Received November 20, 2020, accepted December 21, 2020, date of publication January 5, 2021, date of current version January 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3049249

Intrusion Detection System Based on Integrated System Calls Graph and Neural Networks

F. J. MORA-GIMENO[1], H. MORA-MORA[1], B. VOLCKAERT[2], AND A. ATREY[2]
[1]Department of Computer Technology, University of Alicante, 03690 San Vicente del Raspeig, Spain
[2]Department of Information Technology, Ghent University, B-9000 Ghent, Belgium

Corresponding author: F. J. Mora-Gimeno (fjmora@dtic.ua.es)

This work was supported by the Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital of the Community of Valencia, Spain, within the Program of Support for Research under Project AICO/2020/206.

ABSTRACT Computer security is one of the main challenges of today's technological infrastructures, whereas intrusion detection systems are one of the most widely used technologies to secure computer systems. The intrusion detection systems use a variety of information sources, one of the most important sources are the applications' system calls. The intrusion detection systems use many different detection techniques, e.g. system calls sequences, text classification techniques and system calls graphs. However, existing techniques obtain poor results in the detection of complex attack patterns, so it is necessary to improve the detection results. This paper presents an intrusion detection system model that integrates multiple detection techniques into a single system with the goal of modeling the global behavior of the applications. In addition, the paper proposes a new modified system calls graph to integrate and represent the information of the different techniques in a single data structure. The system uses a deep neural network to combine the results of the different detection techniques used in the global model. The result of the study shows the improvement obtained in the detection results with respect to the use of individual techniques, the proposed model achieves higher detection rates and lower false positives. The proposal has been validated onto three datasets with different levels of complexity.

INDEX TERMS Anomaly detection, intrusion detection system, neural networks, system calls graph.

I. INTRODUCTION

Although application-level intrusion detection systems (IDS) based on the use of system calls are relatively old [1], there has been a significant increase in the number of research papers using system calls to detect intrusions in recent times [2]–[4].

There are two main IDS approaches: anomaly detection and signature detection. The anomaly-based IDS defines the normal behavior of the application monitoring its system calls and different behaviors that exceed a certain threshold are classified as intrusion. The signature-based IDS use a database with the behavior of all known malware (malware signatures) and perform the detection monitoring the running applications and looking for these behaviors or signatures.

Detection systems have used a variety of techniques to represent application behavior. One of the most widely used techniques models the application behavior as system calls sequences. Abnormal sequences of system calls are considered an attack to the application [5]. This type of systems usually uses a set of short sequences of system calls generated by the program.

A different approach uses frequencies of system calls to characterize the normal behavior of an application. Different frequencies of system calls are considered an attack to the application [6].

Another widely used technique represent the application behavior as a directed acyclic graph, where the nodes are the system calls generated by the application [3]. The basic idea is that an attack will generate a different system calls graph and, therefore, the attack can be detected.

Based on the system calls graph technique, there are studies that take into account the importance of the edges of the graph, which represent transitions between system calls of the application [2].

Existing techniques obtain poor results in the detection of complex attack patterns, so it is necessary to improve the detection results. Existing techniques have a common weakness, these techniques use a partial view of the applications behavior: sequences of system calls, frequencies of system

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

calls and importance of the transitions between different calls. Therefore, since they model the application behavior only with partial information, their results can be improved using global information.

Our proposal integrates different techniques or partial views in a single model, the integration allows to represent the global behavior of the application with the aim of improving the results in the detection of attacks. In addition, the model uses deep neural networks to combine the different partial views in a single global view of the application behavior.

The key contributions of our work can be summarized as follows:

- The design of an IDS model that integrates different detection techniques to model the global behavior of the applications.
- The proposal of a new modified system calls graph, which allows to represent the information of the different techniques or views in a single data structure.
- An improvement in the detection accuracy compared to the use of single techniques.

The most important novelty of this study is the integration of multiple intrusion detection techniques into a single system, which improves the detection accuracy. In addition, the work proposes an integrated system calls graph, which allows to represent all the information from the different techniques in a single data structure.

The rest of the paper is organized as follows: Section II presents a review of related studies. Section III analyzes the proposed IDS model. Section IV shows the results of the evaluation with different datasets. Finally, Section V provides some conclusions and future directions.

II. RELATED WORK

IDS are a very active research field, both in network and host IDS. Network IDS detect attacks based on network traffic [7], and one of the main issues in this type of IDS is feature selection [8], [9]. Wireless sensor networks are one of the most important areas of interest in current network IDS [10], where techniques such as restricted Boltzmann machine [11], reinforced learning [12] and a combination of several classifiers [13] have been applied. In recent years, ad hoc vehicular networks have become another area of growing interest in network IDS [14], cloud service frameworks have been proposed to enable intrusion detection mechanisms [15], as well as attack detection fremeworks for the vehicle communication bus [16]. Host IDS detect attacks based on system logs, audit data, Windows Registry, file systems, system calls, and program analysis [17]. Two of the most important areas of interest in host IDS are embedded systems [18] and the Android operating system [19], [20].

Regarding the detection techniques used in IDS, three of the most used techniques are Hidden Markov Models (HMM), data mining and Bayesian networks. HMM have been used to predict multistep attacks in real time [21], as well as to detect anomalies in smart home security [22]. Data mining algorithms have been used to detect intrusions for Smart grids [23], and the use of Ramdom Forest for intrusion classification is also extensive [24]. Finally, Bayesian networks have been used to predict the risk of internal attacks [25], as well as to detect anomalies for IoT technology [26].

Since our proposal consists of an IDS model that uses the application's system calls to carry out the detection process, in the following paragraphs we have analyzed other directly related proposals, showing the most used techniques in this type of systems.

The literature contains a large number of scientific papers that use system calls of the applications to develop IDS [27], [28]. However, most papers have in common that they use a single technique to model the applications behavior. In addition, they often use simple detection methods, based on distance measurements. Many of these works are presented in this section because they are directly related to the proposal of this paper.

The first paper that used system calls of the applications to detect intrusions, used system calls sequences of length 'k' to model the applications behavior [1]. This paper used Hamming's distance to measure the distance between two sequences. In addition, if the distance exceeded a predefined threshold, it was considered an anomaly. Other works using system calls sequences are [29], [30].

Another technique, based on text classification methods, models the applications behavior through a vector, where each component represents a system call and its value is the normalized frequency of the system call in the execution of the program [6]. This work uses the cosine similarity to measure the similarity between two executions of the program. If the similarity measure is below a predefined threshold, it is considered an anomaly. Similar works can be seen in [31]–[33].

A different approach to the previous ones models the applications behavior using system calls graphs [2]. This work uses the PageRank algorithm to calculate the weights of the graph edges and the Hamming distance weighted by the weights. Other works that use system calls graphs can be seen in [3], [4].

There are IDS that have used machine learning techniques to detect system attacks. In [34], the authors show a review of several machine learning algorithms that have been used to develop IDS. Other works have evaluated the performance of different machine learning algorithms used in intrusion detection [35], [36].

The multi-layer perceptron (MLP) algorithm with a single hidden layer has been used as a detection method in IDS [37]. This work has used binary weights and activation functions for intrusion detection. The same MLP neural network with a single layer has also been used in an IDS applied to the field of fog computing [38]. Unsupervised learning algorithms such as self-organizing map (SOM) or growing neural gas (GNG) have also offered promising results in the field of IDS [39].

