Android malware detection attempted by Yuan et al. (2024) using application's static and dynamic features to design machine learning-based models that feed deep learning models (Yuan et al., 2024). Recently, zero-day malware has become a significant threat to the cybersecurity world. This type of malware exploits unknown vulnerabilities to penetrate the system. To analyze the malware code, IDS uses the Cuckoo sandbox and generates features that are fed into feature extraction (Yoo et al., 2021). Installing dynamic complex models (Kim et al., 2023) shows better results in detecting zero-day malware. Researchers use image processing techniques to detect malicious code in candidate applications by analyzing its payload through visual representation. Such methods convert the candidate file into a binary image to train a SOINN based model to detect the malicious payload deviation (Baptista et al., 2019). Tested results show that SOINN enabled malware detection detects malicious code at the earliest possible and similarly, the network traffic data (Bendiab et al., 2020) is collected and converted into RGB images through the Binvis tool. Dataset includes 1000 Binvis images with normal and malignant labels. The Binvis images were used to train the optimized ResNet50 framework for the candidate task. And the obtained testing accuracy is 94.5%. The DeepMal framework (Marín et al., 2021), considers the raw byte stream of network data as its input to detect malicious traffic. Since the proposed model works on raw byte streams, it does not require insight about the network traffic. To derive the spatiotemporal pattern of the raw byte stream, DeepMal combines the merits of CNN and Recurrent Neural networks. The tested results reveal that models built upon raw flow-based stream data showed better results than the one built upon raw packet-based stream data. In machine learning model development, careful selection of dominant features is essential. Selecting dominant features from a static feature set requires domain expertise, but there's no guarantee that human-chosen features will always lead to improved model performance. Hence, Deep learning-based Feature Selection was (DQFSA) built upon reinforcement learning has been presented by (Fang et al., 2019) to select the differentiated features from the dataset.

The results of the experiment reveal that the suggested framework outperforms all the current baseline methods in terms of numeric metrics. Furthermore, research confirms that wrapper-based metaheuristic methods are highly effective in facilitating feature selection. Wrapper-based metaheuristic algorithms were used by (Chakravarthy et al., 2021) to select dominant features for Android malware detection. Machine learning models were built using features selected by various wrapper-based methods. The results showed that models using features selected by the firefly algorithm performed well on the CICInvesAndMal2019 dataset, which has 4119 features. Similarly, Particle Swarm optimization has been used by (Azad et al., 2022) to select the influential features upon the experimented dataset CICAndMal2017. Neural network-based model has been built upon the chosen feature derived through the wrapper method. For feature selection, PSO employs a wide-ranging search across the high-dimensional feature space to identify the most informative features for a machine learning model. Hence, the candidate proposal attempts to leverage the merits of the wrapper established metaheuristic technique, Opposition-Based Crowd Search Algorithm (OCS) to select the dominant features. Diverse ML models were built upon the features derived through OCS. Among them, the results obtained by the RF, ADB and XGB results are promising, hence, the candidate malware traffic classification framework uses these three tree-based models to improve the classification accuracy.

The major contributions of the study presented in this manuscript are as follows.

- Investigation on the merits of ML, DL, and FL for malware detection, along with the importance of feature engineering also reported
- Opposition-Based Crowd Search Algorithm is used to minimize the chances of local minima, while selecting the best features.
- The tested results obtained from diverse machine learning models are leveraged to build an ensemble-voting malware traffic classifier.
- Lightweight malware traffic is designed to make it suitable for IoT networks with minimal features.

The rest of the paper is organized as follows. Section 2 discusses the related works. Section 3 discusses the bench-marking works focused on feature engineering related to the candidate problem. Section 4 presents the proposed methodology. Section 5 discusses the experimented outcomes. Finally, section 6 concludes the paper.

2. Malware Classification

Cyber threats due to malicious code have been rapidly increasing as internet usage continues to rise. Internet users frequently access various applications such as internet banking, e-commerce websites, online reservations, and more, making them vulnerable to cybercriminals who design malware and code to steal sensitive information or compromise systems. Malware could be identified by its method of infecting devices or networks (Kavitha & Muruganantham, 2021). The literature presents several approaches for detecting malware, consisting of signature-based, behavior-based, hybrid, and memory content analysis. To distinguish malware from standard applications, countermeasure methods utilize different sets of features. In

addition to state features such as static or dynamic, the technique that employs these features to detect malware plays a crucial role. This section analyzes the strengths of existing malware classifiers built using machine learning (ML), deep learning (DL), and federated learning (FL).

2.1 Malware Classification Using Machine Learning

This section discusses the use of machine learning techniques for classifying malicious code in malware detection. The technique of signature-based detection is simple and efficient, and it is often complemented by machine learning approaches. Researchers in (Mahajan et al., 2019) focus on using signatures of the target file to identify malware. Features extracted from malware are pre-processed and then fed into machine learning models to detect and categorize malware into appropriate families. Experimentation has been carried out using KNIME and Orange tools. Tested results show that RF based malware family classifiers outperform those of others. Suitable feature selection (Ramaiah et al., 2021; Mangayarkarasi et al., 2023; Vishnukumar and Ramaiah 2024) methods are another challenge associated with feature engineering. Information gain was used to select the best features upon uploading the malicious and benign files on virus tool and cuckoo-sanbox (Babaagba et al., 2019) Both supervised and unsupervised machine learning models were analyzed with and without feature selection. Among the various models, the Information gain-enabled Multi-layer Perceptron (MLP) showed the best accuracy for malware family detection. Detection of trojans and spyware is yet to be included. Al-Kasassbeh et al. (2020) presented framework underscores the practical implications of feature selection in malware detection. These dominant features were chosen upon the belief that different parts of a PE file's characteristics tend to be more related to each other than to the output class labels.

Cuckoo Sandbox has been used in (Kim et al., 2020) to gather malware characteristics. Five groups are created from the attributes of the samples. Recursive feature elimination, or RFE, is first used to calculate the feature significance. Subsequently, a decision tree, random forest, and extra random forest are used to construct a multiclass malware classifier that illustrates the chosen feature's role in malware detection. According to the experimental part, a minimal number of features were used to identify the four different malware variants. The authors in (Romli et al., 2021) Present a novel framework for detecting Android malware. Three different methods are used for feature representation. The filter-based method is used to identify dominant features when designing the malware classifier. The tested results say that the random forest with 23 features produces better results than other methods. In most cases, malware threats target network traffic data, and features extracted from .pcap files have been used to design ML models. Therefore, researchers in (Nugraha et al., 2021) have proposed a lightweight machine learning-based malware classifier based on network traffic packet attributes. Extra Tree classifier (ETC) has been used to find the best features. Upon the features derived from ETC, RF based models show the best results than its counterpart models. As feature selection has been considered for dimensionality detection, the authors (Nugraha 2021) claimed that the proposal is suitable for any sector to detect the malicious applications.

