A Multikernel and Metaheuristic Feature Selection Approach for IoT Malware Threat Hunting in the Edge Layer

Hamed Haddadpajouh, *Member, IEEE*, Alireza Mohtadi, *Member, IEEE*, Ali Dehghantanaha, *Senior Member, IEEE*, Hadis Karimipour, *Senior Member, IEEE*, Xiaodong Lin, *Fellow, IEEE*, and Kim-Kwang Raymond Choo, *Senior Member, IEEE*

***Abstract*—Internet-of-Things (IoT) devices are increasingly targeted, partly due to their presence in a broad range of applications (including home and corporate environments). In this article, we propose a multikernel support vector machine (SVM) for IoT cloud-edge gateway malware hunting, using the gray wolves optimization (GWO) technique. This metaheuristic approach is used for optimum selection of features distinguishing between malicious and benign applications at the IoT cloud-edge gateway. The model is trained with the Opcode and Bytecode of IoT malware samples (i.e., the training data set comprises 271 benign and 281 malicious Cortex A9 samples) and evaluated using the K-fold cross-validation technique. We validate the robustness of the proposed model, in terms of its ability to detect previously unseen IoT malware samples. We achieve an accuracy of 99.72% on the combination of the radial basis function (RBF) and polynomial kernels. Moreover, our proposed model only requires 20 s for training in comparison to the previous deep neural network (DNN) model that requires over 80 s to be trained on the same data. Overall, the proposed multikernel SVM approach outperforms DNNs and fuzzy-based IoT malware hunting techniques, in terms of accuracy, while significantly reducing the computational cost and the training time.**

***Index Terms*—Gray wolves optimization (GWO), Internet of Things (IoT), machine learning (ML) malware hunting, multikernel learning.**

I. Introduction

INTERNET-OF-THINGS (IoT) devices are found in various settings (e.g., homes and smart cities), and will be increasingly common in the foreseeable future. For example, Cisco estimated IoT-related device sales revenue would reach $14.4 trillion by 2022 [1]. Data collected by IoT devices can be shared directly or via application programming interfaces (API), to facilitate pattern collection, behavior observation, attack prediction, quality assessment, and other decision making and policy making [2], [3].

Hence, it is not surprising that attackers are seeking to exploit vulnerabilities in IoT devices (hardware and software) and their implementations. Such vulnerabilities can arise due to inherent computational limitations, insecure network protocols, and even using default credentials [4], [5]. An investigation of ten popular IoT devices, for example, revealed a number of security issues, such as open telnet ports, outdated Linux firmware, and unencrypted transmission of sensitive data [6]. An increasing number of IoT malware have also been reported in recent years that targeted vulnerable IoT devices [7], [8]. For example, it was estimated that the Mirai malware IoT botnet caused more than $4207.03 damage per hour of operation in 2017 [9], [10].

Malware threat hunting refers to the identification of malicious application(s) or malware within a network environment normally prior to their execution. Most malware threat hunting systems use malware analysis techniques to identify malicious and benign applications [11]. Malware analysis approaches can be broadly categorized into those based on static analysis and those based on dynamic analysis [12]. In static analysis, malicious applications are examined without execution using static features (e.g., Bytecodes, control flow graph, Opcodes, strings, and API calls) [13]. In contrast, in dynamic analysis, the malware sample is executed in a controlled environment to observe its behavior. Generally, IoT malware threat hunting uses static analysis. Most IoT communications share similar patterns, and hence a combination of statistical pattern recognition and machine learning (ML) techniques may offer a good performance in hunting IoT malware [6]. ML anomaly detection models can be applied directly to the network communication of IoT devices as well as malware threat hunting. However, a large number of features in ML algorithms may result in overfitting during learning time, higher computational resources consumption, and lower detection accuracy [14]. This phenomenon is known as the high dimensional problem, and can be addressed using feature selection and feature

Manuscript received July 1, 2020; revised September 3, 2020; accepted September 22, 2020. Date of publication September 25, 2020; date of current version March 5, 2021. The work of Kim-Kwang Raymond Choo was supported in part by the National Science Foundation (NSF) under Award 1925723, and in part by NSF CREST under Grant HRD-1736209. *(Corresponding author: Kim-Kwang Raymond Choo.)*

Hamed Haddadpajouh and Ali Dehghantanaha are with Cyber Science Lab, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: hamed@cybersciencelab.org; ali@cybersciencelab.org).

Alireza Mohtadi is with the Department of Software and IT Engineering, École de technologie supérieure, Montreal, QC H3C 1K3, Canada (e-mail: alireza.mohtadi.1@ens.etsmtl.ca).

Hadis Karimipour is with the College of Engineering, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: hkarimi@uoguelph.ca).

Xiaodong Lin is with the School of Computer Science, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: xlin08@uoguelph.ca).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, the Department of Electrical and Computer Engineering, and the Department of Computer Science, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/JIOT.2020.3026660

extraction algorithms [15]. Feature selection removes redundant or irrelevant features to reduce the size of the data, while feature extraction transforms a high dimensional data set into a new space with fewer dimensions [16]. Many feature selection techniques (e.g., wrappers, filters, and embedder) have been proposed in the literature. Metaheuristic search algorithms like particle swarm optimization (PSO) and gray wolves optimization (GWO) are examples of wrappers that are widely used for feature selection [15], [17]. These approaches attempt to identify the optimum feature set to improve ML algorithms' accuracy, and they are less prone to reach the local optima of the features.