In addition to machine learning, there are recent studies that employ deep learning in the field of IDS [40]. In this paper, the authors show a flexible and effective IDS for detecting and classifying cyberattacks, that uses a deep MLP neural network. In addition, the performance of deep neural network has been compared to classical neural classifiers. Other deep neural networks used in IDS are convolutional neural networks and recurrent neural networks [41]–[47].

There are works that introduce the idea of combining two techniques to achieve better performance [48]. The work discusses the combination of the probabilistic models Markov and Bayes for host-based intrusion detection. The results improve the use of the two models separately, especially, the results reduce the false positive rate.

Following the same idea, another paper combines support vector machine and another technique in a network intrusion detection system [49], [50]. In [51], a host detection system based on system calls combines clustering and Bayes techniques. Finally, in [52], a network IDS uses two parallel classifiers to detect and track ransomware. The paper shows an exhaustive analysis of the behavior of Locky ramsomware.

From the analysis carried out in the previous paragraphs, we observed that the IDS have used a multitude of different techniques in an isolated manner. In addition, in order to improve performance, the idea of using two techniques is explored. However, although the use of two techniques improves performance, it is still far from optimal for real production environments. Therefore, existing techniques obtain poor results in the detection of complex attack patterns, so it is necessary to improve the detection results.

On the one hand, the main difference between our proposal and those presented in this section is that our proposal employs multiple techniques and these works use only one technique. On the other hand, several papers show systems that use two methods [48]–[52], but our proposal combines more methods and represents all the information in a single data structure.

Our proposal combines, in a single system, four of the techniques analyzed in this section with the goal of modeling the global behavior of the applications. The system uses a new modified system calls graph and improves the detection accuracy. The proposed system has a global vision of the applications behavior due to the different techniques and the system is able to improve the ability to detect attacks. The main difference between our proposal and those analyzed in this section is that our proposal combines multiple detection techniques in a single system and improves the performance obtained.

III. PROPOSED IDS MODEL

Several techniques used in IDS based on system calls have been described in the previous section. Most of the analyzed papers use a single technique, but some papers have explored the idea of using two techniques simultaneously, improving the results obtained. The purpose of this section is to describe the IDS model proposed in this paper, which delves into the idea of integrating multiple detection techniques into a single system, which uses a new modified system calls graph and improves the detection accuracy.

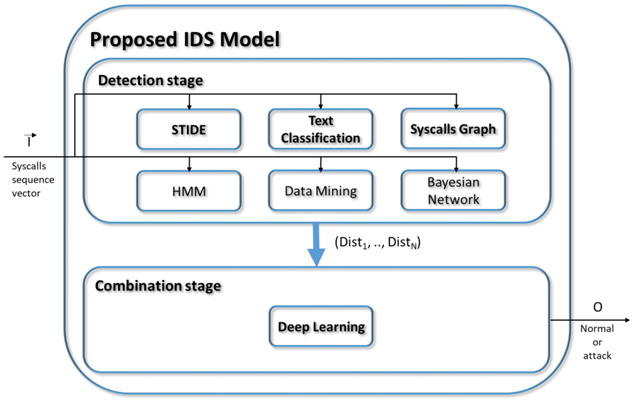

FIGURE 1. **Proposed IDS model.**

First, we analyze the architecture of the system, and then we describe the operation of the different elements.

A. SYSTEM ARCHITECTURE

The architecture of the proposed system has two main characteristics: two stages and modularity. The system has been divided into two clearly differentiated stages, the first stage will carry out the detection process and the second will carry out a process of combination of the results obtained in the first stage. The main novelty of the detection stage is that it is able to use multiple intrusion detection techniques simultaneously. This feature allows the system to perform the detection from multiple points of view, getting a global vision of the applications behavior and increasing the probabilities of detection of attacks.

The second stage combines the results obtained by the different techniques used in the detection phase. The integration stage is able to improve the detection performance, because it is based on information provided by multiple detection engines, which provide a more complete view of the execution of applications.

The second characteristic of the proposed system architecture is that its design is modular. The main advantage of modularity is that the detection stage can use as many modules as desired, where each module can implement a detection technique. In addition, new modules can be added, which implement new detection techniques without affecting the existing modules. It is even possible to change one module for another without affecting the overall operation of the system.

Fig. 1 shows a high-level abstraction view of the proposed IDS model architecture. Although the model allows many methods, the prototype developed uses three different techniques (bold text) in the detection stage: sequence time-delay embedding (STIDE), text classification and system calls graph. These methods use very different analysis techniques, providing different views of the applications behavior and allowing to take advantage of each of them. In addition, the system uses deep learning in the integration stage to combine the results obtained by the three techniques used in the detection stage.

The system input I is a vector that contains the sequence of all system calls made by the application. All detection modules use the same input vector I. Using the input vector I, STIDE generates sequences of system calls of size 'k', the text classification technique calculates the frequency of the system calls and the system call graph method creates the graph with weights on the edges.

The output of each detection module is a measure of distance between 0 and 1. A value close to 0 is interpreted as the normal behavior of the application and a value close to 1 is considered an attack.

The input to the combination module is a vector that contains all the distances of the different detection modules of the detection stage. Finally, the system output O is a Boolean value indicating normal behavior or attack.

B. DETECTION MODULES

The previous architecture shows that the first stage of the proposed model consists of the following three modules: STIDE [1], text classification [32] and system calls graphs with weights on the edges [2]. We chose STIDE because it is the first technique that used application system calls as input to the system and because it is one of the most widely used techniques in many research papers. We selected text classification and system calls graph because they are two of the most used techniques in application-based IDS.

The technique called STIDE or system calls sequences of length 'k' provides a view of the sequential execution of the applications, an application is a sequence of instructions that invoke system calls. It has the advantage of adjusting very well to the normal execution of the applications and it generates compact representations of the applications profiles.

The second method, based on text classification techniques, provides a higher level view, using the frequencies of system calls made by the application. It would be similar to representing the application behavior using a histogram, where the anomalies would be different histograms. It has the advantage of requiring little calculation for its execution and the representation of the application behavior is very compact, basically a vector.

Finally, the system calls graph with weights on the edges provides a perspective from the point of view of the importance of transitions between system calls, not the calls themselves. It has the advantage of offering good results in isolation, but it is more complex than the previous ones.

The three methods used have their own advantages and have achieved good results in isolation. However, all three methods use different and complementary approaches, so when used together in a single system, they achieve a more complete model of application behavior and better results in attack detection.

C. INTEGRATED SYSTEM CALLS GRAPH

One of the novelties of the paper is the proposal of an integrated system calls graph (ISCG), which allows to represent the information of the different techniques or views, in a single data structure.

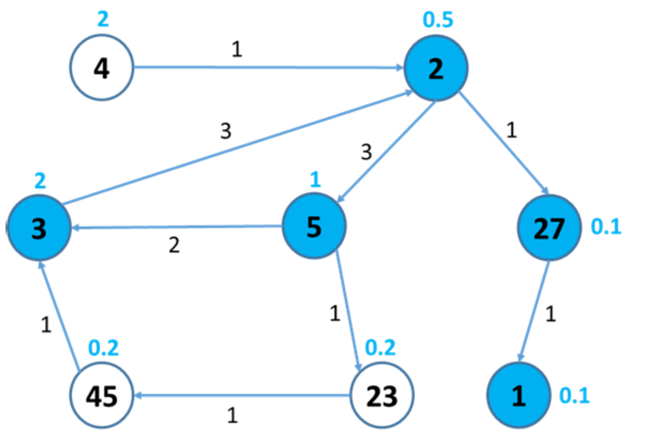

FIGURE 2. Integrated system call graph with system call numbers, system call frequencies and edge weights.

