IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 14, 2018, accepted January 14, 2019, date of publication January 22, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2893871

Introducing Deep Learning Self-Adaptive Misuse Network Intrusion Detection Systems

DIMITRIOS PAPAMARTZIVANOS[1], FÉLIX GÓMEZ MÁRMOL[2], AND GEORGIOS KAMBOURAKIS[1]

[1]Department of Information & Communication Systems Engineering, University of the Aegean, 83200 Samos, Greece
[2]Department of Information and Communications Engineering, University of Murcia, 30100 Murcia, Spain

Corresponding author: Dimitrios Papamartzivanos (dpapamartz@aegean.gr)

This work was supported in part by the Ramón y Cajal Research Contract (RYC-2015-18210) granted by the MINECO (Spain) and co-funded by the European Social Fund, and in part by the Leonardo Grant 2017 for Researchers and Cultural Creators awarded by the BBVA Foundation.

ABSTRACT The intrusion detection systems (IDSs) are essential elements when it comes to the protection of an ICT infrastructure. A misuse IDS is a stable method that can achieve high attack detection rates (ADR) while keeping false alarm rates under acceptable levels. However, the misuse IDSs suffer from the lack of agility, as they are unqualified to adapt to new and ''unknown'' environments. That is, such an IDS puts the security administrator into an intensive engineering task for keeping the IDS up-to-date every time it faces efficiency drops. Considering the extended size of modern networks and the complexity of big network traffic data, the problem exceeds the substantial limits of human managing capabilities. In this regard, we propose a novel methodology which combines the benefits of self-taught learning and MAPE-K frameworks to deliver a scalable, self-adaptive, and autonomous misuse IDS. Our methodology enables the misuse IDS to sustain high ADR, even if it is imposed on consecutive and drastic environmental changes. Through the utilization of deep-learning based methods, the IDS is able to grasp an attack's nature based on the generalized feature reconstructions stemming directly from the unknown environment and its unlabeled data. The experimental results reveal that our methodology can breathe new life into the IDS without the constant need for manually refreshing its training set. We evaluate our proposal under several classification metrics and demonstrate that the ADR of the IDS increases up to 73.37% in critical situations where a statically trained IDS is rendered totally ineffective.

INDEX TERMS Adaptive intrusion detection systems, artificial neural networks, deep learning, information systems security, MAPE-K, sparse auto encoders.

I. INTRODUCTION

Intrusion detection systems (IDSs) are one of the most important entities when it comes to Information and communications technology (ICT) infrastructure protection against cyberattacks. IDSs weaponize defenders with fundamental means to detect offensive events and consequently trigger optimal counteraction plans against them [1], [2]. In fact, the everlasting battle between defenders and attackers has taken the form of an ''arm race'', where both sides constantly upgrade their arsenals in order to prevail against each other. The emergence of new attacks spurs the academia and industry to investigate for novel methodologies which are able to closely monitor this race and adapt rapidly to the changes in the field.

In principle, IDSs fall into two major categories, namely *Anomaly Detection Systems* and *Misuse Detection Systems*. The former regulate their detection engine to identify as intrusive incidents those that exhibit deviations from a predefined normal behavioral profile. This kind of IDSs are able to identify previously unseen attacks, but are known to produce high false alarm rates, rendering them a questionable solution especially for complex infrastructures, where the standardization of the normal profile is challenging. On the other hand, misuse IDSs rely on known signatures trying to designate traffic instances to legitimate or attack traffic classes. This kind of IDS lacks the ability of identifying new attack patterns or deviations from known ones, and their performance depends on the freshness of the signatures database. Hence, the IDSs administrator needs to put significant effort to keep the misuse detection model up to date. If we additionally consider the fact that the protected environment is a dynamic ecosystem where new devices and/or services may

appear or leave the network at any moment (e.g., the Internet of Things), it becomes clear that the adaptability issue becomes a burden on administrators' shoulders. This burden becomes even heavier as the growth of communication networks pushes IDSs into the big data era, where the increased volume of the transmitted data surpasses the limits of human processing capabilities.

In view of the above, adaptive IDSs are becoming an active research field as new researches [3], [4] aim to address the inherent limitations of legacy intrusion detection systems. So far, interesting artificial intelligence-based methods that bear the feature of adaptability have been reported as promising approaches. To name a few, Learning Classifier Systems (LCS) [5], Artificial Immune Systems [6] and Swarm Intelligence [7] combine adaptation and evolution aspects. However, this research topic has still many challenges to face as systems and attacking tactics become more sophisticated.

Keeping any type of IDS up-to-date is a demanding task for several reasons where most of them pertain to the *environmental changes*. The latter term refers to any aspect of a network that can change and consequently affect the profile of the generated network traffic. In practice, the addition (or disengagement) of a device in a network can affect different network aspects, including the topology, the running services, the open ports, the communication protocols and/or applications, the network traffic load, and others. In turn, these environmental changes affect fundamental security features such as the vulnerabilities appearing in the network, which can generate multiple penetration paths for the attackers. Considering a more dynamic network like an IoT environment, an Ad Hoc network, or even a corporate network with a Bring your own device (BYOD) policy applied, one can understand that the attack surface of the network can be increased unexpectedly. It is plausible that, say, the newly introduced device might be already infected by a malware and act as a stepping stone for an attacker to conquer more assets within the network. Yet, new devices are not the only enemies of an IDS in a network, as also already installed devices will eventually proceed with software/OS updates or new software installations that again will bring in alterations in the environment.

Overall, the above mentioned changes are routine actions that constantly appear in every common network, rather than unusual events. Actually, virtually all sort of modifications can significantly affect the performance of an IDS which is placed to protect an ever-changing infrastructure. This reality, combined with the lack of adaptable detection engines, forces the IDS to become quickly outdated and inadequate as it inevitably has to operate in new and ''unknown'' or unforeseen environments for which its engine was not trained. Thus, security administrators undertake the task of constantly retraining the IDS by considering all the new environmental changes to regain the reliability and the performance of the detection system. All in all, the cardinal challenge for any IDS designer, namely find proper ways to automatize at least to a certain degree the retrain process, remains largely unsolved.

To this end, this work proposes a novel adaptive methodology which can significantly boost the performance of a misuse IDS when it is dragged into new, previously unseen environmental states. Our novel solution brings in intelligence in the detection engine update process with the aim of extending its lifetime and sustain the detection ratio above considerably higher levels than it would reach without such intelligence. At least, in network setup transition periods, this ability gives the necessary time to the administrators to smoothly retrofit the IDS to fully meet the new environmental conditions. To do so, we take advantage of the benefits of the Self-Taught Learning (STL) methodology [8], for enabling Transfer Learning from unlabeled data for the sake of assisting the IDS when dealing with unknown environments. Our evaluation proves that the qualities of the STL methodology can fit well in the particular problem and address the challenges raised in the field of adaptive IDSs. Our adaptive methodology is also supported by the MAPE-K model [9] for delivering a self-adaptive IDS that follows the sound practices of autonomic computing.

In short, the contributions of the work at hand are as follows:

- We propose a novel methodology for designing a scalable, self-adaptive and autonomous misuse intrusion detection systems based on advanced artificial intelligence (AI) techniques.
- We take advantage of deep learning methodologies to identify new data feature representations that stem from the unknown environment where the IDS operates. These new representations are used to retrain the IDS in an automated way so as to adapt to the new environment.
- We integrate our proposal in the context of MAPE-K methodology that draws the frame for autonomous and self-adaptive systems.
- We extensively evaluate our system over several metrics and diverse environmental states to deliver a proof of concept, which is supported by experimental results and demonstrates its potentiality for further extension.

The rest of the paper is organized as follows. The next section includes all the necessary information to introduce the reader in our methodology. In Section III we present our methodology and elaborate on its beneficial characteristics, while in Section IV we provide the evaluation results. Section V provides a discussion on the key findings. Section VI reviews the related works in the field. The last achieved section concludes and provides pointers to future research.

II. PRELIMINARIES

Our work recruits two different concepts to provide a holistic framework for self-adaptive and autonomous misuse detection systems. Before introducing the reader to our idea, we first provide an overview of the basic concepts related to our methodology. That is, the following subsections elaborate on the MAPE-K [9] and Self-Taught Learning [8] methodologies.

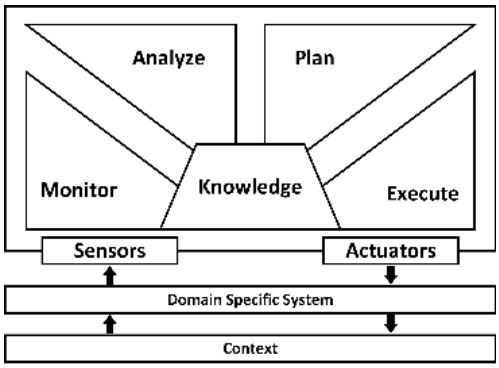

FIGURE 1. MAPE-K methodology sets the principles for an adaptive and autonomous IDS.