There have also been attempts to perform malware threat hunting in IoT networks at the edge layer, with the aim of improving efficiency and minimizing latency. However, edge layer resources are much more constrained in comparison to conventional server farms. Resource limitation and diversity of IoT nodes compound the challenge of threat hunting at the edge layer. For example, to create a sandbox to run a malware, to collect dynamic malware data, and then analyze these sample would be resource demanding, and this would create a bottleneck situation at the edge layer with current technologies. Moreover, most IoT devices are running in real time and hence, threat identification should also be performed in near real time. Therefore, we posit the potential of static analysis for threat hunting at the edge-layer in a typical IoT environment.

In this article, we propose an ML-based threat hunting model utilizing a multikernel support vector machine (SVM) as a classifier and GWO module for feature selection. The novelty of the proposed model lies in the unique combination of SVM classifier's kernels for static malware threat analysis based on Opcode and Bytecode. Patterns in the static analysis (e.g., string signature, byte-sequence n-grams, syntactic library call, control flow graph, Opcode frequency distribution, and syntactic library call) are used for threat hunting. A summary of the contributions in this article is as follows.

1) We propose a malware threat hunting mechanism for cloud-edge gateways in an IoT environment, based on the multikernel SVM approach (in order to obtain high detection accuracy and F1 score). Also, we demonstrate that the proposed model outperforms previous malware threat hunting models based on fuzzy pattern tree and maximal-frequent pattern deep recurrent neural network (RNN).
2) We minimize the computational costs by using a metaheuristic feature selection algorithm to extract the optimum feature set from static IoT applications properties (i.e., Opcode and Bytecode). The proposed optimum feature selection algorithm significantly reduces both training and testing times, in comparison to prior deep RNN malware threat hunting models. In other words, our system is more practical for malware threat hunting in the edge layer of IoT networks.

The remainder of the article is organized as follows. Section II reviews the related literature. The proposed multikernel treat hunting model is explained in Section III. Section IV presents the experimental results, followed by the conclusion in Section V.

II. Related Work

Kang *et al.* [18] proposed a DL model based on RNN using applications' Opcode for malware threat hunting. The authors used word2vec long short-term memory (LSTM) architecture for detecting malware, which resulted in the detection accuracy of 97.59%. In addition, the proposed model has a high computational complexity as using the backpropagation approach consumes more computational resources.

Cakir and Dogdu [19] presented an ML model that was trained on the microsoft malware classification challenge data set Opcodes for training the model. The authors used the Word2Vec Opcodes operand to create feature vectors based on a gradient boosting machine (GBM). The proposed model resulted in the 95% detection accuracy using the five-fold cross-validation.

Yuxin and Siyi [20] developed a deep belief network (DBN) model to detect malware. The authors used Opcode property and employed DBN as an autoencoder to reduce the number of features. Their proposed feature selection module maintained the same performance (detection accuracy) compared to the existing works that use the full feature set.

Santos *et al.* [21] proposed a model for hunting unknown malware which utilized opcode sequences. The authors used weighted term frequency (WTF) for each Opcodes before applying any classification technique. In addition, their model leveraged from frequency-based feature selection module for employing most effective features. In the result, the proposed model obtained 96% detection accuracy against unknown malware.

Darabian *et al.* [22] used the frequent pattern of Opcode files in IoT devices for malware threat detection. The authors obtained the accuracy and F-measure of almost 99% with different classification modules like SVM, KNN, MLP, random forest (RF), DT, and AdaBoost. Moreover, Haddadpajouh *et al.* [23] used Opcodes as features set on a deep RNN model to train a three-layer LSTM network. The proposed LSTM model achieved a detection accuracy of 98.18% and 94% by 10-fold cross-validation for seen and unseen malware, respectively.

Opcode sequences have also received increasing attention in IoT malware detection because of the promising results based on Opcodes features. Azmoodeh *et al.* [24] developed a DL-based model to detect Internet-of-Battlefield-Things (IoBT) malware via the device's Opcode sequence and they achieved 99.68% of accuracy. Dovom *et al.* [3] proposed a fuzzy pattern tree ML model for malware threat hunting. The authors applied Opcode on their proposed model and obtained 99.17% detection accuracy. Although their proposed model could obtain a high detection accuracy but the model has a complex structure.

Overall, previous models showed the suitability of using static properties of executable samples for malware detection in IoT devices. Although previous works achieved a relatively high accuracy in detecting IoT malware samples, they are highly complex and incur significant computational and storage costs.