The proposed IDS model represents the application behavior via an integrated system calls graph. The normal system calls graph has been modified in order to include the information of the three methods.

The graph will have weights on the edges, in order to represent the importance of transitions between system calls. In this sense, the graph will be similar to the graph used in other previous works that use the system calls graph technique [2].

From the previous graph with weights on the edges, we add some additional feature that allows us to represent more information. The additional feature is that the nodes of the graph will have values, which will represent the frequency of system calls made by the application. Each node is associated to a system call and its value will be the frequency of that system call.

Finally, in order to be able to represent the system calls sequences of length 'k' from a given system call, the IDS searches in the graph for paths of length 'k', from the node that represents the given system call.

Fig. 2 shows a simple example of an integrated system calls graph. We can see the weights of the edges that will be the information source of the technique that evaluates the transition between system calls. Fig. 2 shows the values of the graph nodes, which represent the frequencies or percentages of the system calls associated to each node (blue values), these values will be used by the detection method based on text classification. Finally, as an example, two paths of length 3 are shown from the system call 2. These paths generate the sequences of system calls 2-5-3 (fork-open-read) and 2-27-1 (fork-alarm-exit), which can be used by STIDE method.

The STIDE method saves in a database all the combinations of system calls sequences of length 'k' that an application generates when running normally, that is, with the security that the application is not compromised. This database represents the normal (safe) behavior of the application. Then, when the application is running, STIDE compares the generated system calls sequences of length 'k' with those stored in the database (normal behavior) and classifies as anomalies the sequences that do not exist in the database. If the anomalies exceed a predefined threshold, it is considered an attack to the application and the alert is activated.

In this case, the global normal behavior of the application is represented in the integrated system calls graph. But, for the STIDE algorithm, the normal behavior is formed by all the k-length paths that exist in the graph. From the graph, STIDE compares the system calls sequences of length 'k' generated by the application with the k-length paths of the graph. If a program sequence is not in the graph, it is considered an anomaly. Observing the previous figure and being k=3, it can be verified that the sequence 2-5-3 is in the graph and is correct, while the sequence 2-5-4 is not in the graph and it is considered an anomaly.

For the detection method based on text classification, the values of the graph nodes represent the normal behavior of the application. Each node in the graph is associated with a system call, and the value of the node is usually the frequency or percentage of invocation of that system call in the full execution of the application. In the previous figure, the system call 3 (read) has been executed 2% of the time and the system call 5 (open) has been executed 1% of the time.

Each time the application is run, the values of all system calls are calculated and compared with the values of the nodes of the normal behavior graph. If the distance exceeds a previously defined threshold, the application is considered compromised, that is, it is considered an attack. As mentioned above, the most commonly used distance measurements are the euclidean distance and the cosine distance.

Finally, for the technique that evaluates the importance of transitions between system calls, the weights of the graph edges represent the normal behavior of the application. Each directed edge represents a transition between the two system calls and the weight represents the importance of the transition. For example, in Fig. 2 a transition can be observed from the system call 5 (open) to the system call 3 (read) with weight 2.

Each time the application is run, the weights of all edges are calculated and compared with the weights of the edges of the normal behavior graph. If the distance exceeds a previously defined threshold, the application is considered compromised or attacked. As in the previous case, the most commonly used distance measurements are the euclidean distance and the cosine distance.

D. COMBINATION MODULE

The second stage of the proposed model consists of a single component, the combination module. The function of the combination module is to integrate the results of the different detection modules used in the first stage. The combination module determines whether an attack is occurring on the application.

The input to the combination module consists of the three distances obtained by the three different methods used in the detection stage. The combination module determines whether the application is victim of an attack based on the three input distances. Of course, the proposed model is general and it could employ any number of different methods in the first stage.

Each method of the detection stage returns a distance measure between 0 and 1, so each method operates in a one-dimensional space. The combination of three methods operates in a three-dimensional space, where normal behavior is close to the point (0, 0, 0) and attacks are close to the point (1, 1, 1). The combination of three methods works better due to the supervised learning process of the neural network, using the distances of the detection modules and tagged datasets. Because datasets are tagged, the supervised deep neural network learns from the examples and improves the results obtained.

The proposed system employs machine learning in the combination module, specifically deep neural networks due to its generalization capability and promising results. In addition, neural networks have been used as a detection method in IDS many times.

The combination module implements a deep MLP neural network with three main hidden layers. We have used a deep MLP neural network for its simplicity, but we will test recurrent neural networks in future work to exploit the sequential nature of system calls.

The aim of the paper is to show that the combination of multiple detection techniques improves the results compared to the use of individual techniques. In this sense, the goal of the combination module is to integrate the results and we have chosen Deep MLP and SGD because it is the simplest deep neural network that allows combining the results of the different techniques. However, a recurrent neural network such as Long Sort Term Memory (LSTM) or Gated Recurrent Unit (GRU) with "Adam" optimizer would probably get better results.

Tests have been conducted to select the architecture that offered the best performance. From an MLP network with a single hidden layer, hidden layers have been added to test the performance improvement. With three hidden layers, the performance improvement has been significant. However, from four hidden layers, there is little additional gain from the new layers. Therefore, the three-layer hidden architecture has been selected and performance has been improved by incorporating regularization functions and advanced optimization techniques.

The three hidden layers are fully connected and dropout functions are used to provide regularization and prevent overfitting. Each layer of the deep MLP is composed of the following functions: dense (64), activation ('relu'), dropout (0.3). The dense function connects all neurons in one layer with all neurons in the next layer, in this case 64 neurons. The activation function is Rectified Linear Unit (relu), which has better results than the sigmoid function. The dropout function

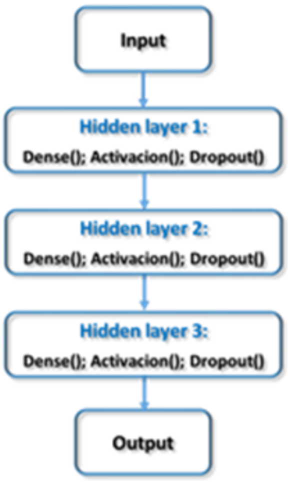
Input
Hidden layer 1:
Dense(); Activacion(); Dropout()
Hidden layer 2:
Dense(); Activacion(); Dropout()
Hidden layer 3:
Dense(); Activacion(); Dropout()
Output

FIGURE 3. Deep MLP neural network architecture.

is a well-known form of regularization in machine learning. Fig. 3 shows the deep MLP neural network architecture and the functions used in the hidden layers.

In the proposed model, the deep MLP neural network input will be formed by a three-component feature vector, which are the three distances obtained by the three detection methods used in the first stage. In the case of using more detection methods in the first stage, the features vector would be formed by more components.