A. MAPE-K CONTROL LOOP

MAPE-K control loop is a reference model in autonomic computing firstly introduced by IBM [9]. Since then, MAPE-K is used to set the principles for self-adaptive systems and significant effort has been put to standardize and formalize the methodology [10], [11]. MAPE-K is a control loop comprised of five activities, namely, *Monitor, Analyze, Plan, and Execute* over a *Knowledge* base, as can be seen in FIGURE 1. As further explained in subsection III-B, these activities provide a general framework for developing self-adaptive systems which are able to sense changing events from their environment. This is done via the use of specialized *Sensors*, and eventually harmonize their behavior by taking actions through *Actuators*.

More specifically, the *Monitor* senses the environment and collects data/events of interest where their presence indicates the need of system adaptation. The collected data/events are gathered in the *Knowledge* database for later reference. Next, the *Analyzer* undertakes the task of processing the collected events to identify patterns of failure or critical events and act upon by initiating a proper adaptation strategy. The *Plan* activity orchestrates the decision making and determines the changes which should be taken for keeping the system aligned with its objectives. Finally, the *Executor* instructs the appropriate alterations to the system through the *Actuators*. The control loop described above is initiated whenever the system identifies the need to adapt its behavior to the underlying environment and always aims to meet the objectives which - in principle - are set by the administrator.

Every single activity in the MAPE-K control loop can contain other autonomic elements that can be used to fulfill sub-objectives of the main activities. These elements can interact among each other by exchanging signals and messages. Overall, the system and its sub-components are coordinated with the aim of providing a fault tolerant self-adaptive system, which is driven by pre-determined objectives.

B. SELF-TAUGHT LEARNING

Self-Taught Learning (STL) is a machine learning framework which is able to exploit unlabeled data with the purpose of improving a supervised classification problem [8]. The motivation of the authors in [8] derives from the fact that labeled data are an ''expensive'' source of information, as it requires a significant investigation budget to acquire and update them. The question posed was whether unlabeled data could be used to improve a given classification task.

In the STL concept, one is provided with both labeled and unlabeled data. The labeled data are used as the initial training set of m samples for a given classification task $T = \{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), ..., (x_l^{(m)}, y^{(m)})\}$, where $x_l^{(i)} \in \mathbb{R}^n$ is the i-th sample with n features, $y^{(i)} \in \{1, ..., C\}$ is the class label, and the l symbol stands for ''labeled''. On the other hand, the set of k unlabeled samples $U = \{x_u^{(1)}, x_u^{(2)}, ...x_u^{(k)}\} \in \mathbb{R}^n$, where $x_u^{(i)} \in \mathbb{R}^n$ is the i-th unlabeled sample with n features, and u stands for ''unlabeled''. U is given as input to a sparse coding algorithm to learn a higher level structure of those data. This structure is then used as a basis to transform the initial labeled dataset T and obtain a new training set $\hat{T} = \{(a_l^{(1)}, y^{(1)}), (a_l^{(2)}, y^{(2)}), ..., (a_l^{(m)}, y^{(m)})\}$, where $a_l^{(i)}$ represents the i-th new training example. In consequence, the new training dataset $\hat{T}$ can be used to train a supervised learning algorithm.

More specifically, sparse coding is a type of unsupervised methods that aims to reconstruct input data as accurately as possible and express them as linear combinations of a basis vector b. The basis vector b enables to accurately capture the inherent information of the input and identify strong patterns in it. Additionally, sparse coding regulates the sparsity of the data by using coefficients (or activations) a_i and encourages most of the coefficients to be zero. In fact, this is an optimization problem that aims to reconstruct the input data by minimizing the reconstruction error, and at the same time to maximize the sparsity of the output. Given the bases b and the training set T, the STL algorithm transforms the inputs $x_l^{(i)}$ to sparse non-linear combinations of the basis b to form a new, but more informative, training set $\hat{T}$.

The interested reader can refer to [8] for more details regarding the STL method. In the following section, we further elaborate on the beneficial features of the STL and explain how these features in conjunction with MAPE-K control loop can deliver a holistic methodology for building a misuse adaptive IDS.

III. PROPOSED METHODOLOGY

This section details on our methodology for adaptive misuse IDSs. With reference to Section I, we first define the term ''environmental state'' and elaborate on the challenges that IDSs face whenever a change appears in the underlying network topology. Next, we combine the two methodologies described in subsections II-A and II-B for the purpose of presenting our full-fledged approach along with its advantages.

A. ENVIRONMENTAL STATES AND NETWORK FLOWS

Undoubtedly, computer networks are highly volatile environments, which can be characterized as a mosaic of diverse interconnected devices usually from different vendors. On top

of that, one needs to consider that modern networks like IoT networks, Wireless Sensor Networks (WSN), VANETs, and others come with such an extended size, dynamics and complexity that far exceed the limits of human managing capabilities. Similarly, IDSs which aim to protect such dynamic networks are unable to automatically adapt to the changes occurring to their environment and their adjustment requires significant human effort. In fact, there are several reasons that can lead a network to a new state, and therefore lower the efficiency of the deployed IDS.

At any given time, a new device can join a network. This is the most common situation, especially in wireless networks, but this event alone can lead to a set of subsequent events that can bring instability. That is, the newcomer may be a host of new services, ports, applications, communication protocols, communication patterns, network workload, and even new vulnerabilities. In the worst case scenario, the new device may be also infected by some ilk of malware that can attempt to exploit other network assets to penetrate the network. A join operation is not the only reason for facing network state changes. The already existing network assets can modify their operational profile as they can be subjected to OS/application upgrades and new application installations. These modifications, apart from introducing changes in the network state, can sometimes increase the attack surface of the network as they can bring in new and zero-day vulnerabilities. The aforementioned changes, even when occurring individually, can significantly affect the decision engine of an IDS and produce a high level of false alarms. Even worse, if the IDS is not retrained to deal with these changes it can become the single point of failure of the infrastructure's security planning.

In the context of our work, we perceive any of the above mentioned events to lead the network into a new state, and thus affect the IDS's operational environment. Such changes also affect the network's behavioral profile, which in turn is reflected in the network flows. According to RFC 2722 [12], a network flow can be seen as an artificial logical equivalent to a call or connection, which has as attribute values aggregated quantities which reflect the events that take place during this connection. These attribute values can bear valuable information regarding numerous aspects of the network's behavior ranging from the topology to the workload and the active services. Thus, network flows are a rich source of information that can improve the network security visibility as they can be leveraged by security analysts to identify and assess hostile actions, new attacks, and the network's security state in general. As a result, when a network is overwhelmed by unknown and previously unseen network flows, an IDS which has been trained to defend a network based on a static training set needs to be retrained in order to sustain a credible security level. This however implies the need of a demanding process on behalf of the security analyst to identify and label manually new network instances for creating a new dataset that can be used to retrain the IDS. Considering that most of the network changes are common actions that can happen regularly, it becomes clear that there is a need for methods capable of automating the retraining process.

To this end, our methodology aims to offer an automated way to keep the detection ratio of a misuse IDS to acceptable levels regardless of the environmental changes that may indicate the presence of previously unknown attacks. In a nutshell, our methodology can empower autonomous and self-adaptive misuse IDSs by enabling them to adapt to their environment and significantly contribute in keeping a high or at least acceptable security level. This quality also significantly alleviates security experts from the demanding task of retraining the IDS.

B. BLENDING MAPE-K AND SELF-TAUGHT LEARNING

As described in Section II, MAKE-K is a reference model to build autonomous and self-adaptive systems. This subsection details on the ways the benefits of MAPE-K and STL can co-work toward coping with the challenges of this particular field and building a solid basis for misuse adaptive IDSs.

As observed from FIGURE 1, MAPE-K comprises 5 activities that operate over a *Domain Specific System (DSS)* and a *Context*. In our case, the DSS is the IDS per se, while the Context can be adjusted to any given type of network where there is a need of an adaptive IDS. FIGURE 2 depicts the proposed system and details on its components, which are described next.

1) MONITOR

This activity undertakes the task of coordinating the sensors for acquiring the basic knowledge that will reveal the need of triggering the adaptive control loop. Network mappers can be used as the basic sensors for network inventory. Such entities are able to determine a gamut of network characteristics, including its topology, the available hosts, the running services, open ports, the operating systems, and even potential vulnerabilities. By collecting such information, the *Monitor* is able to determine any alteration event that requires an IDS adaptation. The monitoring activity is able to grasp the environmental changes in collaboration with the Knowledge activity, which serves as a repository for reference purposes. The *Monitor* can schedule the network mapping process to occur periodically according to the characteristics of the network.