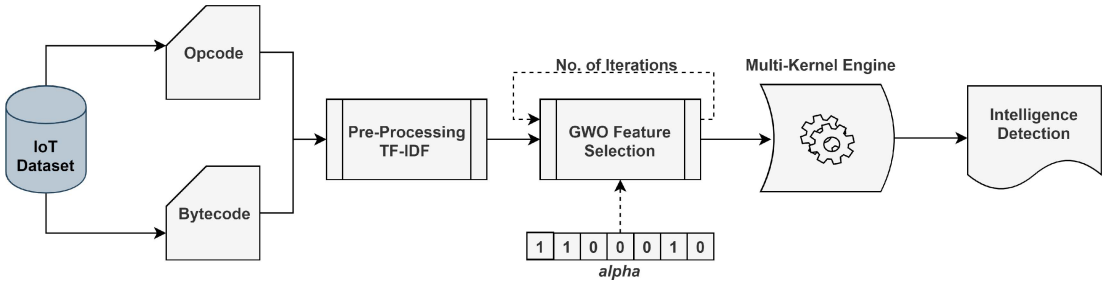


Fig. 1. Proposed model based on an SVM multikernel engine

III. Our Proposed ML-Based IoT Malware Threat Hunting Model

This section presents the proposed ML-based IoT malware threat hunting model, where we use a multikernel SVM classifier on an optimum feature set to increase the detection accuracy while seeking to minimize the computational cost. Optimum features are selected using the GWO algorithm. As shown in Fig. 1, our proposed model consists of three modules, namely, preprocessing, feature selection, and multikernel threat hunting model (see Sections III-A–III-C).

A. Preprocessing

We used our previously collected real-world IoT malware data set [23] to evaluate our proposed model. The data set consists of `Cortex A9` cloud-edge gateway application with training and test sets of 548 and 100 samples, respectively. Due to a lack of diversity in the cloud-edge device's applications, unlike the conventional malware data set, the number of samples in the data sets is limited. To the best of our knowledge, this is the only publicly available IoT malware data set collected at the edge layer. The data set offers two static properties for each sample, namely, Opcode and Bytecode.

1) *Opcode:* Opcode refers to Operational Code, which is the instruction interpreted at the hardware layer [25]. For example, ×86 OpCodes refer to microinstructions that are understandable by Intel ×86 processors. Our proposed model is optimized to work with Cortex A9 Opcodes, which are the most widely used opcode instruction set in IoT devices [23].
2) *Bytecode:* Bytecode refers to a program code that is compiled from the source code into a low-level code designed for efficient execution by software interpreters, such as java virtual machine [26].

Each property includes sequences of textual data, such as sequences of operands `pop, push, mov, mul, mov,..., ret`. Each data set is vectorized before being fed to our proposed model.

Since Bytecode and Opcode data are sequences of words and numbers, text mining techniques are suitable for preprocessing activities. Therefore, a word dictionary is generated from both Opcode and Bytecode for preprocessing the data sets. `TF-IDF` is a common metric for indicating how prominent a word in one document as shown in [27]. The preprocessing module computes the occurrences of each specific word in every Opcode and Bytecode file analyzed. Every word in the dictionary is counted as a feature that can affect malware detection. In the proposed model, all Opcode and Bytecode is transferred into a `TF-IDF` value to keep the information about each token (Opcode, Bytecode) in each malware/benign sample as follows:

$$\mathrm{TF}(s, O) = \log\left(1 + f_{(o,s)}\right) \tag{1}$$

$$\mathrm{IDF}(o, D) = \log \frac{N}{|s \in D : o \in S|} \tag{2}$$

$$\text{TF-IDF} = \mathrm{TF}(S, O) \cdot \mathrm{IDF}(s, o) \tag{3}$$

where s indicates a given malware/benign sample, o represents targeted Opcode/Bytecode, D refers to the applied data set, and N represents the number of sample in the whole data set.

B. Feature Selection

GWO, first introduced in [28], is one of the metaheuristic optimization methods which has been applied in various fields [29]–[31]. GWO is used to facilitate feature selection as it offers an optimum feature set to achieve higher accuracy. Moreover, unlike other feature selection methods, it does not require any threshold parameters to cut the irrelevant features. This module works on the assumption that there are three main types of wolves in nature. The leader is responsible for making decisions, and the leader is referred to as Alpha (α). The second level in the hierarchy of gray wolves are referred to as Beta (β), and the lowest ranking gray wolves are named Omega (ω). If a wolf does not belong to one of the wolf groups, it counts as subordinary or named Delta (δ). In the GWO, α is considered as the best fitness solution. (β) and (δ) are the next fitness functions for generating the suboptimal solution. In our case, they are used to shuffle the selected features.

All types of wolves try to find the best/optimum position for hunting based on several attempts. Therefore, the hunting process can be summarized in the following equations:

$$G = |C \cdot F_{\text{prey}}(t) - F_{\text{wolf}}(t)| \tag{4}$$

$$F(t+1) = F_p(t) - A \cdot G. \tag{5}$$

In the above equation, G is the current position of each wolf, t indicates the current iteration, F indicates the position vector, and F_{prey} is the position vector of the prey. A and C are coefficients and are introduced in the following equations:

$$A = 2ar_1 - a \tag{6}$$

$$C = 2r_2. \tag{7}$$

Algorithm 1 Opcode/Bytecode Feature Selection Algorithm by GWO

function FEATURESELECTIONBYGWO($L*$ <list of all Opcodes>))
  Initialization:
  Initialize wolves' positions, populations, maxIterations;
  Calculate the feature to class mutual information vector;
  Calculate the feature to feature mutual information matrix;

