A Novel Threat Intelligence Detection Model Using Neural Networks

SALEM, Maher <http://orcid.org/0000-0002-6479-4335> and AL-TAMIMI, Abdel-Karim

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 November 2022, accepted 8 December 2022, date of publication 15 December 2022,
date of current version 21 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3229495

RESEARCH ARTICLE

A Novel Threat Intelligence Detection Model Using Neural Networks

MAHER SALEM[1] AND ABDEL-KARIM AL-TAMIMI[2,3]
[1]Department of Informatics, King's College London, WC2R 2LS London, U.K.
[2]Department of Computing, Sheffield Hallam University, S1 1WB Sheffield, U.K.
[3]Computer Engineering Department, Yarmouk University, Irbid 21163, Jordan

Corresponding author: Maher Salem (maher.salem@kcl.ac.uk)

This work was supported by the German Federal Ministry of Education and Research under Grant 17062X10.

ABSTRACT A network intrusion detection system (IDS) is commonly recognized as an effective solution for identifying threats and malicious attacks. Due to the rapid emergence of threats and new attack vectors, novel and adaptive approaches must be considered to maintain the effectiveness of IDSs. In this paper, we present a novel Threat Intelligence Detection Model (TIDM) for online intrusion detection. The proposed TIDM focuses on the online processing of massive data flows and is accordingly able to reveal unknown connections, including zero-day attacks. The TIDM consists of three components: an optimized filter (OptiFilter), an adaptive and hybrid classifier, and an alarm component. The main contributions of the OptiFilter component are in its ability to continuously capture data flows and construct unlabeled connection vectors. The second component of the TIDM employs a hybrid model made up of an enhanced growing hierarchical self-organizing map (EGHSOM) and a normal network behavior (NNB) model to jointly identify unknown connections. The proposed TIDM updates the hybrid model continually in real-time. The model's performance evaluation has been carried out in both offline and online operational modes using a quantitative approach that considers all possible evaluation metrics for the datasets and the hybrid classification method. The achieved results show that the proposed TIDM is able, with promising performance, to process massive data flows in real-time, classify unlabeled connections, reveal the label of unknown connections, and perform online updates successfully.

INDEX TERMS Neural networks, GHSOM, EGHSOM, NNB, threat intelligence, data processing, intrusion detection, clustering.

I. INTRODUCTION

Successful security management is the key to enhancing network services and boosting their management [1], [2], [3]. It provides large computer networks with integrity, availability, and confidentiality of data [4], [5]. The revolutions in networking and information technology have increased interconnection and user interactions and, thus, demand a significant number of services and increased data management efforts. Furthermore, this has led to the generation of massive amounts of data, which makes online processing of the continuous data flow very challenging, hence increasing the chances of successful attacks [6], [7]. In addition, security aspects such as standard security gateway architectures, anti-malware solutions, signature-based intrusion detection methods, and other security tools are still suffering from the complexity of big data in terms of data management and the identification of suspicious connections [8], [9], [10], [11].

Current security approaches, particularly intrusion detection systems (IDSs), rely strongly on the collected data of the network under concern to perform the required monitoring and intrusion detection processes [12], [13]. Principally, an IDS aggregates data and preprocesses them to identify anomalous connections. Revealing the anomalous connections in an *offline* operational mode is a feasible task for most IDSs due to the plausible amount of aggregated data [14].

The associate editor coordinating the review of this manuscript and approving it for publication was Xinyu Du.

However, the massive increase in data flows nowadays hinders the analysis and management of the aggregated data and results in poor performance due to the inability to classify unknown connections into normal or anomalous connections [15], [16].

In addition, big data with its voluminous data streams has rapidly intensified the amount of data flows and degraded the overall performance of the IDS model [17], [18]. Therefore, capturing online data flows, processing them into a proper format for the IDS model, and then detecting any threat is still considered a significant challenge in the field of network intrusion detection research [19]. Furthermore, most IDS models cannot accommodate the huge number of flows and often classify them incorrectly. Thus, an adaptive and practical Threat Intelligence Detection Model (TIDM) that overcomes these challenges is essential in the area of IDSs.

A threat intelligence model gathers raw data about existing and emerging threats and associated threat actors from several sources, and then analyzes and filters the data to produce useable information in the form of management reports and data feeds for automated security control systems. Its primary purpose is to help organizations understand the risks they are exposed to and better protect against zero-day threats, advanced persistent threats and exploits—especially those that are most likely to affect their own specific environments [20].

In this paper, we present an adaptive and practical TIDM that is able to process massive online data flows, and identify and label unknown connections. The TIDM consists of the following main components: a) a novel and optimized filter (viz. OptiFilter), b) an adaptive hybrid classifier, and c) an alarm. The main contributions of the optimized filter (OptiFilter) are as follows: 1) capture massive flows of network packets and hosts' events continuously; 2) process them in a queue of a dynamic window size; and 3) construct unlabeled connection vectors continuously. The hybrid classifier employs a neural network hybrid model that uses both enhanced growing hierarchical self-organizing map (EGHSOM) and normal network behavior (NNB) modeling approaches. The main contributions of the hybrid classifier are as follows: 1) The EGHSOM model classifies the unlabeled connections as ''normal'', ''anomaly'' or ''unknown''; 2) the NNB model examines whether the unknown connections are ''normal'' or ''anomalous''; and 3) novel online updating methods for both EGHSOM and NNB models in the proposed TIDM are presented to update the hybrid model constantly in real-time.

The rest of this paper is organized as follows: Section II explains key concepts and foundations related to this research. Section III provides a comprehensive discussion of the related work. The theoretical and practical declarations of the proposed TIDM architecture are discussed in Section IV. The testing and evaluation of the model are addressed in Section V. Section VI discusses the model's performance, and Section VII draws conclusions based on our work.

II. PRELIMINARY/BACKGROUND

In this section, we will explore the key concepts and fundamentals related to the proposed TIDM model and establishes the key knowledge areas associated with IDS data collection and classification.

A. DATA COLLECTION

Usually, network data flows and internal users' activities are the sources for any IDS. If we observe a computer network for a certain period of time, the observed data flows can be represented by a dataset D (i.e., a set of connection vectors that are constructed from network packets and hosts' events). These connections are formally described as feature vectors $x_1, \ldots, x_M$, where each vector consists of n attributes (features). Let $x_i = (x_{i1}, \ldots, x_{in}) \in \Omega_1 \times \ldots \times \Omega_n$, $i = 1, \ldots, M$ be a feature vector such that $x_{i1} = \text{attribute1}, x_{i2} = \text{attribute2}$, and so forth. Let D be a dataset constructed with M such feature vectors. Accordingly, the dataset D can be formally described as a matrix X_M in the input space of the computer network as follows:

$$\Omega_M : X_M := \begin{pmatrix} x_1 \\ \vdots \\ x_M \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{M1} & \cdots & x_{Mn} \end{pmatrix} \quad (1)$$

If we consider each column $x_j = (x_{1j}, \ldots, x_{Mj}), j = 1, \ldots, n$ in the matrix as M realizations of the j^{th} attribute, then we can interpret each feature as a realization of a random variable X_j of the space Ω_j. Hence, a dataset D consists of $M \times n$ values of the n-dimensional discrete random variables $X = (X1, \ldots, Xn)$ at $\Omega_M := \Omega_1 \times \ldots \times \Omega_n$.

B. IDS TRAFFIC CLASSIFICATION

Intelligent detection systems (IDSs) rely mainly on the following machine learning techniques: statistical-based techniques, such as Naïve Bayes [21], [22], anomaly-based techniques, such as [23], [24], [25], neural networks [26], [27], [28], deep neural networks [29], and customized clustering techniques [30], [31]. The use of artificial neural networks (ANNs) has become the most effective IDS method in network security [28], whereas the most successful applications of neural networks are in classification or categorization and pattern recognition [32]. A growing hierarchical self-organizing map (GHSOM) is considered a special ANN approach and is an improved version of Kohonen's self-organizing map (SOM) [33], [34], [35].

Generally, SOMs can discover knowledge in data and find relations of high-dimensional data and then map these data into a two-dimensional representation space [36], [37]. However, they suffer from static architecture and expensive computational requirements. GHSOMs have solved these problems by structuring several SOMs in a hierarchical growing form [38]. Note that neural networks handle only numeric data and most likely with the same scale. This requires that all nominal values within a connection vector must be transferred to numeric values and then the entire connection must be normalized.

C. GHSOM MODEL

A GHSOM construes high-dimensional data on several layers with several maps to explore supplementary details. Although the training of each map in a GHSOM is the same as with a SOM, a GHSOM controls the growth process horizontally and vertically by examining two main threshold criteria after a certain number of iterations. The main processes of a GHSOM are the competition and the cooperation between neurons [38]. GHSOMs are used to detect anomalies in computer networks [39] and have shown significant results in this regard. The training process of a GHSOM is summarized in the following steps:

1) INPUT

Let D be a dataset with M different connection vectors $[x_1, x_2, \ldots, x_M]^T$. Initialize $layer_0$ with a single root neuron that has the weight vector $w_0 = \frac{1}{M}\sum_{i=1}^{M} x_i$ and then calculate the *quantization error* (qe) and the *mean quantization error(mqe)* of the root node as follows:

$$qe_0 = \sum_{i=1}^{M} \|x_i - w_0\|, \quad mqe_0 = \frac{1}{M} \cdot qe_0 \tag{2}$$

Initialize $layer_1$ with a single map that has four neurons and initialize the weight vector of each neuron randomly. Finally, determine the number of iterations λ (e.g., $\lambda = 400$).

2) COMPETITION PROCESS

Select a random input vector x_j from D and find its closest neuron c (i.e., the winner or the best-matching unit (BMU)), c is determined as follows:

$$c := argmin_i\{\|x_j - w_i\|\}, \tag{3}$$

where i is the number of neurons on the map.

3) COOPERATIVE PROCESS

In this step, we update the neuron c and its neighbor neurons N_c as follows:

$$w_i(t+1) := w_i(t) + h_{ci}(t)[x(t) - w_i(t)], \quad i \in N_c, \tag{4}$$

where $h(t) := a(t) \cdot e^{\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}}$ is a Gaussian neighborhood function with a learning rate $\alpha(t)$, radius σ, BMU position r_c, and the posit ion r_i of the closest unit in the neighborhood of the BMU. The competition and cooperation processes are repeated for λ iterations.

4) GROWTH CONTROL

Determine the control thresholds τ_1 and τ_2 such as $0 < \tau_1, \tau_2 < 1$ then calculate the minimum quantization error (MQE) of the map as

$$MQE_m = \frac{1}{n}\sum_{i=1}^{n} qe_i, \tag{5}$$

where n is number of neurons in the map m. Then, check the following horizontal growth condition:

$$MQE_m < \tau_1 \cdot qe_p, \tag{6}$$

where p is the map's parent node on the upper layer. The highest error node (e) on map m and its most dissimilar i_d in the neighborhood will be calculated if the horizontal growth condition is not met:

$$e = \underset{i \in Map}{argmax}\left\{\sum_{x_j \in RF_i} \|x_j - w_i\|\right\},$$
$$i_d = \underset{i \in Map}{argmin}\left\{\sum_{x_j \in RF_i} \|x_j - w_i\|\right\}, \tag{7}$$

where RF_i is the receptive field that represents the set of all the input vectors mapped on the same neuron i. Based on that, insert a new column or row between e and i_d, then initialize the new nodes randomly, reset $\alpha(t)$ and $h(t)$, and repeat the competition and cooperative processes with new λ iterations.