A hybrid feature selector (Narayanan 2021) combines the best parts of XGBoost and the vote-based backward feature elimination method. A voting algorithm with a filter method selects features initially. The wrapper method then determines the optimal characteristics. Finally, selected features were used to enhance the performance of malware classifiers. As a result, the proposed framework obtains 99.5% classification accuracy, which is higher than the results obtained from conventional machine learning methods, demonstrating the practical relevance of their approach. The authors (Manzano et al., 2022) believe the classification efficacy relies on reduced features. The presented framework uses PCA (Principal component analysis) and Logistic Regression (LR) to select the network flow features. In addition, various machine-learning algorithms are used to experiment with both binary and multiclass malware classification models. The tested result concludes that the selected thirteen features significantly improve the malware traffic data.

2.2 Malware Classification Using Deep Learning

Another category of technique that excels in detecting malware is deep learning techniques. Since 2008, malware growth has rapidly increased, and almost all sectors have suffered due to its impact. Conventional machine learning (ML) techniques can no longer detect all dynamic and complex malware variants. Using a Deep Convolutional Network (DCNN), the method in (Kalash et al., 2018) figures out the hidden pattern to find the malware. The malware binary file is a grayscale image for the facilitated DCNN model to detect the malware precisely. A malware detection method that utilizes hybrid deep learning extracts information from a grayscale image is presented by (Aslan & Yilmaz 2021) and applies it to supervised deep learning models. Two neural networks that are operated and already trained are Alexnet and Resnet-152 Net. Here, the convolution layers of the hybrid model are considered in contrast to other approaches that use traditional feature extraction techniques. The testing indicated that the hybrid deep learning model improves obfuscation variation identification. However, it must still be shown to resist attacks on the carefully constructed features. With little training time, pre-trained models provide passably excellent results. Hence, the authors of (Pant et al., 2021) created a malware classifier using a customized CNN and demonstrated its superiority over the pre-trained modelsVGG-16, Inception-13, and other methods in terms of outcomes. Merits of CNN and LSTM were leveraged to discriminate against the malware family in (Dawra et al., 2023). Features extracted from PE files are converted into gray scale images to train the CNN+LSTM based model. Tested results reveal that CNN+LSTM is able to detect the malware more quickly than that of the other pre-trained models.

2.3 Malware Classification Using Federated Learning

Malware attacks on IoT devices have been on the rise, and detecting infectious IoT devices is a challenging task. Ensuring privacy and security for IoT devices is of utmost importance, especially in Industry 4.0 applications. As IoT devices have a centralized repository for data storage, they are vulnerable to security breaches. To address this issue, a federated learning-based framework for detecting malware that affects IoT devices has been demonstrated in (Rey et al., 2022). Network features from the N-BaIoT dataset were used to train supervised (MLP) and unsupervised (autoencoder) models. To assess the security of this approach, an adversarial setup was employed. The results reveal that the aggregation step used in federated learning is vulnerable to many cyberattacks.

Federated learning involves training on data that is stored on different machines, with the model results being aggregated through a centralized server. This method ensures data privacy and efficacy. Federated learning has been successfully employed to classify malware using data dispersed across different machines by (Lin et al., 2020). Malware behaviors are being used as features to train LSTM and SVM models in a federated learning approach. Results show that sharing models improves accuracy and reduces complexity. However, despite these advancements, federated learning-based methods remain vulnerable to adversarial attacks. Therefore, research is ongoing to improve FL-based malware detection and identify malicious attacks.

Federated learning can struggle when devices have different datasets (non-independent and identically distributed data). This can lead to issues like needing more training time and getting less accurate results. To alleviate this issue, one approach (D'Angelo et al., 2023) used Markov chains with associative rules to develop a customized FL architecture. The performance of this architecture was then compared with the results obtained through various machine learning models to demonstrate its superiority. Additionally, researchers (Venkatasubramanian et al., 2022) integrate static features and graph-based features derived from ELF binaries to train an FL-upon RF model to detect IoT malware. The tested results showed that the FL-based model achieved better malware detection accuracy compared to non-FL-based models. For better readability, Table 1 summarizes recent malware detection systems published in various venues, focusing on those designed for IoT networks and Android-enabled devices.

Table 1
Existing malware detection software using ML, DL and FL, comparative report

Ref	Input	Technique	Merits	Feature selection
(Mahajan et al., 2019)	Behavioral features	RF	Intend to detect Ransomwares	✓
(Babaagba et al., 2019)	Behavioral features	MLP	Collected features influences the unseen malware detection.	✓
(Al-Kasassbeh et al., 2020)	PE file features	J48	Novel feature selection	✓
(Kimet al., 2020)	Behavioral features	RF	Dimensionality reduction using RFE	✓
(Romli et al., 2021)	Behavioral features	RF	Privacy aspects are implemented for android enabled applications	✓
(Nugraha 2021)	Network traffic	RF	Presented Lightweight model is appropriate for any sector	✓
(Narayanan 2021)	PE file features	SVM,RF	Backward Feature Elimination technique	✓
(Manzano et al., 2022)	Flow and Network traffic	RF	Feature extraction (PCA) and feature selection (LR)	✓
(Kalash et al., 2018)	Malware binaries to grayscale image	DCNN	Automatic feature extraction	--
(Aslan & Yilmaz, 2021)	Malware binaries to grayscale image	DNN	Feature extraction done through Alex-net and Resnet-152	--
(Pantet al., 2021)	Malware binaries to grayscale image	DNN	Customized CNN	--
(Dawra et al., 2023)	Malware binaries to grayscale image	CNN-LSTM	Merits of CNN and LSTM influences the task	---
(Rey et al., 2022)	Network traffic	MLP,AE	IoT data privacy is ensured	--
(Lin et al., 2020)	Behavior based features	SVM, LSTM	Data privacy	---
(D'Angelo et al., 2023)	Behavior based features	FL	Suitable for edge and cloud	---
(Venkatasubramanian et al., 2022)	Static and graph based features	RF	Decentralized model	✓