  GWO Optimization Filter-based: ▷ Use mutual information as a fitness function
  α: gray wolf with the highest maximum mutual informarion(fitness);
  β: gray wolf with the second maximum mutual informarion(fitness);
  δ: gray wolf with the third maximum mutual information(fitness);

  GWO Optimization Wrapper-based:
  while l ¡ $max_{iteration}$ **do**
    for $i = 1$:*populationsize* **do**
      Update(current gray wolf position) by Eq. (10);
      $i++$;
    end for
    Update(a, A, C)
    Calculate the fitness if all gray wolves;
    Update(α, β, δ)
    $l = l + 1$;
  end while
  return α; ▷ the highest fitness (the most important feature)
end function

In the above equations, a linearly decreases from 2 to 0 and r_1 and r_2 are random vector lied in [0, 1] interval. Therefore, each type of wolf can find the best position for hunting, which is in our case finding the optimum features to reach high detection accuracy, through several iterations as given in (10)

$$\alpha = |C_1 \cdot F_\alpha - F|, \beta = |C_1 \cdot F_\beta - F|, \omega = |C_1 \cdot F_\omega - F| \tag{8}$$

$$F_1 = F_\alpha - A_1 \cdot (\alpha), F_2 = F_\beta - A_2 \cdot (\beta), F_3 = F_\omega - A_3 \cdot (\omega) \tag{9}$$

$$F(t+1) = \frac{F_1, +F_2 + F_3}{3}. \tag{10}$$

In the above equation, t is the present iteration and F_α, F_β, and F_ω are the location vector of the wolves. Hence, in order to find the best feature set from the IoT data set Opcode and Bytecode properties, GWO is run as given in Algorithm 1. In our scenario, the detection accuracy from selected features is considered as the best position of the wolves obtained from (10).

C. Multikernel Threat Hunting Model

The proposed threat hunting model is using a multikernel SVM classifier to detect IoT malware samples. Since the applied data set includes two classes of samples, the classification task is binary. SVM classifier is very suitable for binary classification problems to draw a good distinction among malware and benign samples. SVM is a supervised ML classifier that is defined with a separating hyperplane. In a 2-D space, this hyperplane can be a line that classifies data into two categories with the same feature set. In high dimensional problems, SVM performs a nonlinear process. In these high dimensional cases, SVM must use multiple kernels to support multiple dimensions to maximize the margin between the classes and to reduce the distance between the hyperplane focuses [32]. The common SVM classifier kernels are as follows.

1) Linear Kernel: It is used when data samples of different class labels can be separated with a simple line as shown in (11), where x and y are vectors of samples (feature vector) for kernel function (k)

$$k(x, y) = x^T y. \tag{11}$$

2) Poly Kernel: It is used to compute the similarity between two vectors. It also considers cross dimensions as

$$k(x, y) = \left(\gamma x^T y + c_0\right)^d. \tag{12}$$

3) RBF Kernel: It calculates the radial basis function (RBF) kernel(k) between two vectors. Equation (13) represents this kernel,where γ is the inverse of the standard deviation of the kernel

$$k(x, y) = \exp\left(-\gamma \|x - y\|^2\right). \tag{13}$$

4) Sigmoid Kernel: It also known as hyperbolic tangent, comes from the neural networks field, where the bipolar sigmoid function is often used as an activation function for artificial neurons. Sigmoid kernel is defined in the following equation:

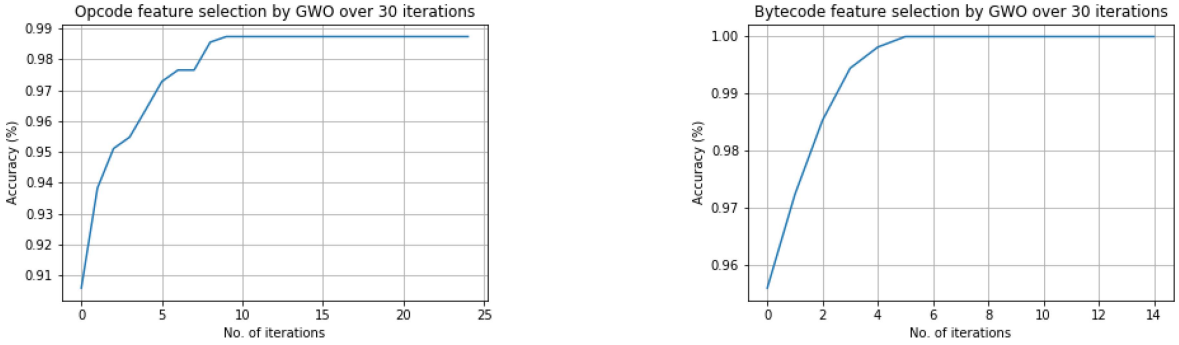
$$k(x, y) = \tanh\left(\gamma x^T y + c_0\right) \tag{14}$$

where x and y are the input vectors, γ is slope and c_0 is known as intercept, and d is degree of polynomial.