But, if the horizontal growth condition (equation 6) is met, calculate the mqe of each neuron on the map m individually and for each neuron i and examine the following vertical growth condition:

$$mqe_i < \tau_2 \cdot mqe_p \tag{8}$$

If the vertical growth condition is not met then add a new layer with a single map that has four neurons, initialize their weight vectors randomly, and repeat the competition and cooperative processes on the new map accordingly. However, if the condition is met then stop the training process.

The GHSOM model can be defined as

$$(C, w_C, \text{Labels}) \tag{9}$$

Accordingly, the final model of the GHSOM after the training is shown in Table 1.

TABLE 1. Final GHSOM model.

Output	Description
BMUs	$C := \{c_1, c_2, \ldots, c_m\},\ m = \lvert\{i \in \aleph \mid RF_i \neq \phi\}\rvert$
Receptive Fields of BMUs	$RF_i := \{x \in R^n \mid \Phi(x) = i,\ x \in \Omega_M\},\ i = 1, \ldots, m$
Weight Vectors of BMUs	$w_c = (w_1, \ldots, w_m)$

where the projection functions or $\Phi(x)$, the receptive field of the BMU c or RF, the GHSOM grid or $\aleph$, number of final BMUs or m, and M is the number of instances in the input dataset.

III. RELATED WORK

We have meaningfully adopted the idea presented by S. Babu and J. Widom, who proposed a general and flexible architecture for processing continuous queries in the presence of data streams and presented a prototype that manages data streams accordingly [40]. The essence of their idea has inspired us to propose an evolutionary queuing concept approach [41], which is further improved in this paper.

Several approaches have investigated the challenges of handling continuous data flows and detecting suspicious connections on computer networks. F. Hashim et al. introduced

a framework that identifies security attacks from cooperation among network entities [42]. The proposed work consists of an anomaly detection TIDM that is based on negative selection and danger theory, and a security control module that incorporates a security update process and an attack recovery process. Their anomaly detection approach is responsible for detecting epidemic and pandemic attacks, whereas the security control module administrates the security update process. Although this work is considered promising in detecting attacks on heterogeneous networks, it does not investigate the propagation of new or undetected attacks.

C. J. Fung et al. proposed a distributed host-based collaborative IDS (CIDS) to enhance the overall network security and evaluate the trade-off between maintenance cost and intrusion cost, using Bayesian learning and a Bayesian decision model [43]. They proposed a simulated CIDS environment to evaluate the Bayesian decision model against threshold-based decision models, and an acquaintance selection algorithm against the brute force approach. As a result, they achieved better performance with less computation time. However, this model focused on the host-based CIDS to perform the comparison and cannot be considered a standalone model for detecting new attacks.

The idea of developing distributed IDSs has also been examined by Fisch et al. [44]. They proposed a detection model for large-scale collaborative intrusion detection agents (IDAs). The detection model includes no central control unit, but instead uses distributed and self-organized agents that operate in four separate layers: sensor layer, detection layer, alert correlation layer, and reaction layer. These agents work in a decentralized manner through data acquisition, analysis, and communication. The detection model focuses on large-scale environments where agents can communicate and synchronize information. However, agent-based solutions are not preferable due to the associated time-consuming communications that might fail during data exchange.

Fisch et al. proposed an organic computing technique for detecting attacks based on probabilistic rule modeling [45]. The work is based on the IDA methodology that recognizes new attacks and reacts by creating new rules and exchanging them with other agents. Although the proposed technique was evaluated using the offline DARPA traffic, it achieved a self-adaptive status and promising results, especially in the area of organic computing.

Zhang et al. investigated the detection of unknown attacks by using an effective network traffic classification method [15]. Their proposed method introduces a flow label-propagation technique to label the flows, and a compound classification process to classify correlated flows in a bag-of-flows model instead of classifying them individually. The results show that their work outperforms other traditional classification methods, such as Naïve Bayes.

Lee et al. [14] proposed the use of oversampling principal component analysis to achieve online anomaly detection by detecting the effects of introducing outliers to a normal data flow. The detection model is based on characterizing a normal data flow profile to help identify any outlier data point. The detection model achieves reasonable results using synthetic datasets and provides satisfactory results in detecting outliers.

Ortiz et al. [39] presented a method using a GHSOM that requires only one parameter to govern the growth process, instead of two parameters as in typical GHSOM-based Anomaly Detection systems (ADSs), to detect anomalous traffic. In their approach, they used a probabilistic model to train their model online.

The work of Sperotto et al. [46] concentrated on SSH traffic as a binary classification problem (benign or malicious) to automatically tune the IDS parameters by formalizing the relation between them and the performance metrics.

Biggio et al. [47] proposed a framework for empirical evaluations of security classifiers in adversarial environments to improve their design. Their main goal was to provide a quantitative and general-purpose basis for the application of the *what-if* analysis to classify security evaluations based on the definition of potential attack scenarios. They proposed: 1) a model of adversaries that allows us to define attack scenarios; 2) a corresponding model for data distribution; and 3) a method for generating training and testing sets that are representative of the data distributions used.

On the other hand, Zhang et al. [48] proposed a novel ensemble-tree (E-tree for short) indexing structure to organize all the base classifiers in an ensemble for fast prediction. This framework was motivated by the fact that there is a linear increase in the prediction time associated with the increase of the ensemble size used. The proposed E-tree structure can achieve logarithmic time complexity for prediction.

Ferrag et al. [11] conducted a comparative study of the latest deep learning techniques, including recurrent neural networks (RNNs), restricted and deep Boltzmann machines, convolutional neural networks (CNNs), and deep belief networks. They also surveyed the datasets used to categorize them for better utilization in future empirical studies. They concluded that CNNs achieved a slightly better classification performance than other deep learning techniques.

Jiang et al. [49] proposed an offline, multi-channel intelligent attack detection approach based on the long short-term memory recurrent neural network deep learning technique. The authors used an ensemble of RNN classifiers trained on basic and content-based features, basic and traffic-based features, and all features. The classifiers' predictions were then fed into a voting system to decide whether the traffic should be classified as benign or malicious. Their initial experimental results show that they outperformed several classic machine learning (KNN, SVM, and Bayesian) and deep learning (GRNN, PNN, and RBNN) techniques.

Liang et al. [50] proposed an offline multi-feature data clusterization optimization mode. Their proposed approach outperformed the deep learning technique proposed in [51], which is based on stacking nonsymmetric deep autoencoders to create a deep learning hierarchy. They compared their results using an NSL-KDD dataset.

Purohit et al. [52] proposed the use of blockchain to share threat information to improve cooperative and collective defense across multiple domain entities. The authors tested the proposed system using the Open Cloud testbed. The shared results demonstrated the system's ability to effectively choose the best peers for threat detection and mitigation.

In summary, our study of related topics in this area confirms that there are several detection models that have achieved satisfactory results. However, unlike the aforementioned proposed IDSs and others such as [53], [54], and [55], our proposed detection model can continuously capture network packets and hosts' events and process them to construct connection vectors based on certain features. In addition, the proposed model is exceptionally capable of detecting new attacks from the identified unknown connections using the combination of EGHSOM and NNB models. Moreover, it is an adaptive approach as it constantly updates its EGHSOM and NNB models based on the current network traffic.

IV. TIDM ARCHITECTURE

The proposed TIDM architecture, shown in Figure 1, consists of three major components: the optimized filter (OptiFilter), which handles and preprocesses all network- and host-captured data; the adaptive hybrid classifier, which is responsible for detecting and classifying the abnormal and unknown connections; and an alarm or a notification component to alert subscribed services. These three components work in a consistent and harmonious manner, as OptiFilter collects data from the network and hosts continuously and with high efficiency and converts it into appropriate data for the adaptive hybrid classifier, which in turn detects and classifies potential and unknown threats. Accordingly, the third component, i.e. Alarm, issues a notice to the concerned authorities.

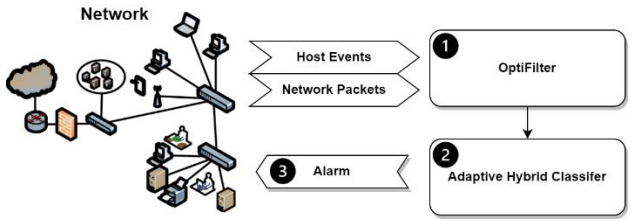

FIGURE 1. The proposed TIDM architecture with OptiFilter.

A. OPTIFILTER

The term ''OptiFilter'' stands for ''optimized filtering''. It consists of data aggregators, a queue as a container for predetermined time slot windows, and a connections exporter. The queue in our method can contain n windows, and each window w is a time slot of t seconds (e.g., 5 seconds). Figure 2 illustrates the internal structure of an OptiFilter with a 5-second time slot window. OptiFilter has been designed and developed in such a way that is configurable using an XML configuration file containing all parameters. One of these parameters is the time slot window. We tested our TIDM in a real industrial partner network for two hours long and with a 5-second time slot, in which the model worked smoothly without dropping any packet and with detecting all constructed connections.

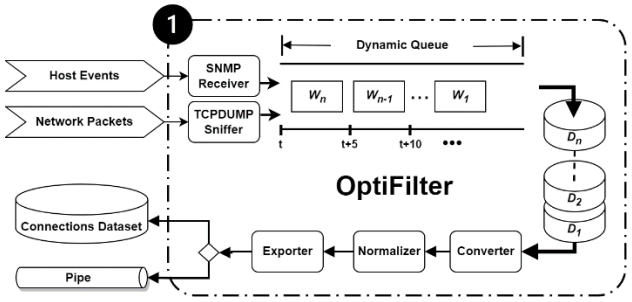

FIGURE 2. The internal structure of the OptiFilter.

Basically, correlated hosts' events and network packets will be constructed as connection vectors in w_1 and pushed forward in the queue (now the queue has 5-sec window); then, the next window w_2 will take place so that a queue with 10 seconds is occupied (w_1 and w_2), and so on. During any window's existence in the queue, all the constructed connections of this window will be continuously modified until the window is pushed out of the queue. Based on the work in [41], at the end of the queue, each time slot window w_i will contain a certain number of connection vectors D_i (also known as a dataset) and will be pushed out of the queue in the form $w_n, w_{n-1}, \ldots, w_1 \rightarrow D_n, D_{n-1}, \ldots, D_1$ (first-in, first-out, viz. FIFO concept). In this work, we have further improved the OptiFilter to manage the massive network data flows in the online operational mode. These improvements are in transferring the nominal features into numeric ones and normalizing all features in the constructed connections into the range [0,1]. The new improvements in the OptiFilter are illustrated in the following sections.

1) PROPOSED CONVERSION AND NORMALIZATION METHODS

Most classifiers in IDSs handle only numeric values: specifically, models based on neural networks [56]. Hence, the proposed datasets in the IDS area need to be preprocessed and prepared in an appropriate format for the detection model.