The complexity analysis of the proposed model depends on the complexity of the two stages of the system [53]. The complexity of the first stage is a function of the complexity of the three detection modules. The time complexity of STIDE is related to the number of sequences of size 'k' (n), the complexity of text classification is based on the number of system calls (m) and the complexity of system calls graph depends on the number of graph edges (p).

Based on the above, C1 is the complexity of STIDE as defined in (1), C2 is the complexity of text classification as shown in (2), and C3 is the complexity of system calls graph as specified in (3).

$$\mathrm{C1} = O(\mathrm{n}) \tag{1}$$
$$\mathrm{C2} = O(\mathrm{m}) \tag{2}$$
$$\mathrm{C3} = O(\mathrm{p}) \tag{3}$$

Since the modules are executed in parallel, the time complexity of the detection stage will be directly related to the module with the highest time cost. In our analysis and simulation of the different techniques, we have observed that the number of STIDE sequences is greater than the number of system calls and graph edges, being STIDE the technique with the greatest time complexity. Therefore, the complexity of the detection stage (CD) will be similar to C1 as defined in (4).

$$\mathrm{CD} = O(\mathrm{n}) \tag{4}$$

The complexity of the second stage is a function of the combination module, which is related to the number of components of the input vector. In this case, the input vector has 3 components. Therefore, the general time complexity of the model (CM) can be formulated as in (5).

$$\mathrm{CM} = O(\mathrm{n}+3) = O(\mathrm{n}) \tag{5}$$

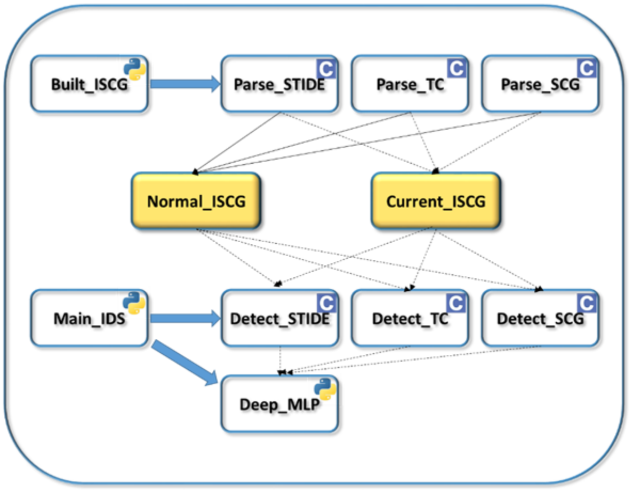

FIGURE 4. Prototype structure.

IV. RESULTS

In this section, we describe the prototype that has been created with all the components of the model and the tests that have been carried out to validate the proposed system. In addition, the tests have been carried out on several data sets, in order to offer a wider vision of the performance of the proposed system in different scenarios. The tests compare the proposed model with three of the existing techniques, specifically with the three techniques used individually in the detection phase: STIDE, text classification and system calls graph.

A. PROTOTYPE

To perform the necessary tests to validate the proposed model, a prototype has been created with all the components of the model. The prototype consists of four main modules, which implement the four components of the model.

The detection components have been developed using python scripts and C language modules. The C-language modules implement the main detection algorithms and other support functions, e.g. parse input data. The proposal can be applied to any dataset by adapting the modules that parse the input, but it is better if each dataset file only contains traces of a single process. Python scripts have been used to orchestrate the operation of the detection stage, obtaining the system inputs and executing the corresponding C modules. Fig. 4 shows the different components of the implemented prototype.

Fig. 4 shows that the prototype has two clearly differentiated parts. The objective of the first part is the creation of integrated system calls graphs, which will be used by the detection algorithms. The objective of the second part is the detection of attacks, based on the information stored in the previous integrated system calls graphs.

The purpose of the Built_ISCG component is to orchestrate the execution of the three modules that parse the input and generate the graphs. Parse_STIDE creates the graph, in which system calls sequences of any length can be found, by means of paths of the same length. Parse_TC will add values to the nodes, which represent the system calls, the values are the frequency of use of each system call. Finally, Parse_SCG will add weights to the graph edges, the weights are the relative importance of the transitions between the different system calls.

The Normal_ISCG is the integrated system calls graph that stores the information that represents the normal behavior of application execution. It is generated by running secure or attack-free instances of the application. On the other hand, the Current_ISCG represents the current application execution. The different detection algorithms compare the two graphs to look for important deviations that indicate an anomalous behavior or attack.

The objective of the Main_IDS component is to orchestrate the execution of the modules that implement the two main phases of the proposal. First, Main_IDS invokes the modules that execute the different detection algorithms: Detect_STIDE, Detect_TC and Detect_SCG. Then, Main_IDS executes the module that implements the integration of the previous results: Deep_MLP.

Detect_STIDE implements the STIDE algorithm and executes it over the two input graphs, considering the system calls sequences of length 'k' as k-length paths of the graph. Detect_TC implements the text classification algorithm and uses the values of the graphs nodes to calculate the distance measurement between the two executions. Detect_SCG implements the edge weighting algorithm and uses the weights of the graphs edges to calculate the distance.

Finally, Deep_MLP implements the deep MLP neural network architecture discussed in the previous section. The input to the neural network is a vector of three components with the three distances obtained by the detection modules and the output will be the classification of the current application execution as NORMAL or ATTACK. The neural network has been implemented in Keras and using TensorFlow as backend.

The three hidden layers of the deep MLP have been implemented using three Dense layers with 64 neurons. After each hidden layer, regularization has been added through a Dropout layer, using a value of 0.3. The activation function used was rectified linear unit (ReLU), because it gave better results than the sigmoid function. The output layer was Dense with 2 neurons and softmax activation function. The optimization technique used was Adam, because it includes the concept of momentum and achieves better results than stochastic gradient descent (SGD). The number of epochs used in the network training has been 20.

In addition, other configurations have been tested such as increasing the number of hidden neurons or increasing the number of epochs. But, they generated a much more complex network that needed much more training time and did not achieve a significant improvement in performance.

B. RESULTS USING THE DARPA DATASET

The following sections analyze the results obtained in the evaluation tests of the proposed model. The tests performed compare the proposed system, which combines multiple detection techniques using neural networks with each of the techniques separately. It should be noted that these techniques have been used independently in detection systems proposed in many research papers [1]–[6].

As mentioned above, the evaluation of the detection model proposed in this paper has been carried out using several datasets. We have chosen the DARPA, University of New Mexio (UNM) and Australian Defense Force Academy Linux Dataset (ADFA_LD) datasets because they have different levels of complexity, the DARPA dataset is the simplest and ADFA_LD is the most complex, it has the most complex attack patterns. The objective is to evaluate the behavior of the proposed system in scenarios with different levels of complexity. Additionally, there are very few datasets that include application system calls, and the selected datasets are the most important.

This section analyses the evaluation results using the DARPA dataset [54], [55]. This dataset is old and has received several criticism related to the synthetic nature of the data and the methodology used [56], [57], but it is still used in modern papers as a starting point [58].

The DARPA dataset consists of seven weeks of training data and two weeks of test data. We have used the basic security module (BSM) audit logs that contain information of the system calls generated by the running programs, the same data that was used in [59]. For the training, we have used four days with about 2000 sessions and, for the test, we have used one day with 412 sessions. The data used contains 55 intrusive sessions with 35 different types of attacks.