In this article, to create the optimal multikernel SVM model every two potential kernels of SVM are combined as shown in [33]. This combination can be achieved by calculating the average of every kernel output as

$$\text{MultiKernel} = \frac{1}{n} \sum_{1}^{n} \text{Kernel}_i(x, y). \tag{15}$$

After transforming the Opcode and Bytecode files into a `TF-IDF` sequential vector, GWO as the feature selection approach is applied to select the optimal subpart of Opcode and Bytecode features. The number of iterations and population in GWO can be varied based on the number of features. Therefore, the population was initialized to a size of 30 over 15 iterations for the GWO. Due to the nature of the GWO techniques, the results are consistent after several iterations, and the objective function converges into the optimal solution after multiple runs. To avoid the local minimum phenomena, the feature selection process is repeated five times to generate the random starting point. Fig. 2 shows the detection accuracy rates of the proposed model on different iteration. For the Opcode, it can be clearly seen that accuracy converges to a constant value after the ninth iteration. Moreover, for the Bytecodes, a constant converged accuracy is reached after the fifth iteration.



(a) (b)

Fig. 2. (a) Detection accuracy in Opcode by GWO over 30 iterations. (b) Detection accuracy of the selected features in Bytecode by GWO over 30 iterations.

TABLE I
TOP 20 SELECTED FEATURES FROM IOT MALWARE OPCODE DATA SET BY GWO

Feature name	Selection Kernel
aam, aas, adc, addb, addl, addps, addw, and, andl, bnd, bndldx, bndstx, bound, bsf, btl, callw, clflush, cmc, cmovae, cmovb	Poly-RBF

Table I shows the top 20 selected features of the Opcode data set based on 30 iterations by the combination of Polynomial and RBF kernels. At the end of feature selection iterations, when constant detection accuracy value was reached, *k*-fold cross-validation was applied on the training set.

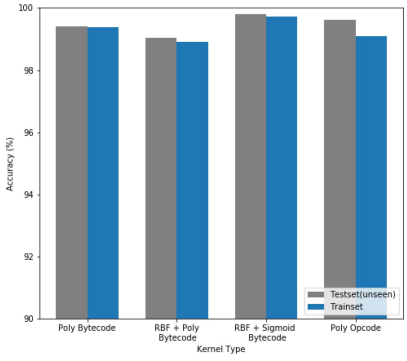


Fig. 3. F1 score of different combinations of SVM kernels on the unseen (test) and train data set.

IV. RESULTS AND DISCUSSION

To evaluate the proposed model, we conducted two sets of experiments. We first evaluated to performance of our model in detecting the malicious and benign IoT applications using common ML performance metrics, such as accuracy (ACC), precision, recall, and F1-Score. Afterward, we compared the computational costs of our model against two recent ML-based IoT malware threat hunting models.

All experiments were processed by Python3.6 and TensorFlow 2.0 [34] environment, which was running on a PC powered by Intel Core i9 CPU with 32-GB RAM and an RTX 2080 Ti GPU.

A. Performance Metrics

The performance of the proposed model is evaluated using the cross-validation technique. The training data set includes 271 benign and 281 malicious Cortex A9 samples. We used unseen samples of IoT malware to verify the robustness of the proposed technique. Unseen malware samples are those collected randomly from VirusTotal by [23] which include 100 Cortex A9 malware samples.

The performance of the proposed model is quantified using the following standard metrics.

1) *True Positive (TP):* When a malicious sample predicted as a malware.
2) *True Negative (TN):* When a malicious sample predicted as a goodware.
3) *False Positive (FP):* When a benign sample predicted as a malicious.
4) *False Negative (FN):* When a malicious sample predicted as a goodware.

Based on the above core metrics, the performance of ML systems can be measured using the following.

Accuracy: Accuracy indicated how the proposed model can accurately predict malware and benign samples

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{(\text{TP} + \text{TN} + \text{FN} + \text{FP})}. \tag{16}$$

Precision: Precision for a certain APT group is the number of samples in a class that are correctly predicted, divided by the total number of samples that are predicted

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}. \tag{17}$$

Recall: Recall for a certain class, is the number of samples in a class that are correctly predicted, divided by the total number of samples in that class

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}. \tag{18}$$

F-Score (F1): F-Score is the harmonic mean of precision and recall. It can be applied as a general classifier performance metric

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{19}$$

TABLE II
EVALUATION METRICS OF POLY-RBF KERNELS BASED ON THE DIFFERENT NUMBER OF EPOCHS ON THE UNSEEN DATA SET SAMPLES' OPCODES

Number of epochs	Accuracy	Precision	Recall	F1Score
3	97.43	95.85	99.27	97.52
5	99.45	99.30	99.64	99.46
7	99.08	98.93	99.28	99.10
10	99.63	99.64	99.64	99.64

TABLE III
EVALUATION METRICS OF POLY-RBF KERNELS BASED ON THE DIFFERENT NUMBER OF EPOCHS ON THE UNSEEN DATA SET SAMPLES' BYTECODE

Number of epochs	Accuracy	Precision	Recall	F1Score
3	97.07	95.80	98.52	97.09
5	97.62	96.86	97.63	97.63
7	98.90	98.26	99.63	98.92
10	99.63	99.63	99.34	99.63

TABLE IV
EVALUATION METRICS OF THE POLY KERNELS COMBINATION BASED ON THE DIFFERENT NUMBER OF EPOCHS ON THE UNSEEN DATA SET SAMPLES' BYTECODE

Number of epochs	Accuracy	Precision	Recall	F1Score
3	98.90	100	97.78	98.88
5	99.45	100	98.89	99.43
7	99.82	100	99.63	99.81
10	99.45	100	98.90	99.44

Findings from our evaluation demonstrated that the proposed model has higher accuracy and lower FP rate with reduced processing time, compared to the existing work in the literature. All experiments were applied to both Opcode and Bytecode representations of the data set samples. Fig. 7 shows the most promising combinations, in terms of detection accuracy for selecting the prominent features from the Opcode and Bytecode properties.