Several classifiers ignore nominal features if they are not converted into numeric values. Moreover, a feature scale that has a large numeric value will dominate any process, and small values of other features will be devolved and become ineffective. For instance, *protocol_type* is a feature with string values; likewise, the feature *logged_in* has small values, while the feature *source_byte* often has large values. Generally, researchers prepare a proper dataset format in three individual steps: data collection, use of a tool for conversion, and normalization. Notably, this approach is time-consuming and requires additional effort. In this regard, we propose a plausible and meaningful conversion method and a minimum-maximum normalization method to scale the features into the

range [0,1]. These methods are considered to be the preprocessing step and are applied to each generated dataset from the queue, as shown in Figure 2. Considering equation 1, network traffic can be described with M feature vectors of finite dimension, where each element represents a specification of a discrete random variable X_j with values from the probability space Ω_j. Let, then, X_j be a random variable with nominal values while $x_{1j}, x_{2j}, \ldots, x_{Mj}$ are samples with K_j different nominal types $nom^j_1, \ldots, nom^j_{K_j}$. We obtain the absolute frequency r_{kj} of the nominal type nom^j_k, $k = 1, \ldots, K_j$ in X_j as

$$r_{kj} = |\{i \in \mathrm{N} | x_{ij} = norm^j_k, i = 1, \ldots, M\}|. \quad (10)$$

Then $\sum_{k=1}^{K_j} r_{kj} = M$ and $0 \leq \frac{r_{kj}}{M} \leq 1$. Hence,

$$f_{kj} := \frac{r_{kj}}{M}, \quad k = 1, \ldots, K_j \quad (11)$$

is called the relative frequency occurrence of the nominal feature type nom^j_k in X_j. Finally, by using these relative frequencies, we define the mapping $pmf : \Omega_j \rightarrow [0, 1]$ that transfers each nominal feature $x_{kj} \in \Omega_j$ into a real number as $pmf(x_{kj}) = f_{kj}$. These steps for the conversion are defined in Figure 3 as *Converter* (X_i).

After converting all nominal features in the current dataset, all features have numeric values with different scales and can be normalized into a new scale *[a,b]* (e.g., $a = 0$ and $b = 1$). Let $f : I \rightarrow [a, b]$, $I = [min, max]$ be the minimum-maximum normalization function and $v \in R$ the numerical value of an element in X_j, then the normalized feature value nv after the normalization process is represented as

$$nv := \frac{v - min(X_j)}{max(X_j) - min(X_j)} \cdot (b - a) + a. \quad (12)$$

Algorithm 1 list the algorithm that contains the conversion and normalization methods employed in the OptiFilter component. The algorithm consists of two main processes, that is, conversion and normalization. The conversion process starts with verifying each feature vector X if it's a nominal or numeric one, so that the feature vector with nominal values will be converted into a real number as stated in equations 10 and 11. The outcome of the conversion process generates a numeric dataset, which will be then used as input to the next process, i.e. normalization. In the normalization process, all feature vectors will be normalized into the same range to avoid any feature dominancy. Accordingly, the output of this algorithm will be a completely normalized dataset.

Algorithm 1 The Conversion and Normalization Steps

Require: D_1, D_2
  while true **do** *continuous loop*
    $conn \leftarrow size(D_i)$ *number of connections in D*
    – – ***Start Conversion*** – –
    while $i = 1 : n$ **do** *number of features*
      if X_i *is nominal* **then**
        $Converter(X_i)$;
      else
        $Output \leftarrow X_i$
      end if
    end while
    – – ***Start Normalization*** – –
    while $j = 1 : n$ **do** *number of features*
      while $k = 1 : conn$ **do**
        $nv_{jk} := \frac{v - min(X_j)}{max(X_j) - min(X_j)} \cdot (b - a) + a$
        $v_{jk} \leftarrow nv_{jk}$
      end while
    end while
    $Output \leftarrow$ *normalized dataset* $\dot{D}_i$
  end while

2) EXPORTING CONNECTION VECTORS

The last component in the OptiFilter is the exporter. Principally, all normalized datasets $\dot{D}n, \dot{D}n - 1, \ldots, \dot{D}1$ are exported in CSV format. The CSV format is easy and can be handled effectively in real-time. The exporter can combine them to present a single dataset of all connections or can send them in a pipe based on the FIFO concept. Exporting a single dataset enables it to be used for different purposes, such as evaluating IDS models in offline mode. However, to classify the data flow in the online mode, the pipe concept is particularly appropriate and is preferable in research and development.

As a result, the OptiFilter can constantly capture data flows and export them as connections in CSV format. Note that the OptiFilter can be configured to perform certain steps by enabling them and disabling others. For instance, users can enable the normalizer and disable the converter. Figure 3 shows a sample dataset that has been generated directly from the OptiFilter in real-time. Some fields, such as IP addresses, have been hidden due to data privacy. Note that our OptiFilter exports unlabeled connection vectors.

The following parameters have been used to generate the sample dataset, Queue capacity = 3 windows, Time slot window w = 5-sec, and Backlog = 1000 connections.

B. ADAPTIVE HYBRID CLASSIFICATION MODEL

The adaptive hybrid classifier, shown in Figure 4, is the second part of the proposed TIDM architecture. Accordingly, it updates the model to keep the response of the proposed TIDM adaptive. The adaptive classifier consists of a) an EGHSOM model that receives connection vectors from the OptiFilter using the pipe concept, see Figure 2, and classifies them into ''normal'', ''anomaly'', and ''unknown'', b) an NNB model that examines whether the unknown connections are ''normal'' or ''anomalous'' based on the continuously updated dataset of normal connections, and update models to ensure the adaptivity of the architecture. The alarm module is activated based on the joint EGHSOM and NNB models decisions that are fed into the controller.

This part of the TIDM also includes management components—i.e., the controller and the alarm. The controller receives classified data flows from the EGHSOM

Features →

timestamp	src_ip	src_port	dst_ip	dst_port	protocol_type	service	src_byte	wrong_fragment	flag	logged_in	...	
899380817	172.16.xx.xx	1024	135.13.yy.yy	25	0.62	25	802	0	0.83	0	...	
899380819	172.16.xx.xx	53	192.168.yy.yy	53	0.5	53	31	0	0.83	0	...	
899380819	0	0	0	0	0.002	100001	0	0	0.83	0	...	
899380819	0	0	0	0	0.002	100002	0	0	.083	0	...	
899380822	172.16.xx.xx	1026	206.79.yy.yy	80	0.62	80	159	0	0.44	0	...	
.	.	.	.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	.	.	.	
899388931	10.20.xx.xx	7834	172.16.yy.yy	804	0.62	804	0	0	0.44	0	...	
899388931	10.20.xx.xx	54372	172.16.yy.yy	78	0.62	78	0	0	0.44	0	...	

connection vectors (instances)

FIGURE 3. A sample of a dataset exported from OptiFilter.

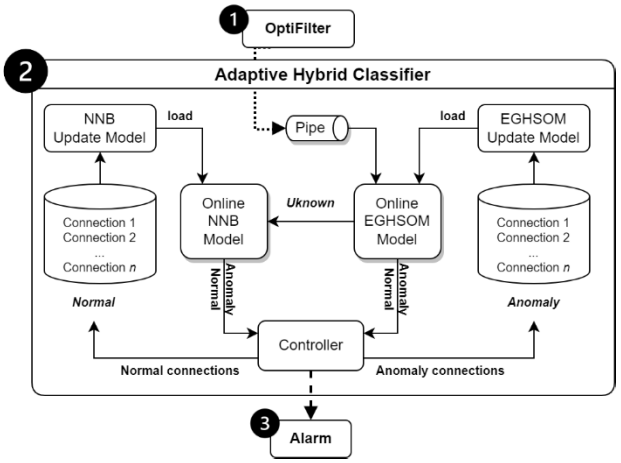

FIGURE 4. The internal architecture of the adaptive hybrid classifier.

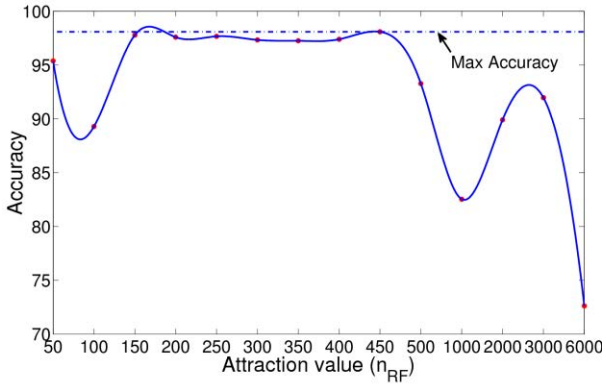

FIGURE 5. The trade-off between n_{RF} and the accuracy of EGHSOM.

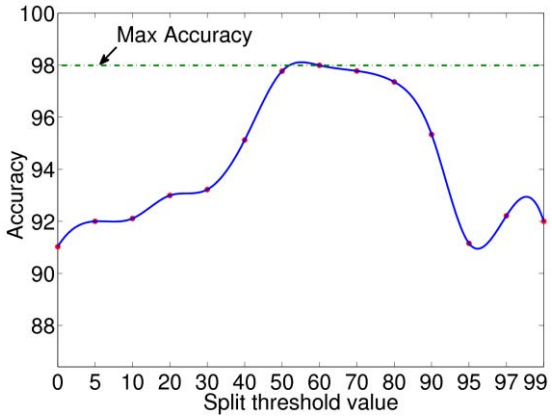

FIGURE 6. The trade-off between ζ and the accuracy of EGHSOM.

and NNB models and manages them properly. Specifically, the controller gathers the detected anomalous and normal connections in different datasets, which are subsequently used to update the models. Moreover, it sends the detected connections to the alarm component to show a proper alert message. More details about the functionality of each model are presented in the next subsections.

1) ONLINE EGHSOM MODEL

In our previous published work [57], we enhanced the original GHSOM model to be more effective and accurate in detecting unknown connections. The enhanced model, called the EGHSOM, includes a meaningful initialization process instead of random initialization, a novel splitting technique to stabilize the growth topology, a method to remedy the final BMUs, and a classification-confidence margin threshold to uncover unknown connections. In this paper, we will further examine the trade-off between the accuracy and the *attraction limit* n_{RF} of merging weak BMUs, as well as the *splitting threshold* ζ. More details about the attraction limit and the splitting threshold are available in [57]. Moreover, we will explain the classification-confidence threshold in the Section V model evaluation. According to the merge conditions in [57], Figure 5 shows the trade-off between the attraction limit and the accuracy of the EGHSOM model using three different datasets, which contain heterogenous network traffic from offline and online networks and hosts evens, as well as injected attacks. These datasets are NSL-KDD dataset and two datasets collected from synthetic networks and real industrial networks.

The trade-off shows that the attraction limit of each BMU must at least be between 200 and 450 input vectors to achieve maximum accuracy.

In addition, we investigated the trade-off between the splitting threshold and the accuracy in the EGHSOM model, as shown in Figure 6.