In parallel, another sensor type which is controlled by the *Monitor* is the Network Sniffers. The latter are used to capture the network traffic. This traffic is used as the basis to extract in a later stage the network flows which have to pass through the detection engine of the IDS. Additionally, the network traffic is stored in the *Knowledge* component to serve the purpose of adaptivity as it is described further down in the Plan/Execute activities.

2) ANALYZE

After collecting the necessary data, the *Analyzer* performs the transformation of the raw network traffic into network flows. By using the stored traffic of the knowledge

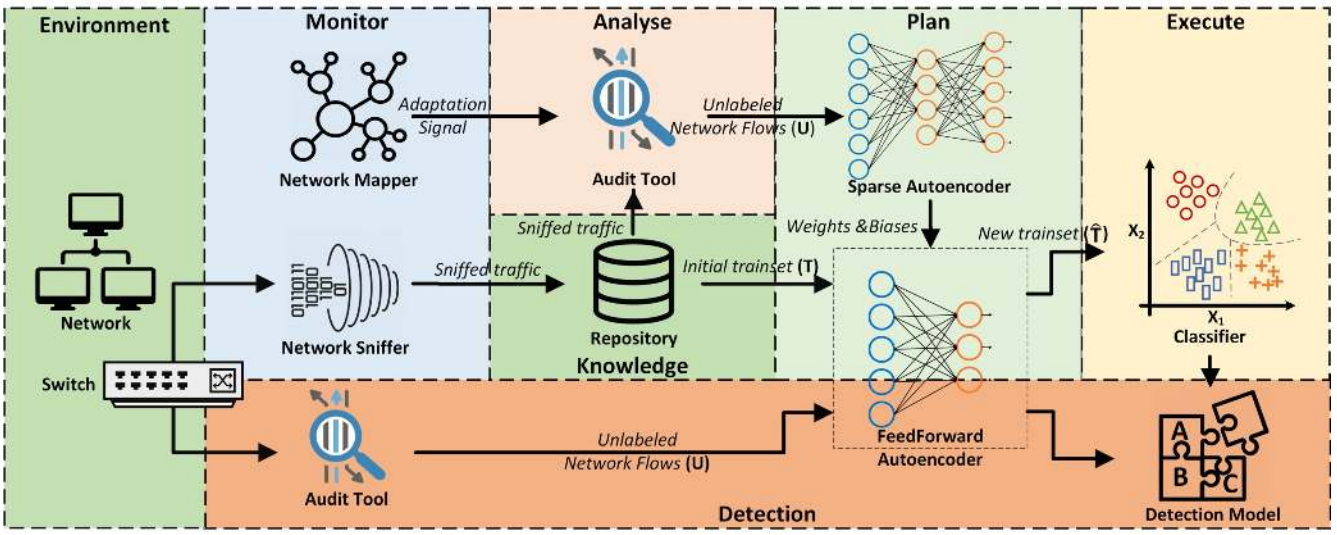

FIGURE 2. Architectural overview of the proposed system.

component, the *Analyzer* utilizes network audit tools such as Argus [13] and CICFlowMeter [14] in order to generate the network flows. These tools are able to analyze large amounts of network traffic even in an in-line manner and process them accordingly to generate highly informative network flows with various features. These features comprise the machine learning features of the network traffic instances, which are fed into the IDS engine for detection purposes. Note that these flows constitute the unlabeled dataset, which on the one hand are given into the supervised model of the IDS to detect potential attacks, while on the other are used as the unlabeled data fed to the STL to fuel the adaptive process. This implies that during the IDS operation, the adaptive process is simultaneously executed with the aim of coming up with a new detection model that will replace the existing one.

3) PLAN

The planning activity undertakes the key process of leveraging the unlabeled data for initiating the machine learning adaptive process. Until that point, the *Monitor* and the *Analyzer* identified environmental changes in the network, while the *Knowledge* component consolidated the network flows, which were generated by the time that the change(s) occurred. This moment is the beginning of a crucial time interval when the IDS may face unknown network traffic instances that can undermine its performance. In this direction, the *Plan* activity aims to cope with this ambiguity by utilizing unsupervised feature learning techniques. In the context of our work, we utilize a Sparse autoencoder as the unsupervised learning algorithm to learn informative and sparse new representations of the unlabeled data, and thus benefit the supervised task of the misuse IDS.

a: Sparse autoencoder

An autoencoder is a neural network that applies backpropagation [15] and aims to reconstruct a given input to an output that approximately resembles to the initial input. That is, the neural network attempts to learn a function $h_{W,b}(x) \approx x$, where the W, b vectors denote the Weights and Biases among the layers and their units of the neural network. This process can be driven also by other objectives apart from minimizing solely the reconstruction error. As already pointed out, in our work we utilize a Sparse autoencoder in order to learn sparse representations of the input data. In practice, the backpropagation process is driven by the following cost function.

$$\begin{aligned} J(W,b) = &\frac{1}{k}\sum_{i=1}^{k}\left(\frac{1}{2}\|x_u^{(i)} - \hat{x}_u^{(i)}\|_2^2\right) \\ &+ \frac{\lambda}{2}\sum_{L=1}^{2}\sum_{i=1}^{s_L}\sum_{j=1}^{s_{L+1}}\left(W_{ji}^{(L)}\right)^2 \\ &+ \beta\sum_{j=1}^{s_2} KL(\rho\|\hat{\rho}_j) \end{aligned} \tag{1}$$

where:

- $x_u^{(i)} \in \mathbb{R}^n$ is the i-th input unlabeled example.
- $\hat{x}_u^{(i)} \in \mathbb{R}^n$ is the i-th output given the i-th input example.
- k is the number of the examples in the unlabeled training set.
- λ is the weight decay parameter.
- L index denotes the number of a layer.
- s_L is the number of nodes in the L-th layer.
- β is the weight of the sparsity penalty.
- $\|x_u^{(i)} - \hat{x}_u^{(i)}\|_2^2$ is the squared L^2 norm.

Through backpropagation the Sparse autoencoder aims to minimize equation (1). As can be seen, the equation consists of three terms (one per line). The first term represents the average accumulated squared error among the input and the reconstructed output. That is, the network tries to reconstruct the output and achieve high similarity with

the input. Note that, $\hat{x}_u^{(i)} \in \mathbb{R}^n$ is a vector of n features, i.e., $\hat{x}_u^{(i)} = \{\hat{x}_{u_1}^{(i)}, \hat{x}_{u_2}^{(i)}, \ldots, \hat{x}_{u_n}^{(i)}\}$. Thus, each $\hat{x}_{u_p}^{(i)} = h_{W,b}(x_{u_p}^{(i)}) = f\big(\sum_{j=1}^{s_2} W_{pj}^{(2)} a_j^{(2)} + b_p^{(2)}\big)$, $p = 1, \ldots, n$ and $a_j^{(2)}$ are the activations of the hidden units (2nd layer). The sigmoid function $f(z) = \dfrac{1}{1+\exp(-z)}$ has been chosen as the activation function for the neurons. This activation function gives values between 0 and 1, while it regulates the weights of the network to change gradually and output better results. Additionally, the sigmoid function introduces non-linearity into the model, thus aiding in capturing non-linear combinations of the input data. The second line refers to the weight decay term that tries to decrease the magnitude of the weights ($W_{ji}^{(L)}$) among the nodes of the layers, while λ controls the importance of the weight decay term. The last term is a function that applies the sparsity penalty, where $KL(\rho \| \hat{\rho}_j) = \rho \log \dfrac{\rho}{\hat{\rho}_j} + (1-\rho) \log \dfrac{1-\rho}{1-\hat{\rho}_j}$ is the Kullback-Leibler (KL) divergence that can determine the difference between two distributions having ρ and $\hat{\rho}_j$ mean values respectively. That is, ρ defines a desired level of sparsity, while $\hat{\rho}_j$ is the average activation of the j-th hidden unit. The magnitude of the sparsity penalty is regulated by the β weight.

b: Feedforward autoencoder

The training process of the Sparse autoencoder will define the Weights and Biases vectors $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$. Next, these vectors can be used in a Feedforward manner over a new input for finding a new and more informative structure of this input. In other words, the knowledge acquired from the unlabeled data U that fed into the Sparse autoencoder can now be exploited for restructuring another dataset. This reconstruction is driven by a new representation which is learned out from unlabeled data, i.e., data that stem from an unknown environment.