The performance of the proposed model was evaluated on different numbers of epochs using the k-fold cross-validation technique. The different number of epochs lead to the GWO approach to select different features until we reach to the optimum set (see Tables II–V). Since the GWO uses an iterative approach to improve accuracy and to find the optimum feature set, the linear kernel models will be overfitted. Therefore, the linear kernel results were removed from the final outputs of the proposed model.

The performance of the model was also evaluated using Opcode and Bytecodes of unseen malware data set as shown in Fig. 4. As observed, the highest detection accuracy is achieved by the RBF and Sigmoid function over Bytecodes. Hence, it can be concluded that the static properties of the samples were sparse with a semi-Gaussian distribution of feature space.

Fig. 5 illustrates the precision rate of the proposed multikernel model. As we observe, the multikernel model achieved the highest precision rate. The proposed model was also evaluated based on recall and F1-score, to show the strength of the multikernel approach for detecting unseen malware samples (see Figs. 3 and 6). Fig. 7 shows that the combination of all kernels reached the highest detection accuracy. Moreover, the combination of all kernels prevents the overfitting phenomena [35]. Hence, the proposed model achieved high detection accuracy with the low false alarm. As shown in Table VI, the proposed multikernel approach outperforms the other competing approaches in terms of accuracy, precision, recall, and F1-score.

TABLE V
EVALUATION METRICS BASED ON THE DIFFERENT NUMBER OF EPOCHS FOR THE OPCODE ON THE COMBINATION OF ALL KERNELS

Number of epochs	Accuracy	Precision	Recall	F1 Score
3	99.64	99.29	100	99.64
5	99.63	99.64	99.64	99.64
7	99.81	99.64	100	99.82
10	99.63	99.64	99.64	99.64

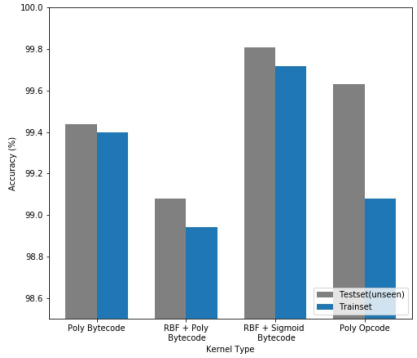


Fig. 4. Detection accuracy of different combinations of SVM kernels on the test and train data set.

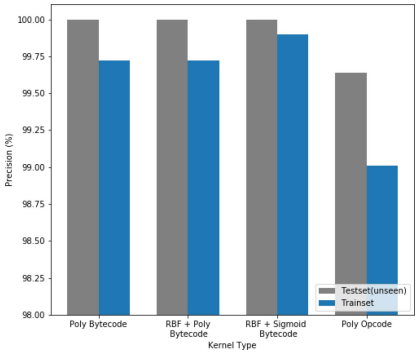


Fig. 5. Precision rates of different combinations of SVM kernels on the test and train data set.

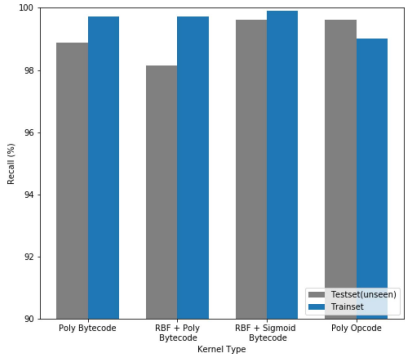


Fig. 6. Recall rates of different combinations of kernels on test and train data set.

TABLE VI
COMPARATIVE SUMMARY

Model	Dataset	Evaluation method	ACC (%)	Precision (%)	Recall (%)	F1-Score (%)
Fuzzy pattern tree [3]	Train	10-fold X validation	98.17	100	98.73	99.3
Maximal-frequent patterns (MFP) [28]	Train	10-fold X validation	99.45	N/A	N/A	98.55
Deep LSTM [29]	Train	k-fold X validation	98.18	N/A	N/A	N/A
Deep LSTM [29]	Test	Full Test	94	N/A	N/A	N/A
Deep Eigen [30]	Train	k-fold X validation	99.68	98.59	98.37	98.48
Santos et al. [27]	Train	k-fold X validation	95.91	86.25	81.55	86.52
Multi-kernel	Test	Full Test	99.63	99.64	99.64	99.64
Multi-kernel	Train	k-fold X validation	99.72	99.9	99.72	99.53

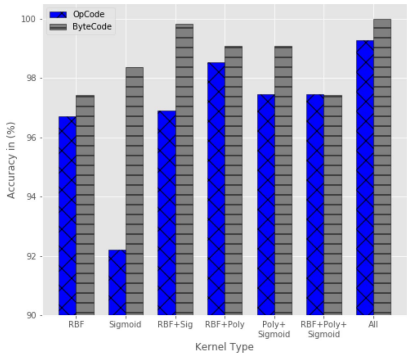


Fig. 7. Overall detection accuracy of the combination of kernels on the unseen malware data set.