Varying the splitting threshold value helps examine the heterogeneity of each node to provide better and higher resolution, stable growth, and robust hierarchical topology. Hence, it is a very helpful factor to improve the detection rate and the accuracy of the adaptive classifier. The figure shows clearly that the best splitting threshold to achieve a high

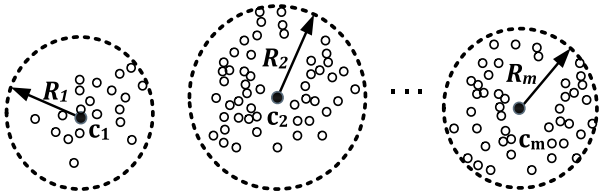

FIGURE 7. Representation of the NNB model.

accuracy range between 45 and 95. Hereby, the first value, i.e., zero, is the accuracy without any splitting technique. That is, it has achieved the poorest accuracy. The final EGHSOM model is therefore represented as

$$(C, wC, Labels, \delta) \tag{13}$$

where δ is the classification-confidence margin [49].

2) ONLINE NNB MODEL

Defining an NNB model is considered one of the known challenges in network management. Several contributions have provided different solutions to tackling this challenge, such as a tool to describe only internal network behavior [58] or a criterion for measuring the statistical state of hosts in the network [59]. However, due to concept drift and the fact that data flows are non-stationary [60], this challenge still exists. Therefore, the proposed EGHSOM is utilized to define a reasonable NNB model.

To achieve this goal, we gathered data flows from an isolated network environment to guarantee suspicion-free connection vectors (i.e., only "normal" connections). Then, we trained the EGHSOM using these connections so that the final BMUs and their receptive fields became assigned with only "normal" labels. Consequently, let the maximum distance of a BMU c_i to its RF_i be

$$R_i := max\{d(x_j, w_{c_i}) | x_j \in RF_i, c_i \in C\}, \quad i = 1, \ldots, m, \tag{14}$$

where m is the final number of BMUs in the EGHSOM model. Geometrically, this can be represented as a circle around the BMU. Thus, if we determine the maximum distance R_i for each BMU c_i in the final model (as shown in Figure 7), we can define a reasonable NNB model based on these distances.

Note that the small circles represent input vectors, and the big, dashed circles are the obtained maximum radius around the BMUs. Accordingly, the NNB model based on the EGHSOM training process can be defined as

$$(C, w_C, R_C), \tag{15}$$

where R_C is the distance set of the final BMUs. The goal of this model is to classify unknown connections as either "normal" or "anomaly". Note that, as mentioned before, the EGHSOM model can classify connections as "normal", "anomaly", or "unknown" [57]. Hence, the NNB is a very important component that classifies unknown connections as "normal" and "anomaly", based Algorithm 2. The algorithm is very simple and effective at the same time. The first step of the algorithm is to consider all connections detected and labeled as "unknown". In the second step, we measure the distance of each unknown connection to the BMU in the normal NNB model. Based on equation 14 and figure 7, each BMU in the NNB has a radius, so if the measured distance in the second step is less than the radius of any BMU in the NNB, then it will be classified as Normal because it belongs to this cluster, otherwise, the unknown connection will be classified as "unknown anomaly" and it should be further analyzed.

In the following subsections, we will describe the update procedures used in both EGHSOM and NNB models. These procedures are vital to ensure that the outcome of the Opti-Filter is dynamically and continuously optimized.

3) EGHSOM UPDATE MODEL

The update procedure of EGHSOM uses connections from the anomaly database and measures the *selection percentage* (η) of each BMU, where η determines which BMU was active, and which one was not, according to the online detection process. The use of the *selection percentage* (η) is necessary to expand the topology of the EGHSOM during online detection in which the growth of the original EGHSOM model is kept within plausible limits. We need to mention that, when our EGHSOM detects an anomalous connection, it sends this connection and the BMU that has detected it to the controller in the following form (*connection, label, BMU*).

Algorithm 2 Labeling Unknown Connections Using the NNB Model

Require: NNB Model (C, w_c, R_c)
  while true **do** *continuous loop*
    input ← unknown connection $\hat{x}(t)$
    while $i < m$ **do** *m = number of BMUs*
      $\hat{d} = ||\hat{x}(t) - w_i||$ *distance to each BMU*
      if $\hat{d} \leq R_i$ **then**
        $\hat{x}(t)$ is *normal connection*
        Write $\hat{x}(t)$ to Normal Dataset
        Output ← $\hat{x}(t)$; *Break*;
      else
        $i++$;
      end if
    end while
    Output ← $\hat{x}(t)$ is *unknown anamoly*
  end while

Accordingly, the controller decides to store the connection in the anomaly database and forwards it to the alarm component. Therefore, one BMU can appear several times in the anomaly database. Let the number of BMUs in the database be N such that $Q := \{bmu_1, bmu_2 \ldots, bmu_N\}$ and $Q \subset C$. Thus, the *selection percentage* of each BMU c_i can

be determined as

$$\eta_i := \frac{|\{bmu_i | bmu_i \in Q\}|}{|Q|}, \quad i = 1, .., N, \tag{16}$$

where $|Q|$ is the total number of connections in the anomaly database. Each BMU with a high selection percentage value should be further analyzed, because the large number of mapped instances onto one BMU can be used to improve detection model accuracy and precision. Therefore, a bmu_i will be divided into two neurons bmu_{i1} and bmu_{i2} if it satisfies the condition $\eta_i > 0.95$. This value has been selected based on several practical tests of the EGHSOM with a continuous tuning process. That means if the BMU has been selected by 95% of the connection vectors, it should be split into two BMUs to maintain plausible growth and accurate detection. In this regard, all connections in the anomaly database detected by the bmu_i are considered as its RF_i. Initializing the weight vectors of the two BMUs bmu_{i1} and bmu_{i2} should be performed using the initialization process presented in [57]. Input vectors in the RF_i will be reassigned on both new BMUs according to the minimal distance measure. As a result,

$$\{bmu_i, RF_i\} \rightarrow \{bmu_{i1}, RF_{i1}\}, \{bmu_{i2}, RF_{i2}\} \tag{17}$$

The next step is to update the classification-confidence threshold $\delta = [d_{min}, d_{max}]$. Let δ_Q be the threshold margin of all BMUs in Q such that $\delta_Q = \left[\hat{d}_{min}, \hat{d}_{max}\right]$. Therefore, the new threshold margin $\acute{\delta} = \left[\acute{d}_{min}, \acute{d}_{max}\right]$ is determined based on the following:

$$\acute{d}_{min} = \begin{cases} \hat{d}_{min} & \hat{d}_{min} \leq d_{min} \\ d_{min} & else, \end{cases}$$

and

$$\acute{d}_{max} = \begin{cases} \hat{d}_{max} & \hat{d}_{max} \geq d_{max} \\ d_{max} & else. \end{cases} \tag{18}$$

After examining the BMUs in Q and updating the margin of the classification-confidence threshold, the final step is to update the BMUs using equation 4, where the RF of each BMU is considered as its input and $h_{ci}(t)$ is an exponentially decreasing function that is used in the same way as training the GHSOM by decreasing the neighborhood area around the unit during the update process, such that

$$h_{ci}(t) = \alpha_0 \cdot e^{-\frac{I}{|Q|}}. \tag{19}$$

Hereby, α_0 is a constant and $0 < \alpha_0 < 1$, I is the total number of input vectors in the RF_i of the $BMUi$, and $|Q|$ is the total number of instances in the anomaly database. Accordingly, the update model loads the new EGHSOM model by replacing it with the current EGHSOM model as

$$(C, w_C, Labels, \delta) \rightarrow (C', w_{C'}, Labels, \delta'). \tag{20}$$

The flowchart in Figure 8 illustrates the steps in the EGHSOM updates model.

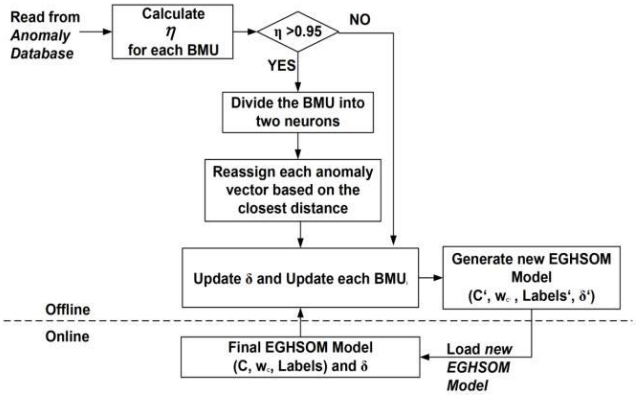

FIGURE 8. Flowchart of EGHSOM update model steps.

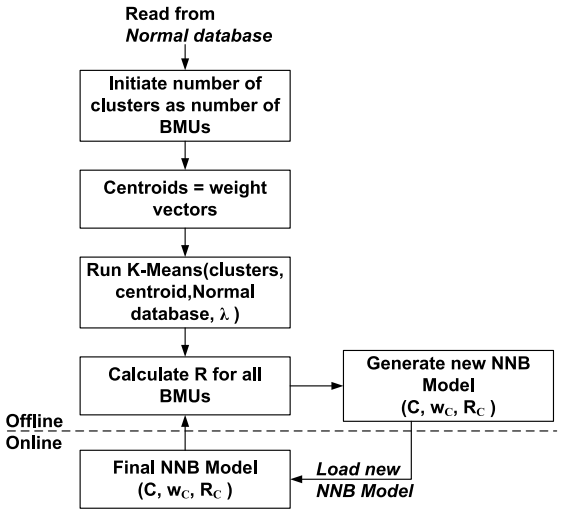

FIGURE 9. Flowchart of NNB update model steps.

4) NNB UPDATE MODEL

The NNB update procedure considers only detected normal connections from the online EGHSOM model and the online NNB model. The normality of the network changes constantly because the data stream is made up of non-stationary data, which are affected by concept drift [60]. Figure 9 shows a flowchart of the steps in the NNB update model.

Due to the constant changes in the normality of the network, the NNB model should be updated continuously. All connections in the normal database will be used to update the state of each BMU in the NNB model (C, w_C, R_C) using the k-means clustering algorithm [61], [62], [63]. In other words, BMUs that were not active in the online detection could be modified during the update procedure, which in turn enhances the NNB model and makes it more adaptive and homogeneous with the new normal state of the network.

Based on Figure 8, the weight vector of each BMU can be considered as a cluster centroid in k-means. Accordingly, to update the NNB model, the k-means algorithm should be performed as

$$K - means(m, centroids, Normal\_database, \lambda), \tag{21}$$

where λ is the number of iterations. After the algorithm is converged, the radius of each cluster is determined as described in equation 13 and, hence, the update model loads the new NNB model as

$$(C, w_C, R_C) \rightarrow (C', w_{C'}, R_{C'}) \tag{22}$$

V. MODEL EVALUATION

In the performance evaluation, the proposed TIDM was evaluated in both its offline and online operational modes. Several benchmark datasets have been used in these evaluations with predefined performance metrics to assess the practicality of the proposed solution.

A. DATASET FOR EVALUATION

We used data flows from four different datasets for evaluation. The first one is the benchmark offline dump data from DARPA (D1) [64]. The second is synthetic data from our simulated network, which is called "*Artificial Dataset*" (D2). We installed a simulated network at the university campus that simulates internal and external traffic on virtual machines. On the internal network, we set up five GNU/Linux server systems, two Windows Internet Information and Exchange servers, and two Windows domain controllers for a total of 10 windows running Windows 7 and XP. On the external network, we set up five virtual machines, each providing a dedicated service (i.e., HTTP, DNS, and SMTP). The virtual machines were assigned 1024 IP addresses each, ranging over the whole IPv4 address space. This allowed us to simulate connections to the internet with a wide range of different IP addresses. In this test network, we used the applications Metasploit, Nexpose, OpenVAS, and other attack scenarios like ping around or DoS to generate new uncovered attacks (or anomalous traffic).