Table 1 confirms that most recent malicious code detection methods have benefited significantly from ML. Among DL based models, Convolutional Neural Network (CNN) based models achieve consistent results without extensive feature engineering, whereas other ML models rely heavily on feature engineering for classification. Federated Learning (FL) aims to improve connected device security. References (Rey et al., 2022; Lin et al., 2020; D'Angelo et al., 2023; Venkatasubramanian et al., 2022) demonstrate various malware detection strategies using network flow and behavior-based features with ML and DL techniques. In contrast to the display of the datasets used in existing works, Table 1 highlights the types of features used for successful malware detection in IoT networks. Such analysis facilitates the researchers in understanding the merits of different types of features. Based on this analysis, the proposed solution will also leverage network traffic features. Another takeaway from the literature analysis is that Random Forest (RF)-enabled frameworks often perform the best.

3. Methodology

With the proliferation of devices and diverse communication protocols in the IoT network, it has become an attractive target for malicious applications and botnets. Unfortunately, most IoT devices compromise their security via botnet attacks. Hackers exploit botnets to execute DDoS and flooding attacks. The IoT architectures, comprising perception, network, and application layers with distinct functionalities, are all susceptible to various types of security breaches. The network layer, in particular, is the most vulnerable to cyber-attacks compared to other layers. In the literature, numerous researchers propose the use of machine learning techniques to detect or prevent software anomalies. Hence, a robust ensemble machine-learning-based

malware classifier is urgently needed. We have proposed such a classifier, leveraging finely crafted network data features through a novel hybrid feature selection technique.

Our proposed malware classification framework, a significant contribution to the field, is developed in two phases. The first phase utilizes the merits of statistical tools and OCS algorithms to select the unique features. Including all the collected features may not be the optimal approach to train the machine learning models. Recognizing that more components may escalate the model's complexity and some variables may be redundant, such variables may significantly impact the model's generalization capability. Subsequently, it would degrade the model's performance. In the second phase, a fine-tuned ensemble voting classifier is designed. This framework actively collects data from various sources during the data collection phase.

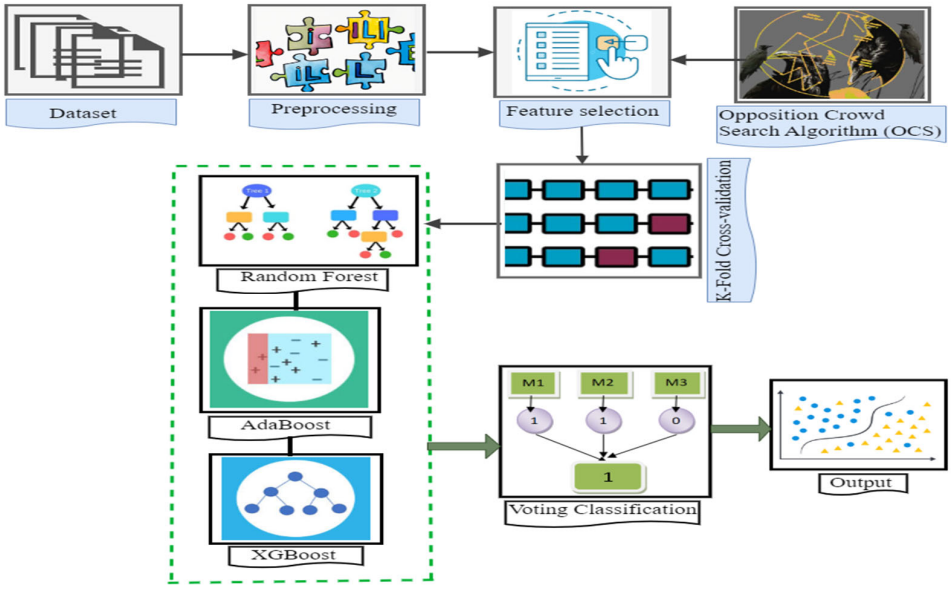

Fig. 1. Components and workflow of the lightweight malware ensemble classifier

4.1 Dataset and Pre-Processing

The tested (MTA-KDD'19) Malware Traffic Analysis Knowledge Dataset is a collection of real-world network traffic data specifically designed for training and testing malware detection systems. It incorporates data from the Malware Capture Facility Project (MCFP), which gathers both malicious and legitimate traffic. The MTA-KDD'19 dataset includes data captured from June 2013 to August 2019, totalling over 7 GB across 2112 files. Initially, the dataset contained 50 features (Hublikar & Shet 2022) describing various network traffic attributes. Researchers then refined the data through a multi-step process, removing redundant features, null values, and outliers. This resulted in a final version with 33 key features (Letteri et al., 2020) for malware detection. As mentioned in Fig. 2, the dataset consists of 30206 legitimate samples and 34350 malicious traffic samples This allows researchers to develop and evaluate malware classifiers effectively.

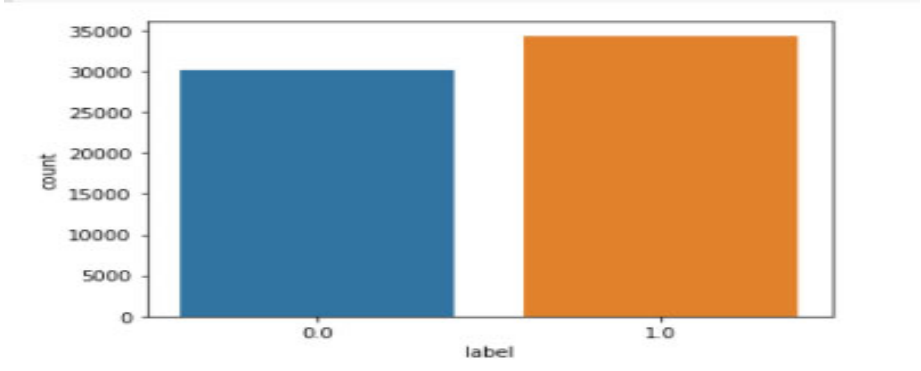

Fig. 2. Class distribution of the MTA-KDD-19 Dataset

4.2 Feature Selection through OCS

In the earlier phase of the feature selection, to eliminate the multicollinearity, the correlation coefficient tool has been used**.** The resultant uncorrelated features are furnished in Table 2. The Crow Search Algorithm (CSA) is inspired by crow behavior involving food anonymity and recovery. CSA characteristics include flocking behavior that mimics collective crow movement, the memory of hiding places that resemble crow caches, probabilistic following that reflects crows following each other, and cache defence that mirrors crows' defensive behavior. The Opposition-Based Crow Search Algorithm (OCS) enhances the performance of the traditional Crow Search (CS) method for the feature selection of the dataset. Introducing an opposite response to each original solution in the search field is the basic principle of OCS. By contrasting several solutions, the best option may be chosen, and this ultimately results in the identification of the ideal qualities.