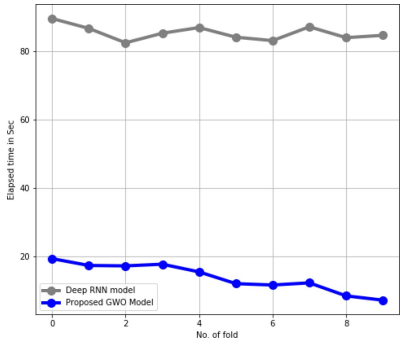


Fig. 8. Comparison of training and testing time between the proposed model and a deep RNN during convergence.

B. Computational Analysis

The SVM classifiers are relaying on the number of free support vectors and the computational complexity of these model can be formulated as $O(\max(n, d) \min(n, d)^2)$ [36], where n is the number of sample and d is the number of dimension. In contrast, the artificial neural network (ANN)-based models computational complexity can be formulated as $O(n^4)$ and $O(n^5)$ for forward propagation and backward propagation approaches, respectively. Hence, the proposed model should have a much lower computational cost due to its computational order as well as its smaller number of features due to the optimal feature reduction algorithm.

We compared the computational costs of the proposed model with two recent ML-based IoT malware threat hunting models, namely, a deep RNN model [23] and a fuzzy model [3]. Fig. 8 shows a comparison between the training and testing time in different epochs of cross-validation between the proposed model and the deep RNN-based model (LSTM).

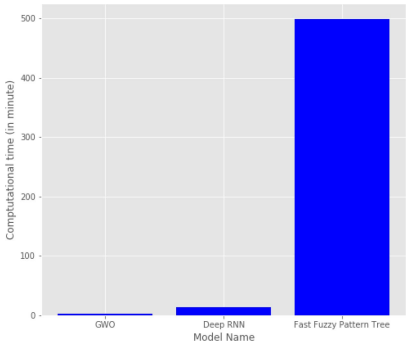


Fig. 9. Comparison of the overall computational time of the proposed model with previous deep RNN and fuzzy pattern tree models.

While the deep model requires more than 1 min to converge, the proposed model achieves the same conversion in less than 20 s. As we could not access the fuzzy model source code we were not able to measure its computational time using the cross-validation process.

However, as shown in Fig. 9, we could compare the overall computation time (training + testing time) of all three models [deep neural network (DNN), fuzzy, and our proposed model]. As can be seen, the proposed model overall computation time is four times less than the deep RNN approach and 50 times less than the fuzzy approach.

We have shown that our proposed multikernel approach achieves increased detection accuracy, with the significantly reduced computational cost.

V. CONCLUSION

IoT devices are increasingly targeted, partly evidenced by the number of reported attacks and detected malware. In this article, a multikernel SVM model based on GWO (a metaheuristic feature selection approach) was designed to detect malware targeting ARM processors on cloud-edge devices. Our proposed approach uses malware's static properties (Opcode and ByteCode). We designed a preprocessing module to transfer the textual content of each sample into a vector using the TFIDF metric. Afterward, a feature selection module is used to reduce the number of features and minimize the computational costs of the proposed model. Finally, a multikernel SVM classifier is used to accurately identify IoT malware. The model achieved a high accuracy rate of 99.72%, outperforming previous deep learning and fuzzy ML based IoT malware detection models. Moreover, the computational analysis of the proposed model showed that the proposed model

converges faster than existing ML systems, such as DNNs and fuzzy pattern three classifiers.

In the future, we will develop a multikernel approach for malware threat hunting using other ML algorithms. Moreover, using multikernel approaches for malware threat attribution is another potential future research.

REFERENCES

[1] I. Yaqoob, I. Abaker, T. Hashem, A. Ahmed, and S. M. A. Kazmi, "Internet of Things forensics: Recent advances, taxonomy, requirements, and open challenges," *Future Gener. Comput. Syst.*, vol. 92, pp. 265–275, May 2019.

[2] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1686–1721, 3rd Quart., 2020.

[3] E. M. Dovom, A. Azmoodeh, A. Dehghantanha, D. E. Newton, R. M. Parizi, and H. Karimipour, "Fuzzy pattern tree for edge malware detection and categorization in IoT," *J. Syst. Archit.*, vol. 97, pp. 1–7, Aug. 2019.

[4] H. H. Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo, "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IOT backbone networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 7, no. 2, pp. 314–323, Apr./Jun. 2019.

[5] C. D. Mcdermott, F. Majdani, and V. A. Petrovski, "Botnet detection in the Internet of Things using deep learning approaches," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)* Rio de Janeiro, Brazil, 2018, pp. 1–8.

[6] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer Internet of Things devices," in *Proc. IEEE Security Privacy Workshops (SPW)*, San Francisco, CA, USA, 2018, pp. 29–35.

[7] H. HaddadPajouh, R. Khayami, A. Dehghantanha, K.-K. R. Choo, and R. M. Parizi, "AI4SAFE-IoT: An AI-powered secure architecture for edge layer of Internet of Things," *Neural Comput. Appl.*, vol. 32, pp. 16119–16133, Feb. 2020.