Accordingly, network data were captured using tcpdump at the virtual bridge interface of the internal physical server. For our tests, we selected 17 of the most common services that should be present in the network traffic dump; these were *ftp, ssh, telnet, smtp, smb, nfs, xmpp, http, ntp, dhcp, syslog, snmp, rdp, IMAP, pop3*, and *rsync*. Note that, for a complete day, we injected different attacks into our simulated network. We called this third data flow "*Artificial Dataset+Anomaly*" (D3).The fourth trace is of real-time data from a large-scale firm computer network operating on 1–10 GB; we called it "*RealSet*" (D4) [65].

B. EVALUATION IN THE OFFLINE OPERATIONAL MODE

In this section, both OptiFilter and the Adaptive hybrid classifier will be evaluated offline using different datasets which are explained in the previous sub-section (V.A.).

1) OPTIFILTER EVALUATION

In this section, the first part of the proposed TIDM is evaluated in its offline mode. Table 2 summarizes the configuration parameters of this part. In this mode, the OptiFilter was evaluated using the datasets provided in Table 2 and

TABLE 2. Configuration of optifilter for offline evaluation.

Configuration Parameter	Value
Data flows used	340 MB of Benchmark DARPA (D1), 1.5 GB of Artificial Dataset (D2), 1.5 GB of Artificial Dataset+Anomaly (D3), and 3.5 GB of RealSet (D4)
Queue concept parameters	Queue capacity = 3 windows Time slot window w = 5 sec Backlog = 1000 connections
Performance metrics	1. Number of processed packets. 2. Total processing time per window (milliseconds). 3. Number of constructed connections. 4. Number of packet drops.

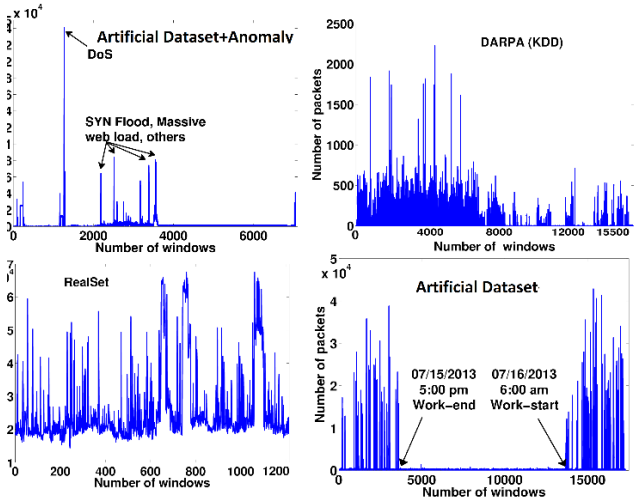

FIGURE 10. Number of processed packets per window.

the performance metrics of this evaluation are discussed individually.

The first performance metric monitors the total number of packets that have been processed by each window in the queue. Further details about the queue concept are available in our work [41]. This metric indicates the adequacy and performance of the OptiFilter by processing packets every 5 seconds. In other words, the OptiFilter reads a certain number of captured packets every w seconds (e.g., $w = 5$) and inserts them in the current window to be analyzed and correlated with the host events. Therefore, the metric will show if the total number of captured packets can be effectively processed in the current window without any failure or limitation. Figure 10 demonstrates the result for the selected data flows.

Note that each data flow has a different size with different packet flows per second, which normally leads to different numbers of windows. This results in uneven curves for each data flow, as shown in Figure 10.

For instance, it is possible that only one connection was established in the first 5 seconds, which would mean the number of packets in the current window is small. In contrast, it is possible that hundreds of connections were established in

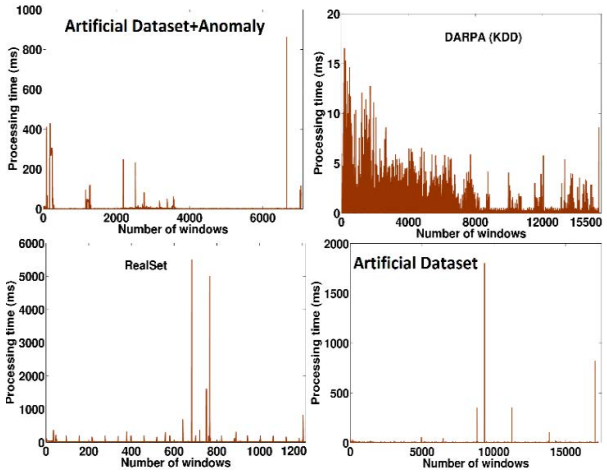

FIGURE 11. Processing time per window.

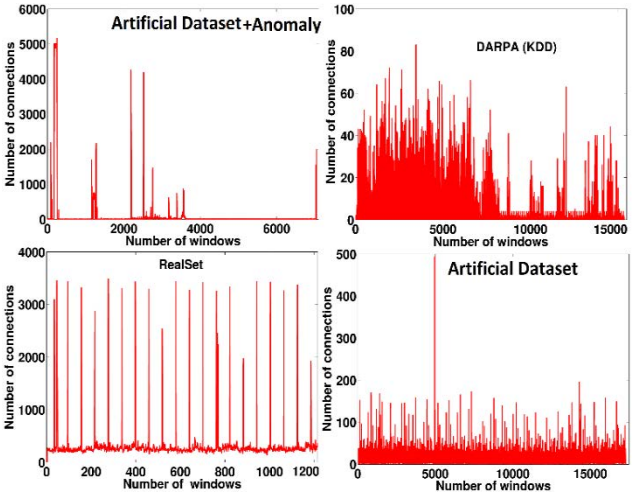

FIGURE 12. Number of constructed connections per window.

the first 5 seconds, which would mean the number of packets in the current window is large. This implies that a packet flows every 5 seconds is not consistent; hence, the number of processed packets in each window will be different.

The OptiFilter can handle a huge number of packets, as shown in Figure 10, without any failure or limitation. For instance, the 3.5 GB data flow of the dataset "*RealSet*" was processed within 1200 windows in the queue, whereas more than 15500 windows were required to process the data flow from the dataset D1. Moreover, some irregular curves, such as the one related to the data flow of dataset D3, provide further information about the nature of the processed packets. This data flow in D3 contains some types of attacks, which can be recognized in Figure 10. We verified if these salient peaks belong to any attack class, and all of them were indeed attacks.

The second monitored metric relates to the total time required for each window to process the corresponding packets and to construct the connection vectors accordingly. This performance metric indicates the superiority of the OptiFilter in managing massive data flows within reasonable timeframes, as shown in Figure 11.

Intuitively, whenever the number of packets increases, the required processing time should also increase. In general, Figure 11 confirms this assumption. The salient peaks on some results, such as in the D2 dataset data flow, are considered to be glitches, but they have no influence on the overall performance of the OptiFilter. Other peaks could be related to programming issues such as the salient peak in the data flow of dataset D3.

The third metric involves observing the number of constructed connection vectors for each window. Figure 12 shows the final constructed connection vectors of each window, which should be proportional to the number of processed packets.

The last metric is *packet dropped*, which is very important in computer network management. We carefully monitored the number of dropped packets for each window and then

TABLE 3. Final results of evaluating the optifilter in offline mode.

metric (see Table II)	DARPA		Artificial Dataset+ Anomaly		Artificial Dataset		Real-Set	
	Min	Max	Min	Max	Min	Max	Min	Max
1	0	2226	0	24120	0	42838	11592	67445
2	0.012	16.52	0.004	860	0.011	1800	0.017	5000
3	0	83	0	5163	0	493	0	3487
4	0	1	0	142	0	21	0	17

calibrated the OptiFilter accordingly. Regarding data flow from the D3 dataset, the number of packet drops was very small. At the start, the OptiFilter dropped around 150 packets due to the cold start—that is, the OptiFilter was still in its start-up phase.

Moreover, some IPv6 packets appear at the beginning and the OptiFilter dropped them immediately. In contrast, the OptiFilter regularly dropped around 20 packets per window while processing the data flow in the D4 dataset. This refers to the IPv6 packets and some packets belonging to other dedicated services that are not included in the OptiFilter's configuration. According to the data flow of the D2 dataset, the OptiFilter dropped all IPv6 packets and, because the data flows were synthetically generated, the same number of packet drops from each window was repeated until the last window. Table 3 summarizes all the metrics for better illustration.

2) ADAPTIVE HYBRID CLASSIFIER EVALUATION

The second part of the offline evaluation focuses on the EGHSOM classifier model. It examines the following performance metrics: the accuracy and the false positive rate. The best-known method to evaluate the IDS classifier is the cross-validation technique.

Cross-validation is an accuracy estimation technique that evaluates the precision of the classifier model by building a confusion matrix for different datasets. Accordingly, for evaluating the adaptive classifier, 10-fold cross-validation

TABLE 4. Datasets for evaluating the classifier in offline mode.

Dataset	# of instances	Normal instances	Anomaly instances
NSL-KDD [67]	125969	97252	28717
Artificial Dataset+Anomaly	150239	130128	20111
RealSet	186414	186414	0

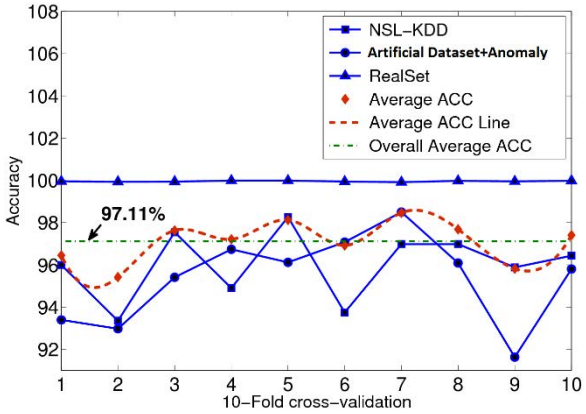

FIGURE 13. Final accuracy results using the offline 10-fold cross-validation.

was selected because it has been shown that 10 levels are about right for obtaining the best estimate of error and 10-fold cross-validation is a widely used technique for evaluating the classifier model [66]. Several labeled datasets, which vary in size and source, were prepared for cross-validation. The datasets prepared are shown in Table 4.

Additionally, the final accuracy results of the 10-fold cross-validation are illustrated in Figure 13.

The cross-validation technique divides the dataset into different parts. For instance, if the classifier is evaluated by a 5-fold cross-validation technique then the total number of instances M in the dataset would be divided into five sets, with each set having a size $M/5$ in such a way that the classifier can be trained using the first four sets and evaluated using the fifth set. This process would then be repeated five times, the mean accuracy being taken as the final performance measure of the classifier. Mean accuracy is simply the mean value of all computed accuracy in each fold. Hence, for 5-fold cross-validation, we have 5 different accuracy measures, so we take the mean value of these 5 performance measures, and this is considered then our mean accuracy. While False Positive Rate (FPR) is the number of normal connections seen by the TIDM as an anomaly.