Following this principle, our methodology can generate a new representation of the labeled dataset $T = \{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), \ldots., (x_l^{(m)}, y^{(m)})\}$, that was initially used to train the IDS. Note that, $x_l^{(i)} \in \mathbb{R}^n$ is a vector of n features, i.e., $x_l^{(i)} = \{x_{l_1}^{(i)}, x_{l_2}^{(i)}, \ldots, x_{l_n}^{(i)}\}$. The new representation is a new labeled training set $\hat{T}$ that has as features the activations of the hidden units. That is, given T as the new input in a Feedforward autoencoder, we can calculate the new activation vectors using the Weights and Biases of the first layer $W^{(1)}$, $b^{(1)}$ by applying the activation function. As a result, the system produces a new dataset $\hat{T} = \{(a_l^{(1)}, y^{(1)}), (a_l^{(2)}, y^{(2)}), \ldots., (a_l^{(m)}, y^{(m)})\}$, where $a_l^{(i)}$ represents the i-th new training example. Thus, each $a_l^{(i)}$ example is a vector of s_2 activations, $a_l^{(i)} = \{a_{l_1}^{(i)}, a_{l_2}^{(i)}, \ldots., a_{l_{S_2}}^{(i)}\}$, and each activation is given as follows:

$$a_{l_p}^{(i)} = f\Big(\sum_{j=1}^{s_1} W_{pj}^{(1)} x_{l_j}^{(i)} + b_p^{(1)}\Big), \quad p = 1, \ldots, s_2 \qquad (2)$$

Finally, the new training dataset $\hat{T}$ can be used to train a supervised learning algorithm. Note that, the acquisition of $\hat{T}$ can occur virtually indefinitely as long as the planning activity is fed with new unlabeled network traffic.

The reader may notice that the Feedforward autoencoder adds an extra layer of data transformation. That is, any instance which will be subjected into the supervised learning algorithm for detection purposes needs to pass first through the Feedforward autoencoder to acquire the same transformation properties.

4) EXECUTE

The outcome of the planning activity is a Feedforward autoencoder which is used for reconstructing the initial labeled dataset T and acquire $\hat{T}$. Hence, the execution activity undertakes the training of a supervised model based on the new dataset $\hat{T}$. This step does not impose any constrains regarding the supervised algorithm that can be used to empower the detection model. In our case, we make use of Softmax Regression to deliver a multi-classification detection module. After training the new model, the old one, which due to the environmental changes had started facing efficiency problems, can be replaced through the actuators.

5) KNOWLEDGE

During the adaptive control loop, the *Knowledge* component is accountable for storing purposes. In fact, *Knowledge* is a repository that supports the adaptive functions and helps exchanging the inputs and outputs of each activity among them. More specifically, the repository stores the sniffed network traffic during the monitoring phase. Upon the adaptation signal of the network mapper, these captures will become the input of the network audit tool for generating the network flows. Additionally, the repository holds the initial labeled dataset T, which is used as a basis every time the adaptive control loop is triggered.

C. DISCUSSION

FIGURE 2 provides a high-level view of the proposed system by highlighting the building blocks of the MAPE-K adaptive control loop. Additionally, in the figure, we can observe the interconnections among the diverse components and follow the flow of the system's actions. As already pointed out at the beginning of this subsection, the combination of MAPE-K and STL serves as the basis for building adaptive and autonomic misuse IDSs. That is, while MAPE-K provides the essential principles to realise such an IDS, STL contributes several features that cover the missing parts of the misuse IDS adaptation puzzle.

- STL is destined to utilize unlabeled data with the aim of improving a supervised learning task. This feature fits directly in the nature of the problem. Once a misuse IDS is powered, it faces unlabeled network instances and tries to classify them. Inevitably, due to environmental changes, the statically trained IDS will face efficiency issues. Nevertheless, since STL is able to capture informative structures from unlabeled data, the

ambiguity of the new environment is exploited to generate new knowledge. This, in turn, will reinforce the misuse IDS by generating a new representation of the initial dataset.

- STL cannot be seen as an unsupervised feature learning or a semi-supervised technique, but as a more powerful setup. This is because the unlabeled data x_u can: 1) be of any class and not necessarily coincide with the classes of the labeled data x_l, and 2) be drawn from a different distribution from the labeled data x_l. Crucially, these are two essential qualities that advocate the suitability of STL as in practice, in an unknown environment, an IDS will face both known and unknown attacks, which both stem from different distributions. Thus, our methodology guarantees the adaptability and autonomy of the misuse IDS.
- When put in the frame of MAPE-K, the characteristics of STL can significantly alleviate the burden of retraining the IDS every time a change appears in its environment. For the retraining process, one needs also to consider the significant effort required to assign labels by hand to Big Data such as network data. That is, if the IDS is trained with a basic labeled dataset, the proposed methodology can significantly extend its autonomy. Though, a reasonable requirement is that the basic labeled dataset needs to be representative enough and this requirement is delegated to the administrator. It is to be stated that we do not claim that our approach eliminates completely the need for human intervention, but it can significantly diminish it.
- STL is able to handle big data in a scalable manner. In fact, the more unlabeled data are given as input to the autoencoder, the more informative will be the new representation.
- STL can significantly uncover strong structures in the data, especially when the data features are statistically correlated.

Overall, the aforementioned characteristics of the combination of STL and MAPE-K address to a great extent the major limitation of misuse IDSs, namely their inability to deal with unknown situations. Considering the fact that misuse IDS are widely used over anomaly detection IDSs, our approach becomes even more impactful.

IV. EVALUATION

In this section we evaluate our novel methodology. More specifically, we define the metrics used to evaluate the performance of the adaptive misuse IDS, we detail on the used dataset and the setup of our evaluation experiments.

A. EVALUATION METRICS

To highlight the advantages of our proposal and to make it comparable to other approaches, we present our results under a variety of legacy classification metrics. More specifically, we use the following metrics.

a: Accuracy

This metric measures the frequency of correct decisions. It is a fraction of the correct decisions made among all the classes (C) (true positives, or TPs) divided by the total number of instances in the dataset (N).

$$Accuracy = \frac{\sum_{i=1}^{C} TP_i}{N} \tag{3}$$

b: Mean F-Measure (MFM)

It is used to measure the balance between the precision and the recall. In the case of a multi-classed problem, this metric is calculated by the following formulas:

$$MeanFMeasure = \frac{\sum_{i=1}^{C} FMeasure_i}{C} \tag{4}$$

$$FMeasure_i = \frac{2 \cdot Recall_i \cdot Precision_i}{Recall_i + Precision_i} \tag{5}$$

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{6}$$

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{7}$$

where:

- FP_i, or false positives, represent the number of instances with actual class other than the i-th, but wrongly predicted to belong in the i-th class.
- FN_i, or false negatives, represent the number of instances with i-th being the actual class, but falsely predicted to belong to another class.

c: Average Accuracy

It is calculated as the average recall among all the classes of the dataset.

$$AvgAccuracy = \frac{1}{C} \sum_{i=1}^{C} Recall_i \tag{8}$$

d: Attack Accuracy

This metric is used to measure the ability of a model to detect solely the attack classes by neglecting the normal traffic. Index $i = 1$ stands for the normal traffic class.

$$AttackAccuracy = \frac{1}{C-1} \sum_{i=2}^{C} Recall_i \tag{9}$$

e: Attack Detection Rate (ADR)

It stands for the accuracy rate for the attack classes.

$$ADR = \frac{\sum_{i=2}^{C} TP_i}{\sum_{i=2}^{C} TP_i + FP_i} \tag{10}$$

f: False Alarm Rate (FAR)

This metric focuses on the normal traffic and quantifies the FNs, i.e., normal instances misclassified as attacks. Index $i = 1$ stands for the normal traffic class.

$$FAR = \frac{FN_1}{TP_1 + FN_1} \tag{11}$$

TABLE 1. Normal and attack classes in KDDCup'99 and NSL-KDD.

Class	KDDCup'99 and NSL-KDD subclasses and the number of instances	#Instances
Normal	Normal traffic is not divided into sub-classes	936,152
DoS	back (2,633), neptune (297,085), smurf (6,688), teardrop (1,828), land (46), pod (448), apache2 (1,531), mailbomb (601), processtable (1,429), udpstorm (4)	312,293
PRB	satan (10,226), portsweep (6,787), ipsweep (7,411), nmap (3,200), mscan (2,044), saint (683)	30,351
R2L	ftp_write (22), warezclient (1,783), spy (4), named (34), warezmaster (1,986), multihop (50), xsnoop (8), sendmail (29), snmpguess (690), imap (25), snmpgetattack (357), worm (4), xlock (18), phf (12), guess_passwd (2,639)	7,661
U2R	buffer_overflow (102), httptunnel (278), loadmodule (22), perl (10), rootkit (46), xterm (26), ps (31), sqlattack (4)	519
Total	40 subclasses	1,286,976

B. KDDCUP'99 AND NSL-KDD DATASETS

KDDCup'99 [16] and NSL-KDD [17] constitute well-known benchmarks for evaluating whether an IDS is able to detect network abuses. In fact, NSL-KDD is a newer version of the KDDCup'99 that deals with some of its shortcomings [17]. Despite KDDCup'99 limitations, it is still considered a standard and it is used by recent studies in the field [18], [19]. This work does not aim to evaluate a detection algorithm per se. Instead, we aim to prove that our methodology is able to exploit unknown network flows to boost the detection efficiency in ambiguous environments. This means that the inherent limitations of KDDCup are orthogonal to our testbed.