We will use the receiver operating characteristic (ROC) curves to show the comparison between the different methods. The ROC curves relate the detection rate and the false positive rate. When analyzing ROC curves to compare the curves from different techniques, it is important to observe the space or area under the curve. The curve with the most space underneath is considered the best. Therefore, the curve closest to the upper left corner will be the technique that achieves the best results. The upper left corner represents the perfect classification, 100% detection with no false positives. We can interpret the ROC curve as a cost-benefit rate, in the upper left corner we get the maximum benefit, without any cost.

Fig. 5 shows the comparison of the results obtained by the different methods using the DARPA dataset.

In Fig. 5, STIDE is the method that obtains the worst results, it generates 6% of false positives to be able to detect all the attacks. The methods of text classification and system calls graph show average and very similar results, generating 5% false positives to detect 100%. Fig. 5 shows that the proposed model (ISCG in the figure), based on a

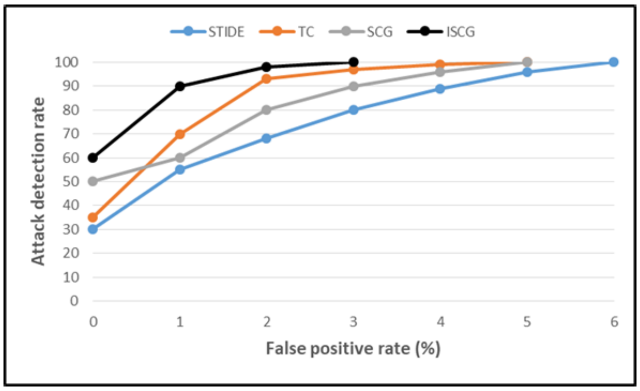

FIGURE 5. ROC curve for DARPA dataset.

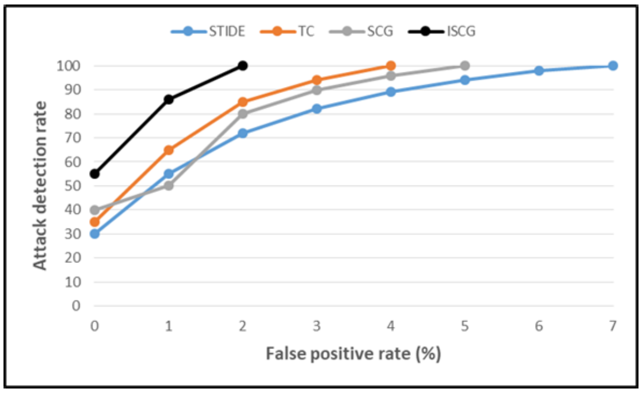

FIGURE 6. ROC curve for lpr application of UNM.

graph that integrates the three previous methods, achieves the best results. The proposed ISCG model shows the combined action of all methods. The proposed model detects 100% of the attacks, generating only 3% of false positives. On the other hand, the model is capable of detecting 60% of the attacks without producing false positives.

C. RESULTS USING THE UNM DATASET

In this section, we have evaluated the proposed model using the dataset provided by the University of New Mexico [60]. This dataset consists mainly of traces of executions of three main applications: lpr, sendmail and ftpd. Each trace contains the system calls sequence that the application has generated in each execution. In addition, the dataset is divided into synthetic data (controlled synthetic environment) and real data (live environment).

The first test compares the performance of the proposed system against the individual techniques for the lpr application executed in real environments. In this case, the dataset contains 2766 different traces. Fig. 6 shows the result obtained by each of the compared methods. The lpr application is very simple, so the results are very similar to the results obtained with the DARPA dataset.

In Fig. 6, STIDE is the method that obtains the worst results, it generates 7% of false positives to be able to detect all the attacks. The methods of text classification and system calls graph show average results, somewhat better in the

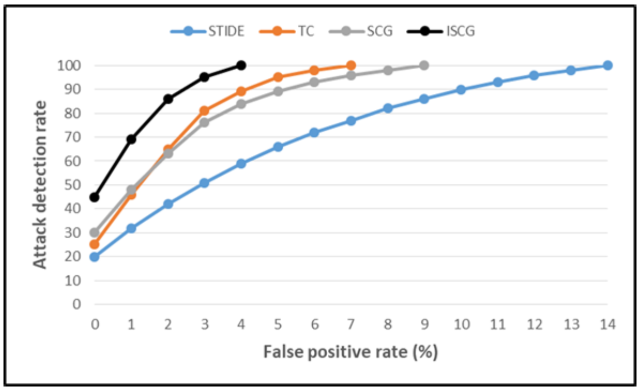

FIGURE 7. ROC curve for sendmail application of UNM.

case of the text classification method, these methods generate 4% and 5% respectively of false positives to detect 100%. Fig. 6 shows that the proposed ISCG model achieves the best results, even increasing the distance to the other methods with respect to the evaluation made with DARPA data. The proposed model detects 100% of the attacks, generating only 2% of false positives. On the other hand, the model is capable of detecting 55% of the attacks without producing false positives.

The following test uses the sendmail application of the UNM dataset to compare the different methods. The sendmail application is more complex than the lpr application, so the results are worse. The dataset contains 147 traces of normal sendmail execution and 16 attacks of 5 different types. Fig. 7 shows the evaluation results of the proposed model using the sendmail dataset.

In Fig. 7, all methods have worse results than with previous datasets, they need higher false positive rates to detect all attacks. This behavior is logical, because the sendmail application and the types of attacks used are more complex.

Fig. 7 shows that STIDE is the method that obtains the worst results with quite a difference over the rest of the methods, it generates 14% of false positives to be able to detect all the attacks. The methods of text classification and system calls graph show average results, somewhat better in the case of the text classification method, these methods generate 7% and 9% respectively of false positives to detect 100%. As in previous tests, the proposed ISCG model achieves the best results. The proposed model detects 100% of the attacks, generating only 4% of false positives. On the other hand, the model is capable of detecting 45% of the attacks without producing false positives.

D. RESULTS USING THE ADFA-LD DATASET

Finally, the last test is performed with the Australian Defense Force Academy Linux Dataset (ADFA-LD) which is a new dataset representative of the methodology, structure and complexity of modern attacks [61]–[63]. The ADFA-LD dataset consists of 833 normal training data traces, 4373 normal validation data traces and 60 attacks of six different complex types. Fig. 8 shows the evaluation results obtained by the different methods, using the ADFA-LD dataset.

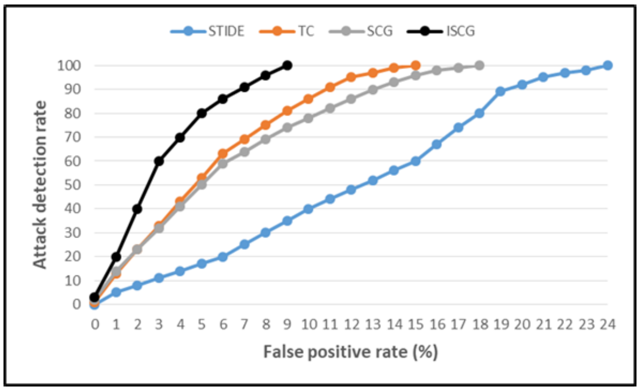

FIGURE 8. ROC curve for ADFA-LD dataset.

TABLE 1. Detection rates without false positives (%).