Initialization: A population of crows is formed to represent features derived from the dataset. According to Eq. (1), these crows are dispersed randomly over the search space.

$$F_j = fS_1 + fS_2 + \cdots \ldots + fS_n \quad \text{where j} = 1,2, \ldots., \text{n} \tag{1}$$

Opposite Process: Each initial solution is followed by opposite solutions produced by OCS, which builds on the principles of meta-heuristic optimization. This procedure enables the search space to be explored outside of the local area of the first solutions, for instance $F \in (c, d)$ is a valid number, then the opposite point represented in Eq. (2).

$$\breve{F}_i = c_i + d_i - F_i \tag{2}$$

Evaluation of Fitness: An objective function unique to the research challenge is used to evaluate each solution's fitness. In this scenario, the goal is to improve the accuracy of feature selection from the provided dataset through Eq. (3).

$$Of_i = Max\ (A) \tag{3}$$

Position Generation: Using a random selection of another crow 'i' and its location into its movement equation, each crow changes its position as $Gp^{j,itr}$ via Equation 6. Exploration and exploitation inside the search space are made more accessible by this procedure. The equation considers a variety of elements, including random numbers and flying duration.

$$Gp^{j,itr+1} = \begin{cases} Gp^{j,itr} + R_j * Fl^{j,itr}, & if\ R_i > P^{j,itr} \\ Random\ position\ p\ of\ crow, & otherwise \end{cases} \tag{4}$$

where, R_j and R_i represent the random number of crow j and i within the range of [0-1], $Fl^{j,itr}$ is represents the crow fly length, $P^{j,itr}$ probability iteration of crow j. The algorithm ends when a predetermined stopping requirement is satisfied after iterating repeatedly. Finding a solution with a continuously higher fitness value or reaching a predetermined number of iterations might be the requirement for this criterion. Number of features has been fixed as a termination criterion for the candidate feature selection experiment. Therefore, the features that are used by the OCS algorithm are displayed in Table 2. Table 3 presents the varying number of dominant features chosen based on the merits of the OCS algorithm.

Table 2
List of uncorrelated features

Feature name	Feature name	Feature name	Feature name
FinFlagDist-F1	1stPktLen-F13	MinLen-F7	NumPorts-F19
RstFlagDist-F2	MaxLenrx-F14	MaxIAT-F8	FlowLENrx-F20
PshFlagDist-F3	MinLenrx-F15	MinIAT-F9	repeated_pkts_ratio-F21
DNSoverIP-F4	StdDevLenrx-F16	AvgIAT-F10	NumCon-F22
TCPoverIP-F5	AvgLenrx-F17	AvgWinFlow-F11	Start_flow-F23
MaxLen-F6	MinIATrx-F18	PktsIOratio-F12	HTTPpkts-F24

Table 3
List of reduced crafted feature columns through OCS

Ten Features	FinFlagDist,DNSoverIP,TCPoverIP',UDPoverIP,MaxIAT,AvgIATPktsIOratio,1stPktLen,StdDevLenrx,MinIATrx
Eight Features	FinFlagDist,DNSoverIP,TCPoverIP,UDPoverIP,MaxIAT,AvgIAT,1stPktLen,MinIATrx
Five features	FinFlagDist,TCPoverIP,MaxIAT,AvgIAT,MinIATrx

Features displayed in Table 3, are crucial in detecting the most probable cyberattacks, DDoS, flooding, and botnets in the context of the IoT. FinFlagDist PktsIOratio features help determine the malware traffic. Hence, such features are perfect for countering the DDoS attack. The 1stPktLen feature is good at detecting flooding attacks through the botnet. The features MaxIAT, AvgIAT, and MinIATrx are measures to define the inter-packet arrival time, which can facilitate detecting flooding attacks. Most malware attacks exploit the vulnerabilities of the communication protocol, So, the values scored against the variables 'DNSoverIP', 'TCPoverIP', and 'UDPoverIP' are very useful to find the malware traffic.

4.3 K-Fold Cross-Validation

Once the feature selection is done, the malware samples must be scaled to feed into the machine-learning models. The candidate experiment preferred to use a standard scalar. This section sketches the idea of k-fold cross-validation. CV (Cross-Validation) is a statistical tool to train machine learning models. One of the merits of k-fold cross-validation is minimizing the chance of models getting stuck into overfitting issues. Cross-validation is not only intended to prevent overfitting but also to enhance the reliability of the prediction. Since the model is trained with the data in a single shot, it may yield the best performance, which may not be reliable. To prevent such a situation, it is always good to use k-fold cross-validation. In k-fold cross-validation, all data samples are divided into k-folds. The model uses the k-1 folds of examples to train and the kth fold to test. Randomness applied in separating the data samples facilitates the model in ensuring generalization. Also,the model can infer insight into the data in a better manner. In the context of the imbalanced dataset, stratified k-fold cross-validation is the better choice to ensure the above-discussed merits. Python 3.4.6's API is used to train as well as to validate the ML models upon cross-validation, and a stratified K-fold function was applied to the training samples. The efficacies of the machine learning models are measured through the cross val_score function.