Precisely, in the context of this work, we merged all the datasets provided by KDDCup'99 and NSL-KDD for creating a single voluminous dataset that bears as many network traffic instances and as many attack classes as possible. TABLE 1 presents the instances of the used dataset. In total, the compiled dataset has of proximately 1.3 million network instances and comprises 40 classes (1 normal + 39 attacks), which come under different probability distributions and fall into the following 5 major categories:

- **Normal**: Normal traffic instances.
- **DoS**: Denial of Service.
- **PRB**: Probing - Surveillance and other means of probing.
- **R2L**: Remote to Local - Unauthorized access from a remote machine.
- **U2R**: User to Root - Unauthorized access to local superuser (root) privileges.

We removed all duplicates from the merged dataset to avoid any bias to the classification end-model. KDDcup dataset has some beneficial characteristics. To begin with, its variety in attack classes seems ideal for evaluating our methodology. Its 39 attack classes can be used to emulate a realistic and challenging testbed, where an IDS has to face unknown traffic instances every time an environmental change occurs. Additionally, the fact that the attack instances are drawn from different probability distributions directly challenges the STL method. Recall that according to its properties, STL is destined to handle efficiently large amounts of unlabeled data with that exact property. Moreover, KDDCup dataset has an imbalanced number of instances among its classes and this feature reflects a realistic network condition. Finally, the KDDCup dataset created over a network experiment that lasted for 9 weeks and the final result was a dataset of approximately 7 million network instances with duplicates. Our compiled dataset consists of 1.3 million instances without duplicates. This implies that our dataset corresponds to a data collection period of at least 12 days. Hence, apart from the beneficial characteristics mentioned above, the compiled dataset comprises a realistic collection of network traffic that spans adequately over time and it is thus suitable for evaluating an adaptive mechanism.

C. TESTBED AND PARAMETERS

To emulate an ever-changing environment for the adaptive IDS, we came up with the following strategy. To emulate the initial state of the IDS, we train the IDS using Softmax Regression with an initial dataset T, which consists of a fraction of 10% of normal traffic and a randomly chosen subset of attack traffic. This attack-focused subset consists of 3, 3, 3 and 4 attacks subclasses of the major classes DoS, PRB, U2R, and R2L respectively. As it is the case with any legacy machine learning-based IDS, we cross-evaluate the IDS for achieving a robust end-model of more than 99% prediction accuracy. In a legacy situation, this end-model would be the one to defend any future environmental state of the network. Consequently, for emulating a new environmental state, we randomly select another piece of the dataset U which consists of 10% of normal traffic and 5, 5, 5, 8 attacks subclasses of the major classes DoS, PRB, U2R and R2L, respectively. U *might or might not* contain the classes or the instances gathered in T. Hence, depending on the divergence between T and U the new environment can be slightly or to very different from the initial one. That is, it is expected to witness a low or even high drop of the IDS efficiency respectively. However, according to the proposed methodology, the adaptive IDS is able to exploit the U in order to obtain a better representation of the new environment by transforming the initial dataset T, and thus resisting to this efficiency drop. Note that U is in practice an unlabeled set of instances which is fed to the detection engine for prediction. Naturally this applies also in our case, but we are beforehand

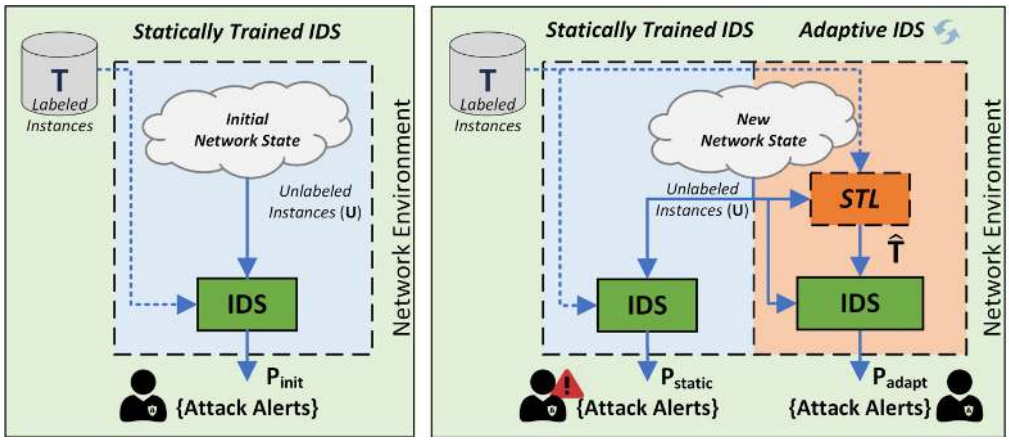

FIGURE 3. In the initial network state, a statically trained IDS can achieve an acceptable performance P_{init} (left). In new network states, the adaptive IDS has the ability to sustain an acceptable performance in contrast to the statically trained IDS. The goal of our methodology is to improve the efficiency of the IDS so that $P_{adapt} > P_{static}$ (right).

TABLE 2. Parameters' setup.

Parameter	Value
Sparse autoencoder	
• Number of hidden neurons	30
• Number of optimization iterations	500
• λ	0.0008
• ρ	0.06
• β	3
Softmax Regression	
• Weight decay parameter	0.0001
• Number of optimization iterations	100

aware of the hidden classes of the instances for being able to measure the efficiency of the adaptive IDS contrary to the static one. If we denote the performance of the IDS in the initial phase, in the new but static phase, and after the adaptation as P_{init}, P_{static}, P_{adapt} respectively, the goal of our methodology is to improve the efficiency of the IDS so that $P_{adapt} > P_{static}$. The evaluation strategy described in this section is presented intuitively in FIGURE 3.

As described in Section III, the *Planning* activity is based on the Sparse autoencoder for identifying the new structure out of the unlabeled data. The autoencoder has to be tuned beforehand by the security administrator. This process requires an initial tuning period on behalf of the administrator that relies basically on the nature of the network and the utilized data features. In our case, we selected the testbed's parameters (TABLE 2) based on an empirical study and our knowledge on the domain of the problems.

D. RESULTS

This subsection elaborates on the results and compares the performance of the adaptive IDS against the static approach, based on the metrics presented in subsection IV-A. In total, we subjected the IDS to 100 environmental changes, i.e., 100 diversely compiled datasets, blended with diverse attacks of each major class of the compiled KDDCup dataset. Figures 4 to 8 provide a graphical representation of the recorded metrics over the 100 environmental states.

As can be seen in FIGURE 4, our adaptive approach surpasses the static one in most of the environmental states. More specifically, in 84% of the states the adaptive approach achieved a higher accuracy score compared to the static one. The average accuracy of the static approach is 59.71%, while the adaptive's one is 77.99%. This means that in average our approach performs better by 18.28% over the 100 unknown states. Additionally, the standard deviation is 30.79% and 18.78% for the static and the adaptive approaches respectively. This fact quantifies what intuitively can be observed from FIGURE 4, where the adaptive curve witnesses less and smaller efficiency drops over the vast majority of the states. It is important to note that the maximum positive accuracy difference between the two approaches is 56.92% (state #8), while the maximum negative difference is -1.6% (state #36). In fact, as can be seen in FIGURE 4, in critical cases where the IDS accuracy drops significantly due to a state's high deviation with respect to the initial training set (T), the adaptive methodology demonstrates a significantly higher contribution that can sustain the IDS to acceptable detection levels. In total, 38% of the states present higher accuracy difference than the standard deviation (18.77%), and for these states, the average accuracy is increased by almost 48%. Hence, we can safely argue that the adaptive approach can significantly contribute to the overall security level in sudden network environmental changes (including attack incidents), while in cases where the IDS accuracy drops to some extent, the adaptive approach achieves almost the same performance as the static one. More precisely, for those 46 states where the accuracy difference per state is positive and less than the standard deviation, the average accuracy of the adaptive approach is greater than the static one by 0.51%. However, only for the 16 out of 100 states where the static approach performs better than the adaptive one, the average performance is 0.57% in favour of the static approach. All in all, the adaptive approach greatly outperforms the static one especially when it comes to critical states.