Dataset	STIDE	TC	SCG	ISCG
DARPA	30	34	50	60
UNM	20	25	30	45
ADFA-LD	0	1	2	3

Fig. 8 shows the evaluation of the different methods with the ADFA-LD dataset obtains the worst results of all evaluated datasets. It is logical because this dataset contains much more complex and current attacks, which mimic the normal execution of applications, making detection more difficult.

Fig. 8 shows that all methods have worse results than previous datasets, due to the complexity of the attacks present in the ADFA-LD dataset. In the worst case, the test generates 24% of false positives to detect all the attacks. In addition, no method achieves a significant detection rate for a false positive rate of 0%.

The distance between the different methods increases, STIDE is the method that obtains the worst results. The methods of text classification and system calls graph show much better results than STIDE, the distance is greater than in the datasets previously evaluated. Finally, the proposed ISCG model achieves the best results and also increases the distance from TC and SCG.

Fig. 8 shows that STIDE is the method that obtains the worst results with much difference over the rest of the methods, STIDE generates 24% of false positives to be able to detect all the attacks. The methods of text classification and system calls graph show average results, somewhat better in the case of the text classification method, these methods generate 15% and 18% respectively of false positives to detect 100%. As in previous tests, the proposed ISCG model achieves the best results. The proposed model detects 100% of the attacks, generating only 9% of false positives.

In order to compare more clearly the performance improvement of the proposed system with respect to existing techniques, Table 1 shows the detection rates obtained by all the systems evaluated, without generating false positives. We can see that the proposed ISCG model obtains the best results in all datasets.

TABLE 2. False positive rates to detect 100% of attacks (%).

Dataset	STIDE	TC	SCG	ISCG
DARPA	6	5	5	3
UNM	14	7	9	4
ADFA-LD	24	15	18	9

TABLE 3. Accuracy rates (%) of STIDE (±1.8), TC (±1.4), SCG (±1.4) and ISCG (±1.2).

Dataset	STIDE	TC	SCG	ISCG
DARPA	95.3	98.4	98.2	100
UNM	92.8	97.3	97.1	99.6
ADFA-LD	87.6	96.1	95.6	98.8

TABLE 4. Processing time performance.

Dataset	STIDE	TC	SCG	ISCG
DARPA	0.64	0.14	0.11	0.67
UNM	0.69	0.15	0.12	0.73
ADFA-LD	1.02	0.21	0.16	1.06

In addition, Table 2 shows the false positive rates obtained by all the systems evaluated, in order to detect 100% of the attacks. We can see that the proposed ISCG model achieves much better false positive rates than the rest of the systems in all datasets.

Finally, Table 3 presents the accuracy of the results for the different techniques and datasets compared. Accuracy shows the ratio of the truly classified instances which are the True Positive (TP) and True Negative (TN) instances [13].

The proposed system has the disadvantage of requiring more processing time. Table 4 shows the processing time (seconds) of the different methods. Tests have been performed on 100 traces of each dataset. ISCG produces a 4% overload in processing time. The tests have been performed on a computer with Intel Core i7-4790 CPUs with four cores and 16 GB RAM.

V. CONCLUSION

One of the challenges and open issues in the field of IDS is that attack patterns are increasingly complex and traditional techniques are becoming obsolete, so it is necessary to introduce new techniques or methods that are better suited to complex attack patterns. In this paper, instead of introducing a new technique, we have used an approach that combines multiple existing techniques into a single system, achieving better performance.

This paper has presented the design of an IDS model that integrates multiple detection techniques or partial views to model the global behavior of the applications. Due to the integration of different detection methods and the global information on the applications behavior, the proposed model is able to take advantage of each method. The integration has been carried out by means of artificial intelligence algorithms, specifically a deep MLP neuronal network, which has allowed combining the results of the different detection techniques. The evaluation of the proposed model has shown the improvement obtained in the detection results with respect to the use of individual techniques, using both simple datasets and complex datasets. The overall accuracy of our model can achieve a success rate of 98.8% for complex datasets and 100% for simple datasets.

In addition, an integrated system calls graph has been developed, which represents all the information from the different techniques in a single data structure. The k-length paths within the graph represent the k-length system call sequences used in the STIDE method. The values of the graph nodes represent the use frequencies of system calls, used by the text classification methods. The edge weights show the importance of the different transitions between system calls, used in the graph methods.

Future work will analyze the idea of integrating more detection methods to improve the results obtained and the ways of representing information in a single data structure. Data mining and hidden Markov models are two detection methods that could be used in phase 1, because they are very powerful techniques that have already been employed in IDS.

REFERENCES

[1] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *J. Comput. Secur.*, vol. 6, no. 3, pp. 151–180, Jul. 1998, doi: 10.3233/JCS-980109.

[2] Q. Qian, J. Li, J. Cai, and R. Zhang, "An anomaly intrusion detection system method based on pagerank algorithm," in *Proc. Int. Conf. Green Comput. Commun.*, Beijing, China, Aug. 2013, pp. 2226–2230.

[3] J.-W. Jang, J. Woo, A. Mohaisen, J. Yun, and H. K. Kim, "Mal-netminer: Malware classification approach based on social network analysis of system call graph," *Math. Problems Eng.*, vol. 2015, pp. 1–20, Aug. 2015, doi: 10.1155/2015/769624.

[4] S. D. Nikolopoulos and I. Polenakis, "A graph-based model for malicious code detection exploiting dependencies of system-call groups," in *Proc. 16th Int. Conf. Comput. Syst. Technol. CompSysTech*, Jun. 2015, pp. 228–235.

[5] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Proc. IEEE Symp. Secur. Privacy*, May 1999, pp. 133–145.

[6] Y. Liao and V. Vemuri, "Use of K-nearest neighbor classifier for intrusion detection," *Comput. Secur.*, vol. 21, no. 5, pp. 439–448, Oct. 2002, doi: 10.1016/S0167-4048(02)00514-X.

[7] Z. Mu, H. Liu, and C. Liu, "Design and implementation of network intrusion detection system," in *Proc. Int. Conf. Intell. Transp., Big Data Smart City (ICITBS)*, Jan. 2020, pp. 494–497.

[8] S. B and M. K, "Firefly algorithm based feature selection for network intrusion detection," *Comput. Secur.*, vol. 81, pp. 148–155, Mar. 2019, doi: 10.1016/j.cose.2018.11.005.

[9] L. Hakim, R. Fatma, and Novriandi, "Influence analysis of feature selection to network intrusion detection system performance using NSL-KDD dataset," in *Proc. Int. Conf. Comput. Sci., Inf. Technol., Electr. Eng. (ICOMITEE)*, Oct. 2019, pp. 217–220.

[10] S. Otoum, B. Kantarci, and H. Mouftah, "Adaptively supervised and intrusion-aware data aggregation for wireless sensor clusters in critical infrastructures," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[11] S. Otoum, B. Kantarci, and H. T. Mouftah, "On the feasibility of deep learning in sensor network intrusion detection," *IEEE Netw. Lett.*, vol. 1, no. 2, pp. 68–71, Jun. 2019, doi: 10.1109/LNET.2019.2901792.