Based on Figure 13, the mean accuracy of the total 10-fold approach for all datasets in Table 3 reached 97.11, and the mean FPR= 0.01417. As mentioned above, the cross-validation technique evaluates the classifier using the untrained set; hence, we utilized this option to verify whether our EGHSOM model can detect new connections as being ''unknown''. Figure 14 shows the number of connections detected as unknown during the cross-validation.

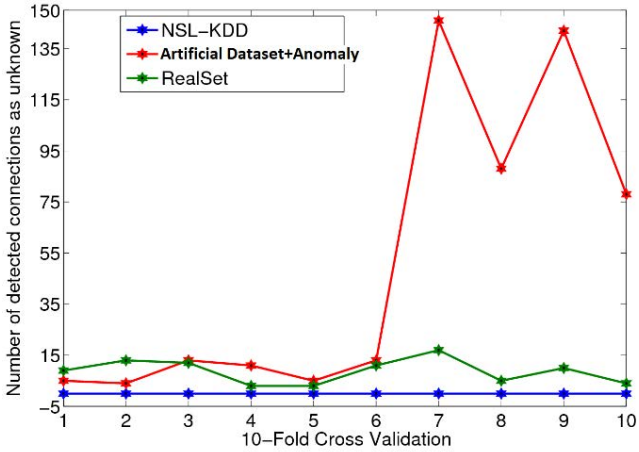

FIGURE 14. Total number of connections detected as unknown during the offline 10-fold cross-validation.

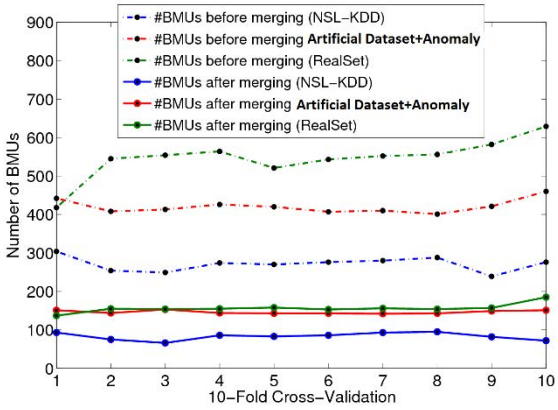

FIGURE 15. Number of BMUs before and after merging during the offline 10-fold cross-validation.

It is very possible that a test subset (fold) contains anomalous connections that have not been present in the training subset, in which case the classifier will classify them as ''unknown''. The next monitored metric during the cross-validation is the number of BMUs in the final EGHSOM before and after performing the proposed merging process [57], as shown in Figure 15.

Notably, the EGHSOM training could reduce the total number of final BMUs significantly, which, in turn, would improve the overall TIDM performance.

C. EVALUATION IN THE ONLINE OPERATIONAL MODE

We installed the TIDM on a real-time 1–10 GB firm computer network and evaluated both parts online. In this mode, the TIDM was executed for almost 2 hours (nonstop) and processed around 920 windows in the queue (window w = 5 sec). Figure 16 demonstrates the nature of the aggregated data flow within these 2 hours.

The same performance metrics in the offline operational mode for the OptiFilter were also used in the online operational mode. Figure 17 summarizes these metrics.

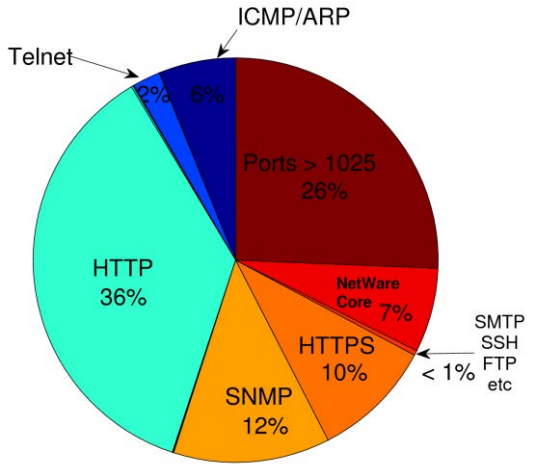

FIGURE 16. Most used services in the real-time data flow.

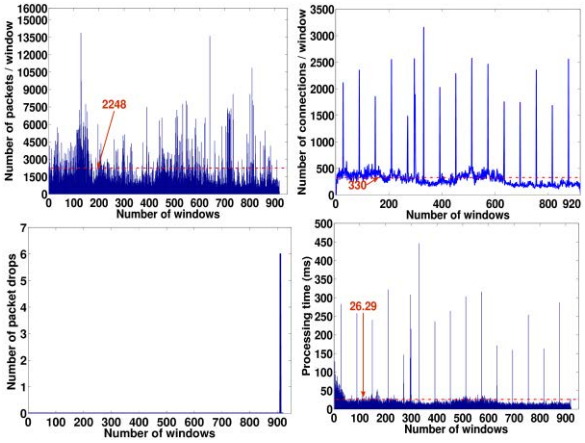

FIGURE 17. OptiFilter performance in real-time.

The proposed TIDM processed more than 13500 packets per window in the dynamic queue and accordingly constructed more than 3000 connections in less than 5 seconds, without any packet drops.

On the other hand, the adaptive classifier shows very superior results in the online operational mode. In addition to monitoring accuracy, detection rate, and false positive rate, the update models for EGHSOM and NNB were also monitored to check the adaptivity of the TIDM in real-time. The update model of EGHSOM performed an update after 30 minutes, provided that the number of detected anomalies was greater than 100 connections. Moreover, the update for NNB was configured to perform an update after 25 minutes, provided that the number detected as ''normal'' was greater than 5000 connections. Accordingly, we checked if the TIDM is able to adapt the classification-confidence margin and the radiuses of the NNB model via massive data flows.

Firstly, we used the classification-confidence margin and some postulates to construct the confusion matrix in online operational mode for the EGHSOM, as shown in Figure 18. These postulates were:

		Predicted Class		
		Anomaly	Normal	Total
Actual Class	Anomaly	TP=3	FN=0	3
	Normal	FP=57	TN=303544	303601
	Total	60	303544	303604
Performance metrics				
Sensitivity		TPR, DR, Recall		1
1- Specificity		FPR		0.0001877
Accuracy		ACC		0.999812

FIGURE 18. Confusion matrix and the corresponding performance metrics.

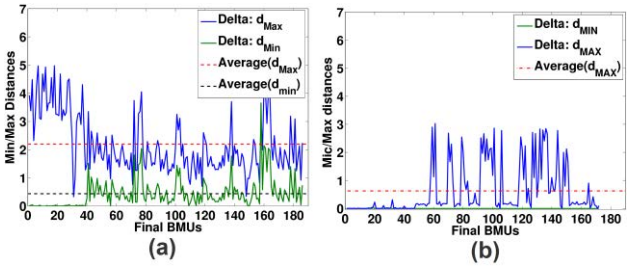

FIGURE 19. (a) Classification-confidence margin used at the beginning; (b) adapted classification-confidence margin at the end.

- Connections between internal IPs are considered normal.
- All connections from internal to external IPs are considered normal.
- Connections from external to internal IPs have been assigned with the same label as is given by the proposed TIDM.

For instance, the false positive (*FP*) value in the matrix is actually the number of connections that have been detected as anomalies but were originally classed as ''normal''. Note that *TP* is the true positive, *TN* is the true negative, and *FN* is the false negative. Moreover, the performance metrics are *TPR* (the detection rate), *FPR* (the false positive rate), and *ACC* (the accuracy). They were obtained as follows.

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + TN} \tag{23}$$

$$ACC = \frac{TN + TP}{TP + TN + FP + FN} \tag{24}$$

Figure 19 shows the classification-confidence at the beginning of executing the TIDM and at the end. It emphasizes that the TIDM could optimally adapt the margin, in real-time, throughout the online classification.

As mentioned previously, the idea of the classification-confidence margin was derived from the mean quantization error of each BMU. Thus, the TIDM is considered effective if the mean quantization error (i.e., the margin) decreases during the online classification. In this regard, Figure 21 shows that the minimum and maximum boundaries of the classification-confidence margin did indeed decrease gradually.

Another major property of the TIDM is the ability to classify new suspicious connections as ''unknown''.

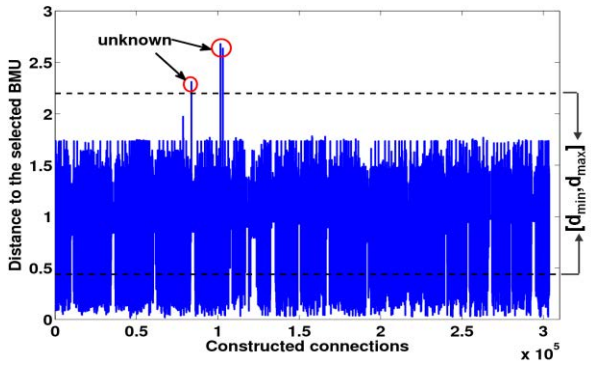

FIGURE 20. Distances of all classified connections in real time.

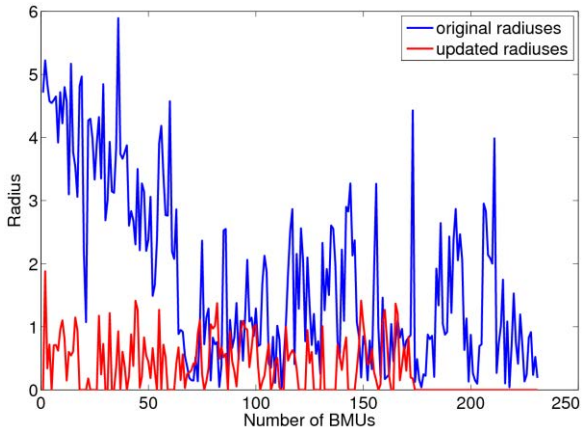

FIGURE 21. Radius adaption by the NNB model in real time.

Figure 20 shows the classification-confidence margin used in the online operational mode and the distance of each classified connection revealed by the TIDM.

The Euclidean distance of each constructed connection to the EGHSOM model was calculated and examined if it fell inside or outside the margin. Note that if the distance of a new constructed connection is smaller than the minimum margin boundary then this connection is considered a known connection, because it is very close to a BMU in the EGHSOM model. On the contrary, if the distance falls outside the maximum margin boundary, then it is considered ''unknown''. Therefore, the EGHSOM model can effectively detect suspicious connections and classify them as ''unknown'', as Figure 20 illustrates.

Regarding the NNB model, the radiuses were also adapted during the online operational mode using the k-means algorithm, which achieved smaller radiuses at the end of the execution. This implies that the proposed TIDM adapted the NNB model so that each BMU has a minimized mean quantization error and hence a robust cluster form. The adaption of the radiuses can be seen in Figure 21.

As a result, the TIDM provided very promising results in online operational mode. It can process a massive data flow within seconds without dropping any packets. In addition, all packets were processed within a reasonable time. Moreover, it can achieve maximum accuracy and a minimum false alarm rate, and can classify connections as ''normal'', ''anomaly'', or ''unknown''. Here, ''unknown'' connections refer to suspicious connections. Finally, the TIDM is effectively able to adapt its classification models and other parameters during the online operational mode.