4.4 Ensemble Malware Voting Classifier

This section describes the design process for the ensemble voting classifier, which aims to improve malware detection performance. By combining diverse machine learning algorithms, this approach aims to achieve higher accuracy and reliability compared to traditional models. Each ML model contributes its prediction ("vote") to collaboratively strengthen the malware traffic classification. The three most effective models, based on their performance in classifying candidate malware samples, are chosen to form the ensemble. The presented ensemble voting classifier leverages the merits of RF, ADB and XGB methods to influence the malware traffic classification. Ensemble voting approach assigns one vote to each ML model's prediction for a specific output class_label. The class_label with the highest number of votes wins. The average probability scores assigned by each ML model for each class have been computed using Eq. (5). The probability scores for a specific class are summed across all the ML models and then averaged. The class with the highest average probability wins.

$$P(y) = \frac{1}{N}\sum_{n=1}^{N} P_n({y_i}^n), \quad Where\ y \in C \tag{5}$$

where, N is number of ML models and C is number of output classes, $P(y)$ is probability of final predicted sample belongs to class C. ${P_n({y_i}^n)}$ is probability assigned by ML model n to sample i belongs to class C

4.4.1 Best Tree-Based Classifier

Random Forest works on a large, uncorrelated decision tree. Samples are collected randomly to create a set of decision trees. At every node, the method finds the best split on the selected samples, which is done by either using the Gini index or entropy. Eq. (6) and Eq. (7) give the Gini and Entropy indices, respectively. The 'p_i' represents the relative frequency of a class from the training set, and the 'C' denotes the number of output classes. Eq. (7) represents the entropy used to decide the branch. Then, decision trees allowed to grow until they reach maximum depth. The votes from different decision trees are summed in the next step to determine the final class label.

$$\text{Gini} = 1 - \sum_{\text{i=i}}^{\text{C}} (\text{p}_\text{i}^2) \tag{6}$$

$$\text{Entropy} = \sum_{\text{i=1}}^{\text{C}} -\text{p}_\text{i} * \log_2(\text{p}_\text{i}) \tag{7}$$

AdaBoost is a machine-learning technique that uses ensemble learning to improve classification performance. Combining several low-performing classifiers into one robust classifier is the logic of AdaBoost. Such action facilitates improvising classifier performance. AdaBoost dynamically adjusts weights for both individual classifiers and training samples. It continually improves its focus on hard-to-classify examples. AdaBoost randomizes the training set and iteratively chooses the training samples that could yield the best prediction in the previous training session to train the current iteration. The misclassified samples are assigned a higher weight, so the probability of including them in the training samples will be higher. One machine learning technique needs to be designated as a base estimator. A decision tree will be the default base model if nothing is specified. The XGBoost(XGB) algorithm uses the gradient-boosting decision tree algorithm. Like the Decision Tree, each node represents a feature condition, the branch denotes the decision on the state, and the leaf denotes the class label; these details are combined for the final prediction. XGB is a fine-tuned gradient-tree boosting method built upon decision trees. XGB is widely known for its speedy performance in designing models. XGB represents the possible solution through a graphical representation like the decision tree. Each node represents the condition of the n feature, the branch denotes the state, and the leaf indicates the class label. Then, the ensemble meta-algorithm item combines predicted decisions through various

decision trees based on bagging. Such a bagging method further developed a forest, a collection of decision trees built upon randomly chosen features. The gradient boosting method creates new models for predicted errors and the prior model's residuals. The XGB algorithm removes the missing values and addresses the overfitting issues using parallel processing. XGB is built upon tree construction and tree pruning processes. The algorithm supports three gradient boosting forms: gradient boosting machines, stochastic gradient boosting, and regularized gradient boosting. While designing the classifiers, issues incurred through high variance can be mitigated through RF model. AdaBoost enables the weak learners to work consistently on the tricky samples to derive better insight. XGBoost's merits, such as sparsity, distributed learning, and early stopping, make it a perfect choice for classification tasks. The candidate ensemble voting classifier has been constructed using Random Forest, ADB, and XGB after their merits have been thoroughly examined.

5. Experimental Results and Discussion

This section portrays the efficacy of the presented work with the benchmarking methods in terms of the various quantitative metrics mentioned in Section 4.

5.1 Performance Indicators

To measure the performance of the candidate proposal, the performance metrics, accuracy, Precision, Recall and then F1-score are preferred. The metric computation is a combination of true positive (T^+), true negative (T^-), and false positive/negative (F^+/F^-). T^+represents the number of malware samples that are accurately identified. F^+ represents the count incorrectly classified. On the other hand, F^- represents the count of samples that have been incorrectly labeled as benign, instead of malign. Considered metrics expressions are represented in Eqs. (8-11).

$$A = \frac{T^+ + T^-}{T^+ + T^- + F^+ + F^-} \tag{8}$$

$$P = \frac{T^+}{T^+ + F^+} \tag{9}$$

$$R = \frac{T^+}{T^+ + F^-} \tag{10}$$

$$F1Score = 2\frac{P.R}{P + R} \tag{11}$$

The framework uses binary classification, so the metrics TPR, FPR and AUC are considered performance measures. The mathematical expression of TPR (True Positive Rate), and FPR (False Positive Rate) is represented in Eq. (15) and Eq. (16).

$$TPR = \frac{T^+}{T^+ + F^-} \tag{12}$$

$$FPR = \frac{F^+}{F^+ + T^-} \tag{13}$$

An ROC curve shows how well a model correctly identifies positive cases (true positives) against mistakenly classifying negative cases as positive (false positives). Ideally, a model should have a low false positive rate (FPR) while achieving a high true positive rate (TPR). AUC, or Area Under the Curve, summarizes this performance by measuring the total area under the ROC curve.

5.2 Discussion on Results

The proposed malware classifier is implemented in Python 3.7.10 using an Intel i5 machine with 12 GB of RAM on the MTA-KDD19 dataset. Keras API with a TensorFlow backend simulates the machine learning models. Table 4 presents the diverse ML classifier's performance, as evaluated by averaging the results of 10-fold and 5-fold cross-validation. At 24 features, the ADB and model results in terms of precision are higher than that of others. In terms of accuracy RF, ADB , XGB and presented Voting classifier results are comparable. At ten number of features, RF based model's results better than the voting classifier. To portray the performance of various ML models alongside the presented ensemble voting model, Tables 5 and 6 summarize the results obtained using eight and five features, respectively. Based on the results in Table 5, the ensemble voting classifier achieves superior performance across all confusion matrix metrics compared to the other models. Following the ensemble classifier, the Random Forest (RF) model performs better than the others. Graphical representation of the tested results furnished in Table 5 is shown in Fig. 3

Table 4
Comparative results of malware classification framework at varying number of features