[12] S. Otoum, B. Kantarci, and H. Mouftah, "Empowering reinforcement learning on big sensed data for intrusion detection," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[13] S. Otoum, B. Kantarci, and H. T. Mouftah, "A novel ensemble method for advanced intrusion detection in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[14] W. Wu, R. Li, G. Xie, J. An, Y. Bai, J. Zhou, and K. Li, "A survey of intrusion detection for in-vehicle networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 919–933, Mar. 2020, doi: 10.1109/TITS.2019.2908074.

[15] M. Aloqaily, S. Otoum, I. A. Ridhawi, and Y. Jararweh, "An intrusion detection system for connected vehicles in smart cities," *Ad Hoc Netw.*, vol. 90, Jul. 2019, Art. no. 101842, doi: 10.1016/j.adhoc.2019.02.001.

[16] S. Tariq, S. Lee, H. K. Kim, and S. S. Woo, "CAN-ADF: The controller area network attack detection framework," *Comput. Secur.*, vol. 94, Jul. 2020, Art. no. 101857, doi: 10.1016/j.cose.2020.101857.

[17] R. A. Bridges, T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, and Q. Chen, "A survey of intrusion detection systems leveraging host data," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–35, Jan. 2020, doi: 10.1145/3344382.

[18] M. Kadar, S. Tverdyshev, and G. Fohler, "Towards host intrusion detection for embedded industrial systems," in *Proc. 50th Annu. IEEE-IFIP Int. Conf. Dependable Syst. Netw.-Supplemental Volume (DSN-S)*, Jun. 2020, pp. 5–8.

[19] J. Ribeiro, F. B. Saghezchi, G. Mantas, J. Rodriguez, and R. A. Abd-Alhameed, "HIDROID: Prototyping a Behavioral host-based intrusion detection and prevention system for Android," *IEEE Access*, vol. 8, pp. 23154–23168, 2020, doi: 10.1109/ACCESS.2020.2969626.

[20] M. Jerbi, Z. C. Dagdia, S. Bechikh, and L. B. Said, "On the use of artificial malicious patterns for Android malware detection," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101743, doi: 10.1016/j.cose.2020.101743.

[21] P. Holgado, V. A. Villagra, and L. Vazquez, "Real-time multistep attack prediction based on hidden Markov models," *IEEE Trans. Depend. Sec. Comput.*, vol. 17, no. 1, pp. 134–147, Jan. 2020, doi: 10.1109/TDSC.2017.2751478.

[22] S. Ramapatruni, S. N. Narayanan, S. Mittal, A. Joshi, and K. Joshi, "Anomaly detection models for smart home security," in *Proc. IEEE IEEE 5th Intl Conf. Big Data Secur. Cloud (BigDataSecurity) Intl Conf. High Perform. Smart Comput., (HPSC) IEEE Intl Conf. Intell. Data Secur. (IDS)*, May 2019, pp. 19–24.

[23] Z. E. Mrabet, H. E. Ghazi, and N. Kaabouch, "A performance comparison of data mining algorithms based intrusion detection system for smart grid," in *Proc. IEEE Int. Conf. Electro Inf. Technol. (EIT)*, May 2019, pp. 298–303.

[24] M. O. Miah, S. Shahriar Khan, S. Shatabda, and D. M. Farid, "Improving detection accuracy for imbalanced network intrusion classification using cluster-based under-sampling with random forests," in *Proc. 1st Int. Conf. Adv. Sci., Eng. Robot. Technol. (ICASERT)*, May 2019, pp. 1–5.

[25] N. Elmrabit, S.-H. Yang, L. Yang, and H. Zhou, "Insider threat risk prediction based on Bayesian network," *Comput. Secur.*, vol. 96, Sep. 2020, Art. no. 101908, doi: 10.1016/j.cose.2020.101908.

[26] W. Alhakami, A. ALharbi, S. Bourouis, R. Alroobaea, and N. Bouguila, "Network anomaly intrusion detection using a nonparametric Bayesian approach and feature selection," *IEEE Access*, vol. 7, pp. 52181–52190, 2019, doi: 10.1109/ACCESS.2019.2912115.

[27] S. A. Maske and T. J. Parvat, "Advanced anomaly intrusion detection technique for host based system using system call patterns," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Aug. 2016, pp. 441–444.

[28] M. Anandapriya and B. Lakshmanan, "Anomaly based host intrusion detection system using semantic based system call patterns," in *Proc. IEEE 9th Int. Conf. Intell. Syst. Control (ISCO)*, Jan. 2015, pp. 94–97.

[29] M. A. Sekeh and M. A. B. Maarof, "Fuzzy intrusion detection system via data mining technique with sequences of system calls," in *Proc. 5th Int. Conf. Inf. Assurance Secur.*, 2009, pp. 224–231.

[30] K. Lu, Z. Chen, Z. Jin, and J. Guo, "An adaptive real-time intrusion detection system using sequences of system call," in *Proc. Can. Conf. Electr. Comput. Eng., Toward Caring Humane Technol. CCECE*, May 2003, pp. 789–792.

[31] G. R. Kumar, N. Mangathayaru, and G. Narasimha, "Intrusion detection using text processing techniques: A recent survey," in *Proc. The Int. Conf. Eng. MIS ICEMIS*, Sep. 2015, pp. 1–6.

[32] S. Rawat, V. P. Gulati, A. K. Pujari, and V. R. Vemuri, "Intrusion detection using text processing techniques with a binary-weighted cosine metric," *J. Inf. Assur. Secur.*, vol. 1, no. 1, pp. 43–50, Jan. 2006.

[33] B. Subba, S. Biswas, and S. Karmakar, "Host based intrusion detection system using frequency analysis of n-gram terms," in *Proc. TENCON IEEE Region Conf.*, Nov. 2017, pp. 2006–2011.

[34] L. Haripriya and M. A. Jabbar, "Role of machine learning in intrusion detection system: Review," in *Proc. 2nd Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA)*, Mar. 2018, pp. 925–929.

[35] A. Dobson, K. Roy, X. Yuan, and J. Xu, ''Performance evaluation of machine learning algorithms in apache spark for intrusion detection,'' in *Proc. 28th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Nov. 2018, pp. 1–6.

[36] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, ''A detailed investigation and analysis of using machine learning techniques for intrusion detection,'' *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 686–728, 1st Quart., 2019, doi: 10.1109/COMST.2018.2847722.

[37] P. V. S. Alpano, J. R. I. Pedrasa, and R. Atienza, ''Multilayer perceptron with binary weights and activations for intrusion detection of cyber-physical systems,'' in *Proc. TENCON IEEE Region 10 Conf.*, Nov. 2017, pp. 2825–2829.

[38] B. S. Khater, A. W. Wahab, M. Y. Idris, M. A. Hussain, and A. A. Ibrahim, ''Lightweight perceptron-based intrusion detection system for fog computing,'' *Appl. Sci.*, vol. 9, no. 1, pp. 178–198, Oct. 2019, doi: 10.3390/app9010178.

[39] F. Maciá-Pérez, F. Mora-Gimeno, D. Marcos-Jorquera, J. A. Gil-Martínez-Abarca, H. Ramos-Morillo, and I. Lorenzo-Fonseca, ''Network intrusion detection system embedded on a smart sensor,'' *IEEE Trans. Ind. Electron.*, vol. 58, no. 3, pp. 722–732, Mar. 2011, doi: 10.1109/TIE.2010.2052533.