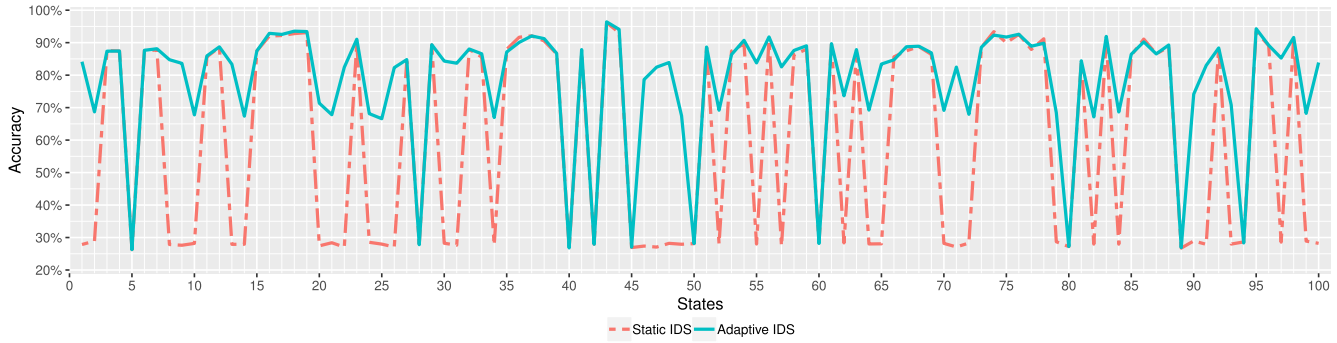

FIGURE 4. Deviation of IDS accuracy over 100 consecutive environmental states.

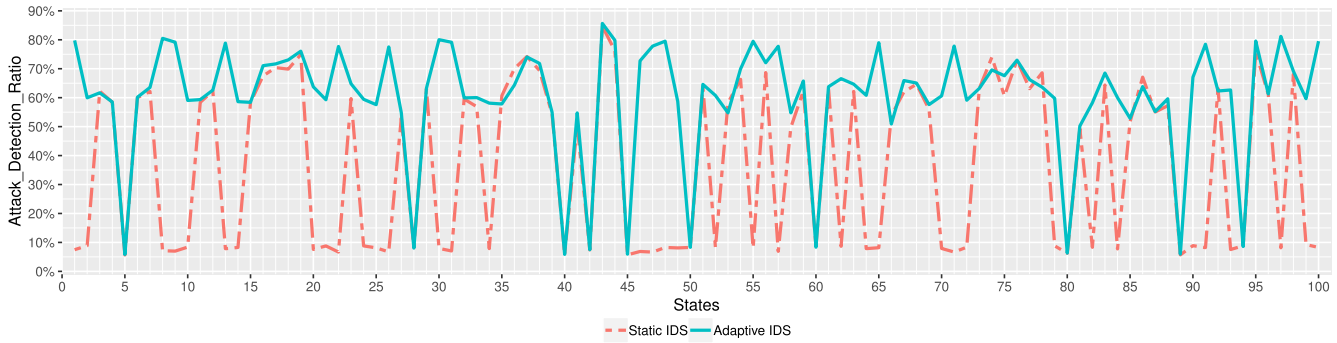

FIGURE 5. Deviation of IDS Attack Detection Ratio (ADR) over 100 consecutive environmental states.

FIGURE 5 presents the ADR performance over the 100 environmental states. Recall from subsection IV-A that ADR measures the accuracy in detecting exclusively attacks instances, and thus reveals the performance in offensive incident detection. Overall, the adaptive approach scores an average ADR of 60.34% and outweighs the static one by 23.8%, as the latter scores an average ADR of 36.54%. The standard deviations are 28.34% and 19.69% for the static and the adaptive approach respectively. In total, the adaptive approach is proved better for the 86% of the states and, notably, the maximum ADR increment is 73.37% (state #8), while the maximum deficient percentage is -5.67% (state #36). As in the case of the accuracy metric, ADR achieves high scores for those states where the static approach witnesses significant performance drops. For those 38 critical states, where the ADR difference between the two approaches is greater than the standard deviation (19.69%), the average increment in the adaptive approach is 60.83%. Additionally, 48 states present an ADR difference smaller than the standard deviation, but again the adaptive approach performs better by 1.84% on average. Finally, only in 14 states the static approach performed better by 2.14% on average. Since ADR is considered a cardinal metric to measure the performance of an IDS, our results suggest that our adaptive method can significantly contribute in sustaining the detection ability of an IDS to high levels. The value of our novel proposal lies in the fact that it can breathe new life into the IDS in critical/sudden situations and increase ADR by up to 73.37%. In principle, in critical situations where the IDS performance drops significantly there is an urgent need for human intervention. Namely, in these cases, ADR can drop to such deficient levels that most of the attacks occurring in the network can go completely unnoticed. Hence, instead of triggering a process of manually retraining the IDS, our solution provides a self-adaptive and autonomous way to keep the IDS's operational ability to high levels.

Furthermore, FIGURE 7 gives an overview of the average accuracy (recall) per class included in the dataset. For each class, we offer a side-by-side comparison of the performance of the static versus the adaptive method. In the figure, the accuracy for the Normal class is almost identical for the two methods. In fact, there is a tiny difference of 0.2% in favor of the static approach. Given that this accuracy is the average of the recall over 100 states, this difference is characterized as minor. On the bright side, the difference concerning the accuracy of the DoS attack class is significant. More specifically, the adaptive method achieves an average accuracy of 39.67% in contrast to the static one that achieves 13.94%. This difference of 25.73% on an average metric is noteworthy. The maximum difference of DoS accuracy recorded among all the 100 states is 80.34% and is perceived in state #8. Actually, in that state, the static IDS witnessed a critical

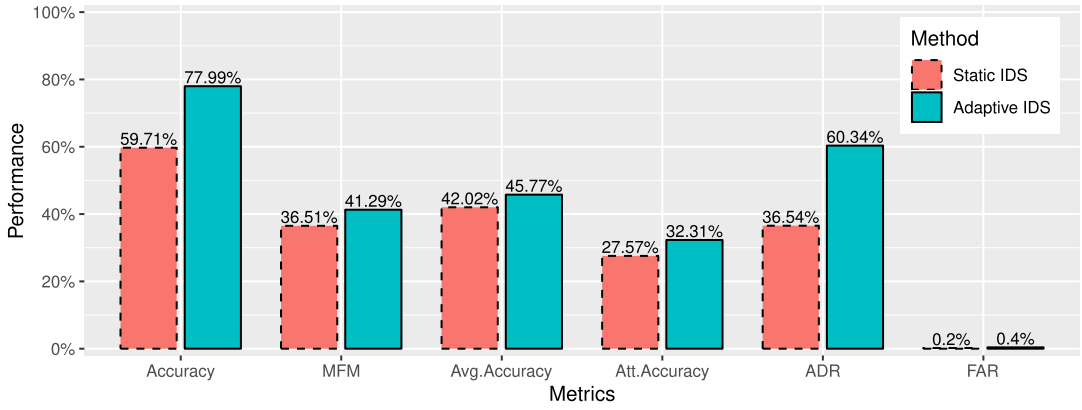

FIGURE 6. Performance comparison over all average metrics.

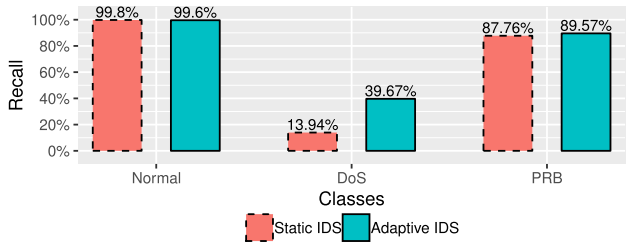

FIGURE 7. Accuracy (Recall) of all classes over the 100 states.

situation as the accuracy in DoS detection was only 0.52% due to the high deviation induced by the new environment. On the other hand, the adaptive approach was able to acquire the necessary knowledge for the unknown network traffic and boost the accuracy metric to 80.87%. Additionally, regarding PRB attack detection accuracy, once again the adaptive approach achieves an average score of 89.57% against the static one, which scored 87.76%. Regarding the attack classes of U2R and R2L, both methods were incapable of detecting any attack vector of those classes. The reason behind this fact is that the aforementioned classes have a small number of instances. Recall from subsection IV-C that any new environmental state stems from a sampling technique. Apparently, the small numbers of instances of those normal/attack classes and the sampling technique to simulate the new environment do not provide adequate instances to build a solid ground truth for the classifier. Overall, the adaptive approach is capable of keeping a stable accuracy on identifying the normal traffic, while its attack detection performance is originated primarily from the DoS attack accuracy and secondarily from the PRB attacks detection accuracy.

The overall performance of the adaptive and the static methods is illustrated in FIGURE 6. The dominance of the adaptive method is verified by all the metrics. Apart from the accuracy and the ADR metrics analyzed above in detail, also the rest of metrics prove the superiority of our methodology. The difference of 4.78% in the MFM metric reveals that the adaptive approach is able to keep the balance between Recall and Precision among all the dataset classes to a greater extent. Note that the MFM metric, as defined in subsection IV-A, is the unweighted average of recall and precision. That is, the unweighted MFM constitutes a more strict metric to evaluate our method, as it treats all classes equally independently of the classes' size. This means that the adaptive method is not only able to provide better attack detection rates, but it is also capable of identifying with higher precision the correct class where the attack instances belong to. Finally, the small deficiency (0.2%) reported in the accuracy metric of the Normal class for the adaptive approach is reflected in the slightly increased FAR of 0.4% in contrast to the 0.2% achieved by the static approach.