Features	Method	A	P	R	F1-score	AUC
24	RF	0.992	0.990	0.998	0.995	0.999
	ADB	0.992	0.987	0.999	0.993	0.991
	KNN	0.988	0.988	0.989	0.988	0.996
	XGB	0.992	0.987	0.999	0.993	0.989
	Voting	0.992	0.991	0.991	0.991	0.991
20	RF	0.988	0.994	0.985	0.989	0.999
	ADB	0.981	0.986	0.977	0.981	0.998
	KNN	0.967	0.985	0.953	0.968	0.985
	XGB	0.981	0.989	0.978	0.983	0.998
	Voting	0.98	0.99	0.98	0.98	0.99
10	RF	0.966	0.974	0.962	0.967	0.994
	ADB	0.933	0.950	0.923	0.935	0.983
	KNN	0.936	0.957	0.921	0.937	0.972
	XGB	0.943	0.954	0.938	0.945	0.989
	Voting	0.961	0.970	0.955	0.962	0.992

Table 5
Results of averaging 10-fold and 5-fold cross-validation for diverse ML models using 8 features.

	A	P	R	F1-Score
LR	0.77	0.79	0.78	0.78
GNB	0.72	0.95	0.5	0.64
KNN	0.92	0.94	0.91	0.92
RF	0.94	0.95	0.94	0.94
SVM	0.77	0.83	0.71	0.75
ADB	0.92	0.94	0.91	0.92
XGB	0.93	0.94	0.93	0.93
Voting	0.95	0.959	0.95	0.95

Again, Table 6 summarizes the results of ML models using five numbers of features. The experiment's results, upon averaging the 10-fold and 5-fold cross-validation, reveal that the ensemble voting model performs best. Following the ensemble model, XGBoost (XGB) achieves better results than the other models considered in the analysis.

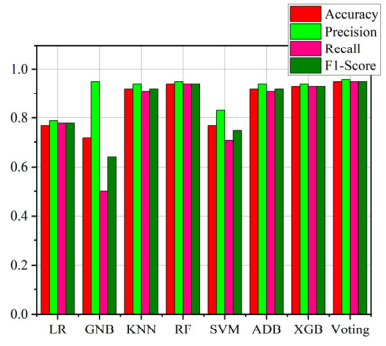

Fig. 3. Tested results comparison in terms of numeric metrics

Table 6
Results of averaging 10-fold and 5-fold cross-validation for diverse ML models using 5 features

Method	A	P	R	F1-Score
LR	0.78	0.81	0.78	0.79
GNB	0.75	0.88	0.61	0.71
KNN	0.82	0.85	0.82	0.83
RF	0.86	0.86	0.86	0.86
SVM	0.77	0.84	0.69	0.75
ADB	0.84	0.86	0.85	0.85
XGB	0.86	0.88	0.86	0.87
Voting	0.87	0.89	0.88	0.88

To illustrate the ML models' learning ability with varying sample sizes, Fig. 4 and Fig. 5 depict the evolution of training and cross-validation scores obtained by various ML models. The reason for including the snapshots in Figure 4 and Figure 5 is to reveal the ML model's generalizability. Training score indicates the ML models performance on training data, whereas the cross-validation score represents the efficacy of the ML models upon the unseen data. Figure 4 shows various ML models performance on both training and unseen samples upon 10-fold cross-validation. Deviation between the training score curve with the cross-validation score is expected to be minimal. Initially, when the models begin to learn, there is a high discrepancy between the training and validation scores. This is natural. As the models are exposed to more samples, the gap between the training and validation scores starts to shrink. While Fig. 4a reveals a high initial discrepancy between the GNB model's training and validation scores, these curves converge. This suggests the model achieves good generalizability despite the initial difference. However, evaluating the secured value against the specific metric is important to determine its effectiveness. ADA Boost and XGBoost have repeated the same scenario, but the accuracy obtained is far better than that produced by the GNB. Right from the beginning, the voting classifier exhibits a higher accuracy score than other models. However, it struggles to minimize the training and validation scores gap. Interestingly, the voting classifier's initial validation score surpasses that of the other models, but it seems unable to improve this difference further.

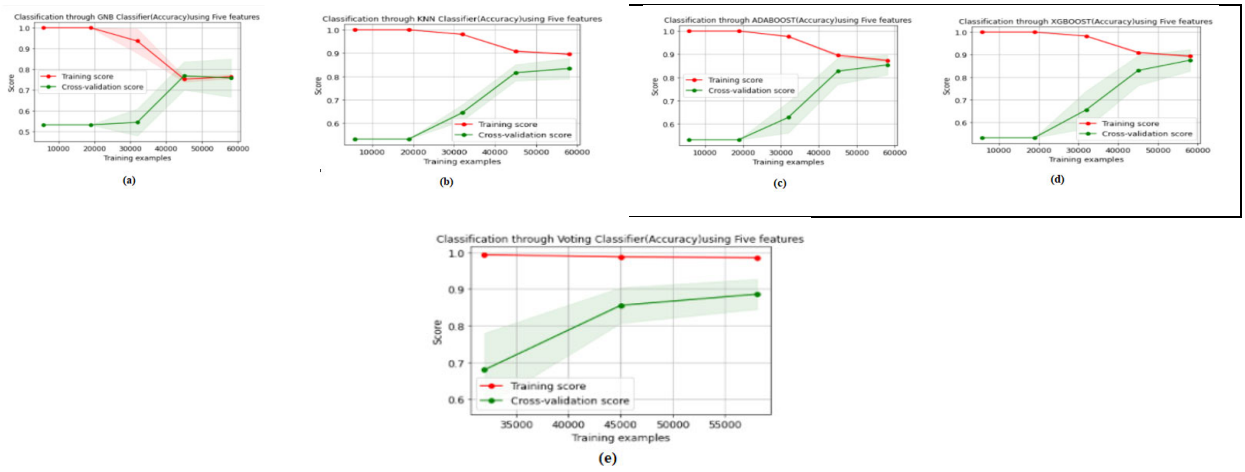

Fig. 4. Training versus (10-fold) validation score curves produced by various ML models.