VI. MODEL PERFORMANCE

It is also important to evaluate other necessary performance metrics of the model. First, we compare the proposed TIDM with other competitive models. Second, based on our expertise, and by referring to some industrial standards [68], we have chosen the following metrics: user satisfaction/Apdex scores, average response time, and scalability.

A. COMPARISON STUDY WITH OTHER MODELS

In this section, the proposed TIDM model is compared against other intrusion detection models, which are summarized by Ozkan-Okay et al. [69]. It is worth mentioning that these models including TIDM, have different configurations and datasets. Therefore, the comparison will focus on the performance metrics for all models. These metrics are Detection Rate (DR), False Positive Rate (FP), and Accuracy (ACC). Table 5 shows the comparison result.

TABLE 5. Summary of intrusion detection comparison.

Paper (year)	Dataset	DR	FP	ACC
Wattanapongsakorn et al. [70] (2012)	LAN traffic	99.8	--	--
Kumar et al. [71] (2014)	LAN traffic	99.6	0.018	97.5
Qassim et al. [72] (2016)	KDD	--	--	93-99
Qureshi et al. [73] (2018)	NSL-KDD	95.6	0.0128	94.5
Mazini et al. [74] (2019)	NSL-KDD, ISCXIDS12	99.6	0.01	98.9
Meftah et al. [75] (2019)	UNSW-NB 15	--	--	86.04
Devan and Khare [76] (2020)	NSL-KDD	97	0.98	97.6
Bedi et al. [77] (2021)	NSL-KDD CIDDS-001	97	--	89
Our Model – TIDM (2022)	NSL-KDD, Synthetic, Real Firm Traffic (online)	**99.9**	**0.0002**	**99.9**

As clearly shown from the table, used datasets for the evaluation are different, but obviously the TIDM model has been evaluated by several datasets. The salient difference here is the evaluation on a real industrial firm network in the online operational mode (that means during the firm normal business day). In addition, some articles from the table evaluated known machine learning methods against the selected

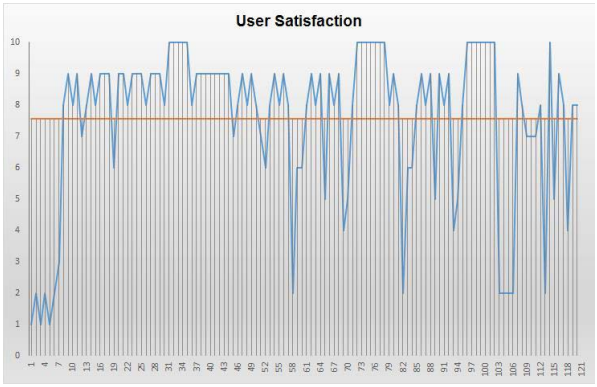

FIGURE 22. User satisfaction with using TIDM.

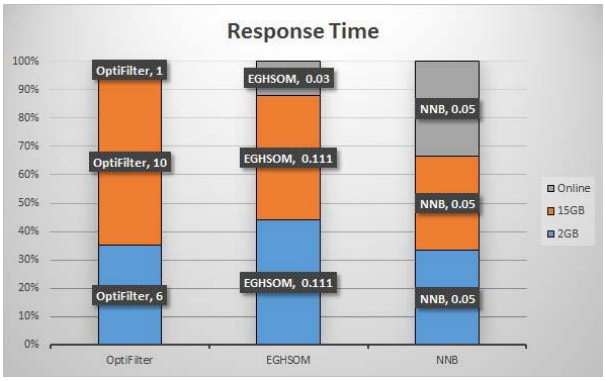

FIGURE 23. Response time for all components.

dataset. For instance, Meftah et al. [75] evaluated Logistic Regression and SVM using the UNSW-NB 15 dataset, hence, no new or novel proposed model on their work is presented. Moreover, some articles did not include certain performance metric, such as Qassim et al. [72]. They covered only the performance metric accuracy and did not cover other both metrics. In addition, we have selected the best performance metric value in the comparison, for example, if the author of certain article in the table conducted several tests and reported several accuracies, we considered the best resulted accuracy of that article in the comparison. As a result, all selected articles in table 5 are competitive, but the proposed TIDM model delivers the best performance. Moreover, it's the only model capable of updating itself continuously in real-time.

B. USER SATISFACTION/APDEX SCORES

Apdex is an industry standard to measure users' satisfaction with the response time of web applications and services. It helps to reveal how satisfied users are with an application. We distributed a survey to about 120 users and asked them to score their satisfaction with using the TIDM framework from 1 (''not satisfied'') to 10 (''very satisfied''). Figure 22 shows the results of the survey.

As is obvious from Figure 22, most of the users were very satisfied with using the application: the average value of 7.55 gives a good indication of the degree of user satisfaction. In general, the application still needs improvements, and these will be considered during future work.

C. AVERAGE RESPONSE TIME AND SCALABILITY

This metric shows the response time of any request made to the model. That means the response time of the model from the time a data item is captured until a result is displayed as a classification of the generated instance.

We evaluated this metric using three parameters: namely, small offline traffic up to 2 GB, large offline traffic up to 15 GB, and online traffic over 2 hours. Figure 23 displays the response time, in seconds, based on these scenarios.

The OptiFilter component takes more time than the other components (e.g., 10 seconds for 15 GB) because it needs to reprocess the flows and events and then generate a dataset based on the sliding window mechanism, which in turn depends on the size of the input and whether it is offline or online. However, the online response time is still very reasonable.

Regarding scalability, the application should be suitable for any network, and it works properly even if new network devices are installed or the amount of collected traffic changes. Indeed, the previous test showed that the application will work with networks beyond our simulated one and with different collected data sizes.

VII. CONCLUSION AND FUTURE WORK

In this paper, an adaptive real-time IDS TIDM has been introduced as a means to manage the data flows in computer networks and reveal unknown connections in the online operational mode. The TIDM consists of two parts. The first part namely, the OptiFilter aggregates continuous data flows and hosts' events, processes them, and constructs connection vectors based on the selected features set. The second part known as the adaptive classifier uncovers anomalous and unknown connections. Moreover, the latter is able to further classify unknown connections and to adapt itself to the actual network behavior by using novel update models. Specifically, the adaptive classifier utilizes an EGHSOM model to classify the constructed connections and an NNB model to further classify the unknown connections. It uses the detected anomalous connections to update the EGHSOM model and the detected normal connections to update the NNB model. We evaluated the TIDM in the offline operational mode using various data flows and in the online operational mode in a real-time firm 1–10 GB computer network. The TIDM achieved high accuracy and very low false positive rates in both operational modes. Moreover, it can handle massive data flows within reasonable timeframes and uncover anomalous and unknown connections with precision. The TIDM can adapt the classifier models and other parameters continuously in real-time. However, the proposed model does not consider IPv6, the OptiFilter occasionally needs some manual configuration, and the alarm should ideally be more interactive.

Finally, the application has been attractive to users, who have expressed their satisfaction. It has also been recommended by some of them, with further improvement, to become an industrial application. In our future work, we will tune the TIDM to handle IPv6 packets, and will develop a dynamic labeling technique.