Additionally, in order to better understand the distributional characteristics of our results over the chosen metrics for the two methods, FIGURE 8 provides a side-by-side comparison of the boxplots of the metrics. Note that the FAR metric is absent since its deviation is minor. In fact, FIGURE 8 puts in a nutshell the behavior of the two methods as it was already portrayed in FIGURE 4 and FIGURE 5. The dominance of the adaptive approach becomes clear as in all boxes the medians are comparatively higher than those in the static approach. Especially for the boxes representing the Accuracy and ADR metrics, we can notice a significant difference. Regarding the static method, the long size of the second quartile, both for the Accuracy and the ADR, reveals the inefficiency of this method to sustain an acceptable detection level for the IDS in critical situations. This is not the case for the adaptive method, as the concise inter-quartile reveals an overall high stability in both metrics. Additionally, it can be observed that the box of the Accuracy metric of the adaptive approach is slightly higher for the observations above the median (third quartile). Regarding the box of the ADR metric, the adaptive approach achieves significantly higher scores as it is noteworthy that its third quartile starts at that point where the third quartile of the static approach ends. This proves the higher robustness of the adaptive method in detecting offensive incidents in previously unseen environments. Regarding the Attack Accuracy, Average accuracy, and the MFM metrics, once again the boxplots reveal the benefits of the adaptive approach. Note that the aforementioned metrics

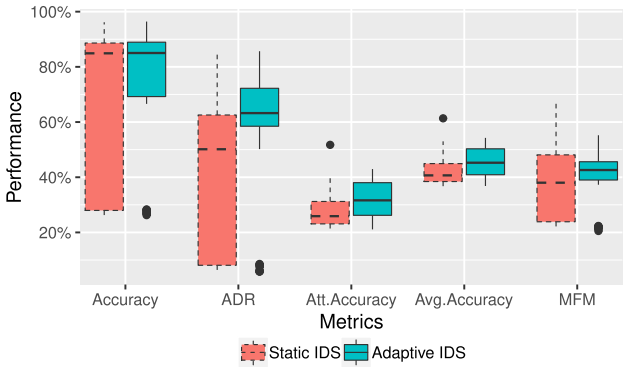

FIGURE 8. Side-by-side Boxplots of metrics for both methods.

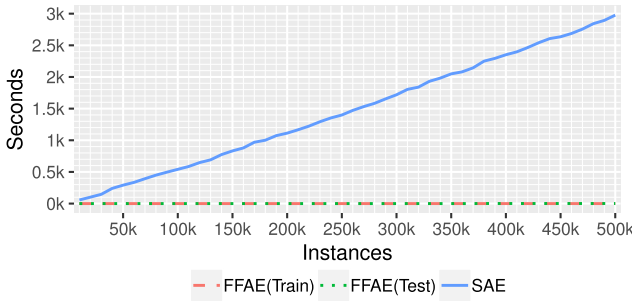

FIGURE 9. Time performance analysis for autoencoders.

are average metrics, and thus it is normal to present a lower deviation in contrast to the Accuracy and the ADR metrics. It is noteworthy that the concise size of the MFM boxplot demonstrates the ability of the adaptive approach to keep the balance between the recall and precision over all the classes and across all the environmental states.

E. TIME PERFORMANCE ANALYSIS

Since our proposal aims to provide an autonomous method for IDS self-adaptation it is critical to analyse the time performance and scalability of its core components. More specifically, the Sparse autoencoder (SAE) and the FeedForward autoencoder (FFAE) of the planning activity are those components that enable the IDS to adapt to a new environmental state. Hence, we measured the time performance of the autoencoders using an incrementing size of dataset instances to evaluate also their scalability. As can can be seen in FIGURE 9, we fed the autoencoders with dataset pieces ranging from 10k to 500k instances having a pivot step of 10k instances. FIGURE 9 reveals that the SAE behaves linearly, while the FFAE (both in the training and testing phase) needs less that 1 sec to obtain the transformation of the whole dataset. The SAE training phase needed 2.977 secs, i.e., ∼49.5 min. given a dataset of 500k instances. The linear performance behavior of the SAE advocates the ability of our approach to scale in big data network environments. From our analysis, one could say that the adaptation cycle can occur virtually indefinitely, while the extra transformation layer added by the FFAE is negligible. Our python implementation was executed on a server empowered with an Intel Xeon E5-2630 v4 @ 2.20GHz CPU. For measuring the time performance, we exclusively utilized only one thread of the CPU. Naturally, the training time could be significantly reduced with the use of GPU accelerators.

V. DISCUSSION

The novel methodology described and evaluated in the previous sections combines the benefits of STL [8] and MAPE-K [9] to deliver a holistic deep learning-based methodology toward self-adaptive and autonomous misuse IDSs. Our solution addresses the challenges of this particular field to a great extent and, to the best of our knowledge, it is the first one to evaluate an IDS under consecutive environmental changes. That is, we prove its ability not only to adapt to new and unknown environments, but to achieve significantly higher scores contrary to a static approach. In fact, as detailed in section IV-C, the compiled dataset which consists of 39 attack classes along with the strategy of simulating new environmental states out of it, reflects with high consistency a realistic situation. The robustness and agility of the proposed methodology is advocated by its superiority over a wide range of legacy classification metrics and over 100 different environmental states. It is worth mentioning that across the vast majority of states our novel approach was able to deliver a significantly higher detection ratio surpassing the static IDS by up to 73.37%. In few states we achieved better but comparable results with the static approach, while only in a handful of states the static approach proved better for a small percentage (2.14% on average). In a nutshell, the acquired results demonstrate that the proposed methodology can revitalize a misuse IDS and boost its ADR by up to 73.37% in critical situations.

Additionally, our simulation posit practical challenges to our methodology. In realistic occasions, an IDS will have to deal with unknown attacks of any class, where their features might be drawn of different probability distributions. Our sampling approach tries to imitate such a challenge and stresses our method over consecutive environment changes. However, this challenge is compensated inherently by the STL properties. Moreover, our evaluation is given in the context of a multi-classification instead of a binary one (Attack/Normal). This provides deeper insights about the performance of the presented approach. Namely, it is important for an IDS solution, not only to be able to detect an attack, but also to designate the class where the attack belongs to in order to aid well-defined counteraction plans [1].

Our proposal comes to deal with a well-known disadvantage of misuse IDSs, namely their stiffness to adapt upon changes. Note that we do not claim that our proposal is able to identify new attack classes, but it is indeed able to grasp an attack's nature based on generalized features reconstructions stemming directly from the unknown environment and its unlabeled data. Crucially, this reconstruction is a product of a scalable method which is able to handle big network data.

We prove that given an initial labeled training set that serves as a basis, the proposed solution is able to revitalize the efficiency of the IDS without the constant need to refresh it, and then retrain the IDS. The transformation of the initial training set – based on the knowledge acquired directly from the current state of the network – provides a high lever of automation in the retraining process. That is, the presented methodology addresses the inherent limitation of misuse IDSs to adapt to new environments, while it significantly alleviates the burden of administrators of constantly refreshing its training set.

VI. RELATED WORK

Machine learning IDSs is a research topic that attracted the attention of the research community for decades. In the recent years, several neural networks-based solutions arose and offered promising solutions. This section offers an overview of deep and shallow learning methods, while it also elaborates on whether they bear any adaptable features. This section also refers to methodologies that aim to provide a level of automation in the IDS adaptation and discusses our contributions over them.

A. NEURAL NETWORKS-BASED APPROACHES

The work in [20] introduces a deep learning solution for NIDSs. The authors utilize the self-taught learning methodology exclusively as an unsupervised feature learning method for supporting a statically trained IDS. Instead, our solution provides a more powerful setup of the STL in conjunction with MAPE-K methodology to deliver a deep learning methodology for adaptive IDSs. In fact, contrary to [20], we pass the IDS through environmental changes to prove that our approach is able to generate knowledge out of unknown environments.

Ma *et al.* [21] propose an IDS algorithm based on Spectral Clustering (SC) [22] and Deep Neural Networks (DNN) [23]. Through SC the proposed method is able to identify cluster centers that divide a raw dataset into data clusters with similar features. Those data clusters are fed as training data into DNN's of multiple layers. The algorithm trains as many DNN's as the clusters identified by the SC and aggregates the final result in an ensembled way. However, the proposed deep learning approach does not provide any kind of adaptiveness to the system.

In the context of MAPE-K control loop, Lee and Park [4] proposed an adaptive IDS in terms of network environmental changes. By exploiting the MAPE-K model, the authors were able to perceive the environment changes and plan appropriate update actions in the Snort IDS detection rules. Lee and Park [4] utilize MAPE-K model for regulating the adaptive process. However, our approach goes beyond that point and exploits MAPE-K model to build a deep learning misuse IDSs in the adaptive frame.