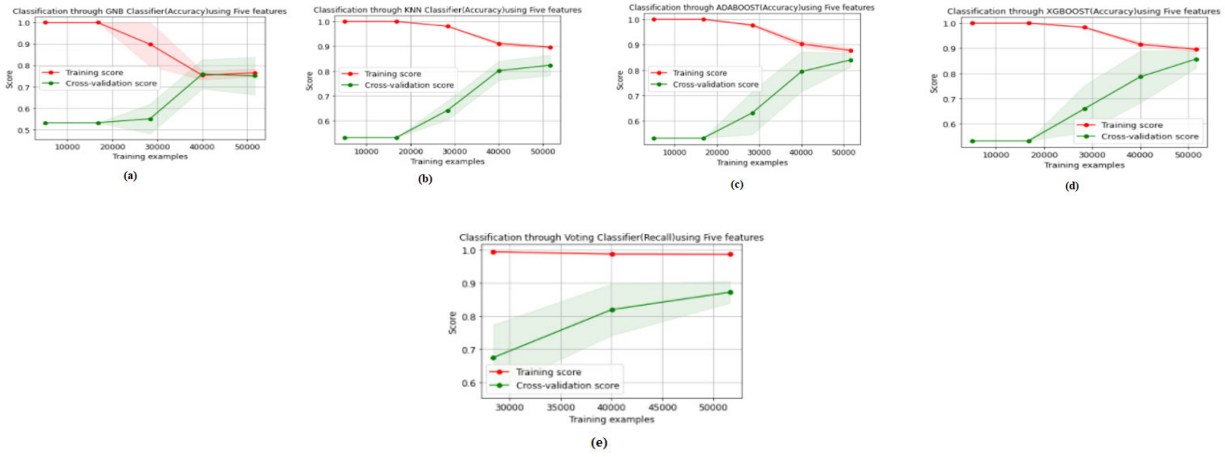

Fig. 5. Training versus (5-Fold) validation score curves produced by various ML models

Fig. 5 depicts the performance of various machine learning models on training and validation sets. However, this experiment utilized 5-fold cross-validation. Interestingly, no significant difference was observed when comparing the effectiveness of the models using 10-fold and 5-fold cross-validation. One possible explanation for the models' inability to reduce the gap between training and validation scores could be the limited number of features considered (five features) in Fig. 4 and Fig. 5. The positive aspect of this candidate experiment is that the value of numeric metrics is appreciable in the presence of five features.

6. Conclusion

In Industry 4.0 applications, the Internet of Things (IoT) enables smooth automation. However, securing IoT data from intruders is still a challenging task. Among the diverse cyber-attacks, malware poses significant threats to IoT networks. Although several research attempts have been put forward in the literature on ML and DL models, fewer studies have focused on creating a malware classifier based on attributes that may ease malware traffic identification. This study proposed a lightweight ensemble voting-based malware classification system that uses OCS for feature selection. To make the malware classifier suitable for the IoT environment with minimal resource constraints, the proposed work utilizes a minimal feature set from the MTA-KDD'19 dataset. The tested findings showcase the merits of the proposed malware traffic classifier in terms of accuracy. Therefore, the malware traffic classifier is a viable solution for large-scale IoT settings. Here are some additional steps that could be considered for future development. Experimenting with different feature selection algorithms could identify optimal features and mitigate overfitting. Data sampling techniques, especially for minority classes, could improve the model's performance, particularly when considering deep learning approaches. Investigating the potential of deep learning models could lead to further improvements in accuracy and malware detection capabilities.

Acknowledgments

The authors thank all the respondents from the leather industry who provided valuable responses and support for the survey. They offer special gratitude to INTI International for publishing the research work and, in particular, to INTI International University for funding the publication of this research work.

Funding

The authors offer special gratitude to INTI International University for the opportunity to conduct research and publish the research work. In particular, the authors would like to thank INTI International University for funding the publication of this research work. Also, we extend our heartfelt gratitude to all research participants for their valuable contributions, which have been integral to the success of this study.

Data Availability Statement

Data will be made available upon reasonable request.

References

Abusitta, A., Li, M. Q., & Fung, B. C. (2021). Malware classification and composition analysis: A survey of recent developments. *Journal of Information Security and Applications, 59*, 102828.

Al-Kasassbeh, M., Mohammed, S., Alauthman, M., & Almomani, A. (2020). Feature selection using machine learning to classify malware. In Handbook of computer networks and cyber security (pp. 889-904). Springer, Cham.

Almin, S. B., & Chatterjee, M. (2015). A novel approach to detect android malware. *Procedia Computer Science, 45*, 407–417.

Aslan, Ö., & Yilmaz, A. A. (2021). A new malware classification framework based on deep learning algorithms. *IEEE Access, 9*, 87936-87951.

Azad, M. A., Riaz, F., Aftab, A., Rizvi, S. K. J., Arshad, J., & Atlam, H. F. (2022). DEEPSEL: A novel feature selection for early identification of malware in mobile applications. *Future Generation Computer Systems*, *129*, 54-63.

Babaagba, K. O., & Adesanya, S. O. (2019, March). A study on the effect of feature selection on malware analysis using machine learning. *Proceedings of the 2019 8th international conference on educational and information technology* (pp. 51-55).

Baptista, I., Shiaeles, S., & Kolokotronis, N. (2019, May). A novel malware detection system based on machine learning and binary visualisation. *In 2019 IEEE International Conference on Communications Workshops (ICC Workshops)* (pp. 1-6). IEEE.

Bendiab, G., Shiaeles, S., Alruban, A., & Kolokotronis, N. (2020, June). IoT malware network traffic classification using visual representation and deep learning. *In 2020 6th IEEE Conference on Network Softwarization (NetSoft)* (pp. 444-449). IEEE.

Canfora, G., Medvet, E., Mercaldo, F., & Visaggio, C. A. (2015). Detecting android malware using sequences of system calls. *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile,* 13–20.

Chakravarthy, S. J. (2021). Wrapper-based metaheuristic optimization algorithms for android malware detection: a correlative analysis of firefly, bat & whale optimization. *J Hunan Univ, 48*(10).

D'Angelo, G., Farsimadan, E., Ficco, M., Palmieri, F., & Robustelli, A. (2023). Privacy-preserving malware detection in Android-based IoT devices through federated Markov chains. *Future Generation Computer Systems*, *148*, 93-105.

Dawra, B., Chauhan, A. N., Rani, R., Dev, A., Bansal, P., & Sharma, A. (2023, February). Malware Classification using Deep Learning Techniques. In *2023 2nd Edition of IEEE Delhi Section Flagship Conference (DELCON)* (pp. 1-7). IEEE.