REFERENCES

[1] S. M. Bellovin and R. Bush, "Configuration management and security," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 3, pp. 268–274, Apr. 2009.
[2] P. Skeffington and R. Ward, "Network computer management," in *Proc. 6th Annu. Comput. Secur. Appl. Conf.*, Tucson, AZ, USA, 1990.
[3] G. Y. Keung, B. Li, and Q. Zhang, "The intrusion detection in mobile sensor network," *IEEE/ACM Trans. Netw.*, vol. 20, no. 4, pp. 1152–1161, Aug. 2012.
[4] L. Kufel, "Security event monitoring in a distributed systems environment," *IEEE Security Privacy*, vol. 11, no. 1, pp. 36–43, Jan. 2013.
[5] H. Duan and J. Wu, "Security management for large computer networks," in *Proc. 5th Asia–Pacific Conf. 4th Optoelectronics Commun. Conf. Commun.*, Beijing, 1999, pp. 1208–1213.
[6] F. Majeed, M. S. Mahmood, and M. Iqbal, "Efficient data streams processing in the real time data warehouse," in *Proc. 3rd Int. Conf. Comput. Sci. Inf. Technol.*, Chengdu, China, Jul. 2010, pp. 57–61.
[7] V. Gulisano, R. Jimenez-Peris, M. Patino-Martinez, C. Soriente, and P. Valduriez, "StreamCloud: An elastic and scalable data streaming system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2351–2365, Dec. 2012.
[8] L. Vokorokos, P. Fanfara, J. Radusovsky, and P. Poor, "Sophisticated honeypot mechanism—The autonomous hybrid solution for enhancing computer system security," in *Proc. IEEE 11th Int. Symp. Appl. Mach. Intell. Inform. (SAMI)*, Herl'any, Slovakia, Jan. 2013, pp. 41–46.
[9] M. Missbach, T. Staerk, C. Gardiner, J. McCloud, R. Madl, M. Tempes, and G. Anderson, *SAP on the Cloud* (Management for Professionals), Berlin, Germany: Springer, 2016.
[10] J. Bayuk and A. Mostashari, "Measuring systems security," *Syst. Eng.*, vol. 16, no. 1, pp. 1–14, 2013.
[11] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, pp. 1–18, Feb. 2020.
[12] B. Sun, X. Shan, K. Wu, and Y. Xiao, "Anomaly detection based secure in-network aggregation for wireless sensor networks," *IEEE Syst. J.*, vol. 7, no. 1, pp. 13–25, Mar. 2013.
[13] Y.-L. Hu, W.-B. Su, L.-Y. Wu, Y. Huang, and S.-Y. Kuo, "Design of event-based intrusion detection system on OpenFlow network," in *Proc. 43rd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Budapest, Hungary, Jun. 2013, pp. 1–2.
[14] Y. J. Lee, Y. R. Yeh, and Y. C. F. Wang, "Anomaly detection via online oversampling principal component analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1460–1470, Jul. 2013.
[15] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and A. Vasilakos, "An effective network traffic classification method with unknown flow detection," *IEEE Trans. Netw. Service Manage.*, vol. 10, no. 2, pp. 133–146, Mar. 2013.
[16] A. Hofmann and B. Sick, "Online intrusion alert aggregation with generative data stream modeling," *IEEE Trans. Dependable Secure Computing*, vol. 8, no. 2, pp. 282–294, Mar. 2011.
[17] S. LaValle, E. Lesser, R. Shockley, M. S. Hopkins, N. Kruschwitz, and S. LaValle, "Big data, analytics and the path from insights to value," *MIT Sloan Manag. Rev.*, vol. 52, no. 2, pp. 21–31, 2011.
[18] A. Jacobs, "The pathologies of big data," *Commun. ACM*, vol. 52, no. 8, pp. 36–44, Aug. 2009.
[19] M. Guarascio, N. Cassavia, F. S. Pisani, and G. Manco, "Boosting cyber-threat intelligence via collaborative intrusion detection," *Future Gener. Comput. Syst.*, vol. 135, pp. 30–43, Oct. 2022.
[20] W.-S. Choi, S.-Y. Lee, and S.-G. Choi, "Implementation and design of a zero-day intrusion detection and response system for responding to network security blind spots," *Mobile Inf. Syst.*, vol. 2022, pp. 1–13, Apr. 2022.
[21] N. Ye, S. M. Emran, Q. Chen, and S. Vilbert, "Multivariate statistical analysis of audit trails for host-based intrusion detection," *IEEE Trans. Comput.*, vol. 51, no. 7, pp. 810–820, Jul. 2002.
[22] N. B. S. Amor and Z. Elouedi, "Naive Bayes vs decision trees in intrusion detection systems," in *Proc. ACM Symp. Appl. Comput.*, Nicosia, Cyprus, 2004, pp. 420–424.
[23] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Exp. Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, 2009.
[24] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang, "A modular machine learning system for flow-level traffic classification in large networks," *ACM Trans. Knowl. Discovery Data*, vol. 6, no. 1, pp.1–34, 2012.
[25] M. Q. Ali, E. Al-Shaer, H. Khan, and S. A. Khayam, "Automated anomaly detector adaptation using adaptive threshold tuning," *ACM Trans. Inf. Syst. Secur.*, vol. 15, no. 4, pp. 1–30, Apr. 2013.
[26] G. P. Zhang, "Neural networks for classification: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 30, no. 4, pp. 451–462, Nov. 2000.
[27] T. C. Silva and L. Zhao, "Network-based high level data classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 6, pp. 954–970, Jun. 2012.
[28] J. Zhao, M. Chen, and Q. Luo, "Research of intrusion detection system based on neural networks," in *Proc. IEEE 3rd Int. Conf. Commun. Softw. Netw.*, May 2011, pp. 174–178.
[29] M. Al-Fawa'reh, M. Al-Fayoumi, S. Nashwan, and S. Fraihat, "Cyber threat intelligence using PCA-DNN model to detect abnormal network behavior," *Egyptian Informat. J.*, vol. 23, no. 2, pp. 173–185, Jul. 2022.
[30] R. Xu and D. C. Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, Nov. 2005.
[31] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Nov. 1999.
[32] M. Norouzian and S. Merati, "Classifying attacks in a network intrusion detection system based on artificial neural networks," in *Proc. 13th Int. Conf. Adv. Commun. Technol. (ICACT)*, Seoul, South Korea, 2011, pp. 868–873.
[33] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, no. 1, pp. 59–69, Jan. 1982.
[34] H. Ritter and T. Kohonen, "Self-organizing semantic maps," *Biol. Cybern.*, vol. 61, no. 4, pp. 241–254, Aug. 1989.
[35] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.
[36] T. Kohonen, *Self-Organization and Associative Memory*, Berlin, Germany: Springer-Verlag, 1989.
[37] T. Kohonen, *Self-Organizing Maps*. Berlin, Germany: Springer, 1995.
[38] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data," *IEEE Trans. Neural Netw.*, vol. 13, no. 6, pp. 1331–1341, Nov. 2002.
[39] E. J. Palomo, J. M. Ortiz-de-Lazcano-Lobato, E. Dominguez, and R. M. Luque, "An anomaly detection system using a GHSOM-1," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Barcelona, Spain, Jul. 2010.
[40] S. Babu and J. Widom, "Continuous queries over data streams," *ACM Sigmod Rec.*, vol. 30, no. 3, pp. 109–120, Sep. 2001.
[41] M. Salem, S. Reissmann, and U. Buehler, "Persistent dataset generation using real-time operative framework," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Honolulu, HI, USA, Feb. 2014.
[42] F. Hashim, K. S. Munasinghe, and A. Jamalipour, "Biologically inspired anomaly detection and security control frameworks for complex heterogeneous networks," *IEEE Trans. Netw. Service Manage.*, vol. 7, no. 4, pp. 268–281, Dec. 2010.
[43] C. J. Fung, J. Zhang, and R. Boutaba, "Effective acquaintance management based on Bayesian learning for distributed intrusion detection networks," *IEEE Trans. Netw. Service Manage.*, vol. 9, no. 3, pp. 320–332, Sep. 2012.
[44] D. Fisch, A. Hofmann, V. Hornik, I. Dedinski, and B. Sick, "A framework for large-scale simulation of collaborative intrusion detection systems," in *Proc. IEEE Conf. Soft Comput. Ind. Appl.*, Muroran, Japan, Jun. 2008, pp. 125–130.
[45] D. Fisch, F. Kastl, and B. Sick, "Novelty-aware attack recognition—Intrusion detection with organic computing techniques," in *Proc. IFIP Conf. Biologically-Inspired Collaborative Comput.*, Brisbane, QLD, Australia, 2010, pp. 242–253.
[46] A. Sperotto, M. Mandjes, R. Sadre, P.-T. de Boer, and A. Pras, "Autonomic parameter tuning of anomaly-based IDSs: An SSH case study," *IEEE Trans. Netw. Service Manage.*, vol. 9, no. 2, pp. 128–141, Jun. 2012.
[47] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 984–996, Apr. 2014.

[48] P. Zhang, C. Zhou, P. Wang, B. J. Gao, X. Zhu, and L. Guo, "E-tree: An efficient indexing structure for ensemble models on data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 2, pp. 461–474, Feb. 2015.
[49] F. Jiang, Y. Fu, B. B. Gupta, Y. Liang, S. Rho, F. Lou, F. Meng, and Z. Tian, "Deep learning based multi-channel intelligent attack detection for data security," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 2, pp. 204–212, Apr. 2018.
[50] W. Liang, K.-C. Li, J. Long, X. Kui, and A. Y. Zomaya, "An industrial network intrusion detection algorithm based on multifeature data clustering optimization model," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2063–2071, Mar. 2020.
[51] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[52] S. Purohit, R. Neupane, N. R. Bhamidipati, V. Vakkavanthula, S. Wang, M. Rockey, and P. Calyam, "Cyber threat intelligence sharing for cooperative defense in multi-domain entities," *IEEE Trans. Dependable Secure Comput. Comput.*, vol. 135, no. 1, pp. 30–43, Oct. 2022.
[53] F. Bao, I.-R. Chen, M. Chang, and J.-H. Cho, "Hierarchical trust management for wireless sensor networks and its applications to trust-based routing and intrusion detection," *IEEE Trans. Netw. Service Manage.*, vol. 9, no. 2, pp. 69–183, Jun. 2012.
[54] Z. Yu, J. J. P. Tsai, and T. Weigert, "An adaptive automatically tuning intrusion detection system," *ACM Trans. Auto. Adapt. Syst.*, vol. 3, no. 3, pp. 1–25, Aug. 2008.
[55] M. Zolotukhin, T. Hamalainen, and A. Juvonen, "Online anomaly detection by using N-gram model and growing hierarchical self-organizing maps," in *Proc. 8th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Aug. 2012, pp. 47–52.
[56] Y. Yang, D. Jiang, and X. Min, "Using improved GHSOM for intrusion detection," *J. Inf. Assurance Secur.*, vol. 5, pp. 232–239, May 2010.
[57] M. Salem and U. Buehler, "An enhanced GHSOM for IDS," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Manchester, U.K., Oct. 2013, pp. 1138–1143.
[58] S. Kakuru, "Behavior based network traffic analysis tool," in *Proc. IEEE 3rd Int. Conf. Commun. Softw. Netw.*, Xi'an, China, May 2011, pp. 649–652.
[59] M. Burgess, H. Haugerud, S. Straumsnes, and T. Reitan, "Measuring system normality," *ACM Trans. Comput. Syst.*, vol. 20, no. 2, pp. 125–160, May 2002.
[60] R. Elwell and R. Polikar, "Incremental learning of concept drift in non-stationary environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.
[61] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Boston, NY, USA: Addison-Wesley, 2005.
[62] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, Berkeley, CA, USA, 1967, pp. 281–297.
[63] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient K-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.
[64] MIT. (Jan. 9, 1999). *1999 Darpa Intrusion Detection Evaluation Dataset*. Accessed: May 30, 2022. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset
[65] *JUMO*. Accessed: Jun. 5, 2022. [Online]. Available: https://www.jumo.de/
[66] I. Witten, E. Frank, and M. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*. Amsterdam, The Netherlands: Elsevier, 2011.
[67] University of New Brunswick. *NSL-KDD Dataset*. Accessed: Jun. 6, 2022. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html
[68] M. Watson. Mar. 7, 2017. *8 Key Application Performance Metrics & How to Measure Them*. Stackify. Accessed: Jun. 6, 2022. [Online]. Available: https://stackify.com/application-performance-metrics/
[69] M. Ozkan-Okay, R. Samet, O. Aslan, and D. Gupta, "A comprehensive systematic literature review on intrusion detection systems," *IEEE Access*, vol. 9, pp. 157727–157760, 2021, doi: 10.1109/ACCESS.2021.3129336.
[70] N. Wattanapongsakorn, S. Srakaew, E. Wonghirunsombat, C. Sribavonmongkol, and T. Junhom, "A practical network-based intrusion detection and prevention system," in *Proc. Trust, Security Privacy Comput. Commun. (TrustCom), IEEE 11th Int. Conf.*, Jun. 2012, pp. 209–214.
[71] S. Kumar, A. Viinikainen, and T. Hamalainen, "Machine learning classification model for network based intrusion detection system," in *Proc. 11th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2016, pp. 242–249.
[72] Q. Qassim, A. M. Zin, and M. J. A. Aziz, "Anomalies classification approach for network-based intrusion detection system," *Int. J. Netw. Secur.*, vol. 18, no. 6, pp. 1159–1172, 2016.
[73] A.-U.-H. Qureshi, H. Larijani, J. Ahmad, and N. Mtetwa, "A novel random neural network based approach for intrusion detection systems," *Proc. 10th Comput. Sci. Electron. Eng. (CEEC)*, Sep. 2018, pp. 50–55.
[74] M. Mazini, B. Shirazi, and I. Mahdavi, "Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 31, no. 4, pp. 541–553, Oct. 2019.
[75] S. Meftah, T. Rachidi, and N. Assem, "Network based intrusion detection using the UNSW-NB15 dataset," *Int. J. Comput. Digit. Syst.*, vol. 8, no. 5, pp. 478–487, 2019.
[76] P. Devan and N. Khare, "An efficient XGBoost–DNN-based classification model for network intrusion detection system," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12499–12514, Jan. 2020.
[77] P. Bedi, N. Gupta, and V. Jindal, "I-SiamIDS: An improved siam-IDS for handling class imbalance in network-based intrusion detection systems," *Int. J. Speech Technol.*, vol. 51, no. 2, pp. 1133–1151, Feb. 2021.

MAHER SALEM received the M.Sc. degree from Duisburg-Essen University, and the Ph.D. degree from Kassel University, Germany. He is currently a Lecturer with the Department of Informatics, King's College London, U.K. He is also the Deputy Chair of the Assessment Sub-Board of Online Cybersecurity. He has more than 20 years of industrial and academic experience in cybersecurity. His research interests include intrusion detection systems, threat intelligence, machine learning, and cybersecurity in education.

ABDEL-KARIM AL-TAMIMI received the M.Sc. and Ph.D. degrees in computer engineering from Washington University in St. Louis. He is currently a Senior Lecturer with the Department of Computing, Sheffield Hallam University, U.K. His research interests include machine learning/AI, multimedia networks and applications, and computer security.