[8] X. Zhang, O. Upton, N. L. Beebe, and K.-K. R. Choo, "IoT botnet forensics: A comprehensive digital forensic case study on Mirai botnet servers," *Forensic Sci. Int. Digit. Invest.*, vol. 32, pp. 51–60, Apr. 2020.

[9] M. Antonakakis *et al.*, "Understanding the Mirai botnet," in *Proc. 26th USENIX Security Symp. (USENIX Security)*, 2017, pp. 1093–1110.

[10] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[11] A. N. Jahromi *et al.*, "An improved two-hidden-layer extreme learning machine for malware hunting," *Comput. Security*, vol. 89, Feb. 2020, Art. no. 101655.

[12] H. Haddadpajouh, A. Azmoodeh, A. Dehghantanaha, and R. M. Parizi, "MVFCC: A multi-view fuzzy consensus clustering model for malware threat attribution," *IEEE Access*, vol. 8, pp. 139188–139198, 2020.

[13] H. H. Pajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "Intelligent OS X malware threat detection with code inspection," *J. Comput. Virol. Hacking Techn.*, vol. 14, no. 3, pp. 213–223, 2018.

[14] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *Proc. Int. Work Conf. Artif. Neural Netw.*, 2005, pp. 758–770.

[15] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Elect. Eng.*, vol. 40, no. 1, pp. 16–28, 2014.

[16] J. Li *et al.*, "Feature selection: A data perspective," *ACM Comput. Surveys*, vol. 50, no. 6, p. 94, 2018.

[17] Q. Al-Tashi, S. J. A. Kadir, H. M. Rais, S. Mirjalili, and H. Alhussian, "Binary optimization using hybrid grey wolf optimization for feature selection," *IEEE Access*, vol. 7, pp. 39496–39508, 2019.

[18] J. Kang, S. Jang, S. Li, Y.-S. Jeong, and Y. Sung, "Long short-term memory-based malware classification method for information security," *Comput. Elect. Eng.*, vol. 77, pp. 366–375, Jul. 2019.

[19] B. Cakir and E. Dogdu, "Malware classification using deep learning methods," in *Proc. ACMSE Conf.*, 2018, pp. 1–5.

[20] D. Yuxin and Z. Siyi, "Malware detection based on deep learning algorithm," *Neural Comput. Appl.*, vol. 31, no. 2, pp. 461–472, 2019.

[21] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Inf. Sci.*, vol. 231, pp. 64–82, May 2013.

[22] H. Darabian, A. Dehghantanha, S. Hashemi, S. Homayoun, and K. K. R. Choo, "An opcode-based technique for polymorphic Internet of Things malware detection," *Concurrency Comput.*, vol. 32, pp. 1–14, Mar. 2020.

[23] H. Haddadpajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep recurrent neural network based approach for Internet of Things malware threat hunting," *Future Gener. Comput. Syst.*, vol. 85, pp. 88–96, Aug. 2018.

[24] A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust malware detection for Internet of (battlefield) Things devices using deep Eigenspace learning," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp. 88–95, Jan.–Mar. 2019.

[25] S. Jeon and J. Moon, "Malware-detection method with a convolutional recurrent neural network using opcode sequences," *Inf. Sci.*, vol. 535, pp. 1–15, Oct. 2020.

[26] Y. Ding, X. Zhang, J. Hu, and W. Xu, "Android malware detection method based on bytecode image," *J. Ambient Intell. Hum. Comput.*, pp. 1–10, Jun. 2020.

[27] J. Ramos, "Using TF-IDF to determine word relevance in document queries," in *Proc. 1st Instruct. Conf. Mach. Learn.*, vol. 242. Piscataway, NJ, USA, 2003, pp. 133–142.

[28] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[29] S. Mirjalili, "How effective is the grey wolf optimizer in training multi-layer perceptrons," *Appl. Intell.*, vol. 43, no. 1, pp. 150–161, 2015.

[30] S. A. Davidsen and M. Padmavathamma, "Multi-modal evolutionary ensemble classification in medical diagnosis problems," in *Proc. IEEE Int. Conf. Adv. Comput. Commun. Informat. (ICACCI)*, Kochi, India, 2015, pp. 1366–1370.

[31] S. Saremi, S. Z. Mirjalili, and S. M. Mirjalili, "Evolutionary population dynamics and grey wolf optimizer," *Neural Comput. Appl.*, vol. 26, no. 5, pp. 1257–1263, 2015.

[32] Y. Ma and G. Guo, *Support Vector Machines Applications*, vol. 649. Cham, Switzerland: Springer, 2014.

[33] A. Kaveh and P. Zakian, "Improved GWO algorithm for optimal design of truss structures," *Eng. Comput.*, vol. 34, pp. 685–707, Oct. 2018.

[34] M. Abadi *et al.*. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: tensorflow.org

[35] F. Aiolli and M. Donini, "EasyMKL: A scalable multiple kernel learning algorithm," *Neurocomputing*, vol. 169, pp. 215–224, Dec. 2015.

[36] L. Bottou and C.-J. Lin. (2007). *Support Vector Machine Solvers Large Scale Kernel Machines*. [Online]. Available: https://doi.org/10.7551/mitpress/7496.003.0003