Fang, Z., Wang, J., Geng, J., & Kan, X. (2019). Feature selection for malware detection based on reinforcement learning. *IEEE Access, 7*, 176177-176187.

Huang, W., & Stokes, J. W. (2016). MtNet: a multi-task neural network for dynamic malware classification. In Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings 13 (pp. 399-418). Springer International Publishing.

Hublikar, S., & Shet, N. S. V. (2022). Hybrid Malicious Encrypted Network Traffic Flow Detection Model. In Computer Networks and Inventive Communication Technologies: Proceedings of Fifth ICCNCT 2022 (pp. 357-375). Singapore: Springer Nature Singapore.

Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D., Wang, Y., & Iqbal, F. (2018, February). Malware classification with deep convolutional neural networks. In 2018 9th IFIP international conference on new technologies, mobility and security (NTMS) (pp. 1-5). IEEE

Kavitha, P. M., & Muruganantham, B. (2021). An Extensive Review on Malware Classification Based on Classifiers. *Intelligent Computing and Innovation on Data Science*, 371-381.

Kim, C., Chang, S. Y., Kim, J., Lee, D., & Kim, J. (2023). Automated, reliable zero-day malware detection based on auto-encoding architecture. IEEE Transactions on Network and Service Management.

Kim, D. W., Shin, G. Y., & Han, M. M. (2020). Analysis of feature importance and interpretation for malware classification. Computers, *Materials & Continua, 65*(3), 1891-1904.

Letteri, I., Di Cecco, A., & Della Penna, G. (2020). Dataset Optimization Strategies for Malware Traffic Detection. arXiv preprint arXiv:2009.11347

Lin, K. Y., & Huang, W. R. (2020, February). Using federated learning on malware classification. In 2020 22nd International Conference on Advanced Communication Technology (ICACT) (pp. 585-589). IEEE.

Mahajan, G., Saini, B., & Anand, S. (2019, February). Malware classification using machine learning algorithms and tools. In 2019 Second international conference on advanced computational and communication paradigms (ICACCP) (pp. 1-8). IEEE

Mangayarkarasi, R., Vanmathi, C., & Ravi, V. (2023). A robust malware traffic classifier to combat security breaches in industry 4.0 applications. *Concurrency and Computation: Practice and Experience*, *35*(23), e7772.

Manzano, C., Meneses, C., Leger, P., & Fukuda, H. (2022). An Empirical Evaluation of Supervised Learning Methods for Network Malware Identification Based on Feature Selection. Complexity, 2022.

Marín, G., Caasas, P., & Capdehourat, G. (2021). *DeepMAL-deep learning models for malware traffic detection and classification. In Data Science–Analytics and Applications* (pp. 105-112). Springer Vieweg, Wiesbaden.

Mohy-eddine, M., Guezzaz, A., Benkirane, S., & Azrour, M. (2023). An effective intrusion detection approach based on ensemble learning for IIoT edge computing. *Journal of Computer Virology and Hacking Techniques, 19*(4), 469-481.

Narayanan, M. E. (2021). Malware Classification Using Xgboost With Vote Based Backward Feature Elimination Technique. *Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12*(10), 5915-5923.

Nugraha, U. (2021). Malware Classification Using Machine Learning Algorithm. *Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12*(8), 1834-1844

Pant, D., & Bista, R. (2021, November). Image-based Malware Classification using Deep Convolutional Neural Network and Transfer Learning. *In 2021 3rd International Conference on Advanced Information Science and System (AISS 2021)* (pp. 1-6).

Pascanu, R., Stokes, J. W., Sanossian, H., Marinescu, M., & Thomas, A. (2015, April). Malware classification with recurrent networks. *In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1916-1920). IEEE.

Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., & Nicholas, C. K. (2018, June). Malware detection by eating a whole exe. In Workshops at the thirty-second AAAI conference on artificial intelligence.

Raff, E., Zak, R., Cox, R., Sylvester, J., Yacci, P., Ward, R., ... & Nicholas, C. (2018). An investigation of byte n-gram features for malware classification. *Journal of Computer Virology and Hacking Techniques, 14,* 1-20.

Ramaiah, M., Chandrasekaran, V., Ravi, V., & Kumar, N. (2021). An intrusion detection system using optimized deep neural network architecture. *Transactions on Emerging Telecommunications Technologies*, *32*(4), e4221.

Rao, V., & Hande, K. (2017). A comparative study of static, dynamic and hybrid analysis techniques for android malware detection. *International Journal of Engineering Development and Research*, *5*(2), 1433-1436.

Rey, V., Sánchez, P. M. S., Celdrán, A. H., & Bovet, G. (2022). Federated learning for malware detection in IoT devices. *Computer Networks*, *204*, 108693.

Romli, R. N., Zolkipli, M. F., & Osman, M. Z. (2021, June). Efficient feature selection analysis for accuracy malware classification. In Journal of Physics: Conference Series (Vol. 1918, No. 4, p. 042140). IOP Publishing.

Saxe, J., & Berlin, K. (2015, October). Deep neural network based malware detection using two dimensional binary program features. In 2015 10th international conference on malicious and unwanted software (MALWARE) (pp. 11-20). IEEE.

Shibahara, T., Yagi, T., Akiyama, M., Chiba, D., & Yada, T. (2016, December). Efficient dynamic malware analysis based on network behavior using deep learning. *In 2016 IEEE Global Communications Conference (GLOBECOM)* (pp. 1-7). IEEE.

Venkatasubramanian, M., Habibi Lashkari, A., & Hakak, S. (2022, December). Federated Learning Assisted IoT Malware Detection Using Static Analysis. In *Proceedings of the 2022 12th International Conference on Communication and Network Security* (pp. 191-198).

Vishnukumar, R., & Ramaiah, M. (2024). Optimized deep learning-based intrusion detection framework for vehicular network. *Journal of Intelligent & Fuzzy Systems*, (Preprint), 1-18.

Yoo, S., Kim, S., Kim, S., & Kang, B. B. (2021). AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification. *Information Sciences, 546*, 420-435.

Yuan, Z., Lu, Y., Wang, Z., & Xue, Y. (2014, August). Droid-sec: deep learning in android malware detection. *In Proceedings of the 2014 ACM conference on SIGCOMM* (pp. 371-372).