[40] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, ''Deep learning approach for intelligent intrusion detection system,'' *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[41] H. Yang, G. Qin, and L. Ye, ''Combined wireless network intrusion detection model based on deep learning,'' *IEEE Access*, vol. 7, pp. 82624–82632, 2019, doi: 10.1109/ACCESS.2019.2923814.

[42] T. H. Le, Y. Kim, and H. Kim, ''Network intrusion detection based on novel feature selection model and various recurrent neural networks,'' *Appl. Sci.*, vol. 9, no. 7, pp. 1392–1420, Apr. 2019, doi: 10.3390/app9071392.

[43] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, ''Machine learning and deep learning methods for cybersecurity,'' *IEEE Access*, vol. 6, pp. 35365–35381, 2018, doi: 10.1109/ACCESS.2018.2836950.

[44] C. Yin, Y. Zhu, J. Fei, and X. He, ''A deep learning approach for intrusion detection using recurrent neural networks,'' *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

[45] Y. Yang, K. Zheng, C. Wu, and Y. Yang, ''Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network,'' *Sensors*, vol. 19, no. 11, pp. 2528–2547, Jun. 2019, doi: 10.3390/s19112528.

[46] G. Thamilarasu and S. Chawla, ''Towards deep-learning-driven intrusion detection for the Internet of Things,'' *Sensors*, vol. 19, no. 9, pp. 1977–1995, Apr. 19, doi: 10.3390/s19091977.

[47] M. A. Khan, R. Karim, and Y. Kim, ''A scalable and hybrid intrusion detection system based on the convolutional-LSTM network,'' *Symmetry*, vol. 11, no. 4, pp. 583–596, Apr. 19, doi: 10.3390/sym11040583.

[48] T. Mouttaqi, T. Rachidi, and N. Assem, ''Re-evaluation of combined Markov-bayes models for host intrusion detection on the ADFA dataset,'' in *Proc. Intell. Syst. Conf. (IntelliSys)*, Sep. 2017, pp. 1044–1052.

[49] Y. Chang, W. Li, and Z. Yang, ''Network intrusion detection based on random forest and support vector machine,'' in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Jul. 2017, pp. 635–638.

[50] O. A. Alimi, K. Ouahada, and A. M. Abu-Mahfouz, ''Real time security assessment of the power system using a hybrid support vector machine and multilayer perceptron neural network algorithms,'' *Sustainability*, vol. 11, no. 13, pp. 3586–3603, doi: 10.3390/su11133586.

[51] O. Koucham, T. Rachidi, and N. Assem, ''Host intrusion detection using system call argument-based clustering combined with Bayesian classification,'' in *Proc. SAI Intell. Syst. Conf. (IntelliSys)*, Nov. 2015, pp. 1010–1016.

[52] A. O. Almashhadani, M. Kaiiali, S. Sezer, and P. O'Kane, ''A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware,'' *IEEE Access*, vol. 7, pp. 47053–47067, 2019, doi: 10.1109/ACCESS.2019.2907485.

[53] S. Otoum, B. Kantarci, and H. T. Mouftah, ''Detection of known and unknown intrusive sensor behavior in critical applications,'' *IEEE Sensors Lett.*, vol. 1, no. 5, pp. 1–4, Oct. 2017, doi: 10.1109/LSENS.2017.2752719.

[54] *Knowledge Discovery Data Mining Dataset*. Accessed: Oct. 21, 2020. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html

[55] *Knowledge Discovery Data Mining Dataset*. Accessed: Oct. 21, 2020. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[56] J. McHugh, ''Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory,'' *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000, doi: 10.1145/382912.382923.

[57] M. Mahoney and P. Chan, ''An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection,'' in *Proc. 6th Int. Symp. Recent Adv. Intrusion Detection (RAID)*, Pittsburgh, PA, USA, Sep. 2003, pp. 220–237.

[58] X. An, X. Zhou, X. Lü, F. Lin, and L. Yang, ''Sample selected extreme learning machine based intrusion detection in fog computing and MEC,'' *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–10, Jan. 2018, doi: 10.1155/2018/7472095.

[59] A. Sharma, A. K. Pujari, and K. K. Paliwal, ''Intrusion detection using text processing techniques with a kernel based similarity measure,'' *Comput. Secur.*, vol. 26, nos. 7–8, pp. 488–495, Dec. 2007, doi: 10.1016/j.cose.2007.10.003.

[60] *University New Mexico Intrusion Detection Dataset*. Accessed: Oct. 21, 2020. [Online]. Available: http://www.cs.unm.edu/~immsec/systemcalls.htm

[61] G. Creech and J. Hu, ''Generation of a new IDS test dataset: Time to retire the KDD collection,'' in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 1–6.

[62] G. Creech and J. Hu, ''A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns,'' *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 807–819, Apr. 2014, doi: 10.1109/TC.2013.13.

[63] *Australian Defense Force Academy Dataset*. Accessed: Oct. 21, 2020. [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-IDS-Datasets/

F. J. MORA-GIMENO received the Engineering degree from the Polytechnic University of Valencia, Valencia, Spain, and the Ph.D. degree in computer science from the University of Alicante, Alicante, Spain. Since 2002, he has been an Assistant Professor with the Department of Computer Technology and Computation, University of Alicante. His research interests include intrusion detection systems, network security, computer networks, distributed systems, and cloud computing. He has published numerous papers in the above areas.

H. MORA-MORA received the B.S. degree in computer science engineering and the B.S. degree in business studies from the University of Alicante, Spain, in 1996 and 1997, respectively, and the Ph.D. degree in computer science from the University of Alicante, in 2003. Since 2002, he has been a member of the Computer Technology and Computation Department, University of Alicante, where he is currently an Associate Professor and a Researcher with the Specialized Processors Architecture Laboratory. His work has been published in international journals and conferences, with more than 50 published papers. His research interests include computer modeling, computer architectures, high-performance computing, embedded systems, the Internet of Things, and cloud computing paradigm.

B. VOLCKAERT received the Master of Computer Science degree from Ghent University, in 2001, and the Ph.D. degree from Ghent University, in 2006, with a focus on data-intensive scheduling and service management for grid computing. He is currently a Professor of advanced programming and software engineering with the Department of Information Technology (INTEC), Ghent University, and a Senior Researcher with imec. He has worked on over 35 national and international research projects and is an author or a coauthor of more than 80 papers published in international journals and conference proceedings. His current interests include reliable and high-performance distributed software systems for City-of-Things (the Internet of Things (IoT) for smart cities), distributed decision support systems for UAVs, intelligent railway transportation applications, and autonomous optimization of cloud-based applications.

A. ATREY received the master's degree in computer science from the Vellore Institute of Technology (VIT), Vellore, India. She is currently pursuing the Ph.D. degree with the Department of Information Technology (INTEC), Ghent University, Belgium, and imec. She has internship experience from CNRS, France, and IIT Kanpur, India. At INTEC, she is working on research problems encircling intelligent resource provisioning in multi-tenant multicomponent applications. She has published her research in reputed cloud and service management journals and conferences, such as IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (TNSM), IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. Her research interests include cloud computing, resource scheduling and provisioning, data-placement, service management, and service-oriented architectures. She is serving as a Reviewer for CLOSER, *Journal of Network and Systems Management* (JONS), and IEEE TNSM.