Fernández Maimó et al. [24] use deep learning techniques to detect network anomalies in 5G networks. The authors utilize DNN and Long Short-Term Memory (LSTM) Recurrent Networks to empower the anomaly detection. Their proposal takes self-adaptation actions with respect to the network load requirements by applying management policies. Nevertheless, the adaptation policies are applied to the network load management rather the detection framework per se.

The work in [25] proposes a deep learning method for detecting DoS attacks based on Restricted Boltzmann Machine (RBM). Specifically, the authors used a Gaussian-Bernoulli RBM with 7 hidden layers with 100 neurons each. Through the aforementioned setup the proposed method learns a reduced set of new features of the NSL-KDDTrain+_20Percent dataset [17] (train: 25,194 instances, test: 4,508 instances). According to the authors, the Gaussian–Bernoulli RBM is able to outperform the deep learning approach of Bernoulli-Bernoulli RBM and Deep Belief Network, and the legacy machine learning methods of SVM (radial basis), SVM (epsilon-SVR), and decision tree. Even though the proposed method learns new features representation out of unlabeled data, it is not exposed to unknown data for supporting an adaptable approach.

Yu *et al.* [26] designed an IDS model by stacking dilated convolutional autoencoders (DCAEs) for learning features representations from unlabeled data. In their experiments, the authors tested the generalization ability of their detection model by testing it with previously unknown attacks. Even though the authors pose their trained model against new attacks, they do not proceed to any automated retraining method. Rather they aim to exclusively test the generalization ability of the learned features employed to statically train the IDS. Additionally, the authors approach the problem as a binary one (normal/attack), while in our classification case we deliver a multi-classification method.

Shone *et al.* [27] propose a new type of autoencoder namely non-symmetric deep autoencoder (NDAE) and they utilized it in a classification model using stacked NDAEs. According to the authors, the NDAE engaged only an encoding phase for reducing the complexity of the network with minor effect on the accuracy of the model. At the end of the stacked NDAE, the authors attach Random Forest algorithm that undertakes the classification task based on the features learned from the NDAEs. The proposed setup achieves high accuracy rates, but the author's methodology does not bear adaptability characteristics.

B. ADAPTIVE METHODOLOGIES IN IDS REALM

Although the IDS research on IDS has offered an large amount of works [18], [19], [28], [29] the vast majority of them focus on providing highly accurate end-models with minor false alarm rates. Thus, the adaptability property remains an open issue and it is a well-known drawback, especially for the misuse IDS domain. Still, there are methodologies that, according to the literature [3], could provide the necessary foundations to adaptive IDSs. Among them, Learning Classifier Systems (LCS) [5], Artificial Immune Systems (AIS) [6], Swarm intelligence [7], Evolutionary computing [30] and Reinforcement learning [31]. There

is no doubt that all of these approaches can offer their principles to build adaptive systems for practical problems. However, the intrusion detection problem has some inherent characteristics that one has to take into consideration when aiming to build a self-adaptive and autonomous IDS. That is, a self-adaptive and autonomous approach implies the complete disengagement of the human factor or at least a minimal interaction in the form of a supervision.

The self-adaptation property implies that an IDS should be able to adapt itself to the needs of a new environment even without the need of feedback from the administrators. This means that the adaptation process of an IDS cannot be based on labeled data because labeling network data of a new environment is a demanding engineering task. That is, approaches that base their adaptation on labeled data can be considered adaptive, but not self-adaptive. As a result, to support self-adaptation in the context of intrusion detection, we need to invest on methods that can exploit unlabeled data to improve the detection performance.

Additionally, in an on-line machine learning problem, where new instances need to be classified instantly and accurately, there is a need of approaches that can adapt to new environments also in a realtime fashion. This entails that solutions that rely their adaptation on a trial-and-error approach as those based on reinforcement learning, seem to be impractical for this nature of problems. In fact, an IDS cannot learn in the same way that, say, a robot does. For example, if a robot encounters an obstacle then it learns out of this incident and proceeds to an adaptation of its objectives. Unfortunately, in the intrusion detection context, the only entity that can identify a mis-classification (i.e., an obstacle) is the administrator who notices an ongoing attack. An IDS is not in position to know if a new instance is misclassified or not. In other words, the IDS cannot see the ''obstacles'', but the ''obstacles'' are in practice attacks that go unreported or normal instances which are detected as attacks and increase the false alarm rate. In this sense, we need to invest on methods that can learn from the unknown environment in an autonomous way.

To this end, our proposal tries to address the aforementioned challenges. The combination of Self-taught learning [8] and MAPE-K [9] brings together the benefits of transfer learning from unlabeled data and places this ability in the frame of autonomic computing. That is, our self-adaptive and autonomous method enables an IDS to extract new features out of the unlabeled and unknown traffic of a new environment and exploit them for retraining in an autonomous way its detection engine. Furthermore, our solution enables the IDS to adapt according to the dynamics of the new environment even if this is overwhelmed by previously unseen traffic. Given an initial training set, a new features construction is learned using a neural networks-based sparse autoencoder, and via a feed-forward autoencoder the initial training set is updated to meet the challenges of the new environment.

VII. CONCLUSIONS AND FUTURE WORK

This paper introduces a novel methodology that advances the state-of-the-art in the literature of misuse IDS. The highlight of our scheme is that it can practically render any misuse IDS autonomous, i.e., self-adaptive to the ever-lasting changes made in its network environment. This means that in such frequently occurring (and sometimes sudden) transition periods, the IDS is able to maintain an at least acceptable attack detection rate, which otherwise is fated to drop abruptly, rendering the IDS useless. This quality is also a great relief to the security administrators who after a network environment change are granted enough time to possibly update the IDS's detection model. The proposed methodology uniquely blends the MAPE-K reference model and a deep-learning technique called self-taught learning to enable an IDS to identify previously unseen attacks via reconstructions made on unlabeled data. The linear performance behavior for acquiring the aforementioned reconstructions, renders our proposal suitable for contemporary big data network environments. The effectiveness of our proposal is demonstrated through extensive experimentation considering several metrics and a plethora of attacks included in widely used datasets. As future work, we aim to test our methodology with more contemporary datasets, while we aim to investigate ways to improve detection rates for small dataset classes. We additionally aim to experiment with other methodologies like sparse RBMs, K-means clustering and Gaussian mixture models (GMMs) to investigate further improvement of our method.

REFERENCES

[1] P. Nespoli, D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, ''Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks,'' *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1361–1396, 2nd Quart., 2018.

[2] D. Diaz-López, G. Dólera-Tormo, F. Gómez-Mármol, and G. Martínez-Pérez, ''Dynamic counter-measures for risk-based access control systems: An evolutive approach,'' *Future Gener. Comput. Syst.*, vol. 55, pp. 321–335, Feb. 2016.

[3] H. Bensefia and N. Ghoualmi-Zine, ''Adaptive intrusion detection systems: The next generation of IDSs,'' in *Network Security Technologies: Design and Applications*. Hershey, PA, USA: IGI Global, 2013, p. 239. [Online]. Available: https://www.igi-global.com/chapter/adaptive-intrusion-detection-systems/105811, doi: 10.4018/978-1-4666-4789-3.ch014.

[4] K.-H. Lee and Y. B. Park, ''A study of environment-adaptive intrusion detection system,'' in *Advances in Computer Science and Ubiquitous Computing*, J. J. Park, Y. Pan, G. Yi, and V. Loia, Eds. Singapore: Springer, 2017, pp. 625–630.

[5] R. J. Urbanowicz and J. H. Moore, ''Learning classifier systems: A complete introduction, review, and roadmap,'' *J. Artif. Evol. Appl.*, vol. 2009, pp. 1:1–1:25, Jan. 2009.

[6] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross, ''Immune system approaches to intrusion detection—A review,'' *Natural Comput.*, vol. 6, no. 4, pp. 413–466, Dec. 2007.

[7] C. Kolias, G. Kambourakis, and M. Maragoudakis, ''Swarm intelligence in intrusion detection: A survey,'' *Comput. Secur.*, vol. 30, no. 8, pp. 625–642, 2011.

[8] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, ''Self-taught learning: Transfer learning from unlabeled data,'' in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 759–766.

[9] J. O. Kephart and D. M. Chess, ''The vision of autonomic computing,'' *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[10] Y. Brun *et al.*, ''Engineering self-adaptive systems through feedback loops,'' in *Software Engineering for Self-Adaptive Systems*. Berlin, Germany: Springer, 2009, pp. 48–70.

[11] R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, "Software engineering for self-adaptive systems: A second research roadmap," in *Software Engineering for Self-Adaptive Systems II*. Berlin, Germany: Springer, 2013, pp. 1–32.
[12] N. Brownlee, C. Mills, and G. Ruth, *Traffic Flow Measurement: Architecture*, document RFC 2722, 1999.
[13] Argus. *The Network Audit Record Generation and Utilization System*. Accessed: Jan. 23, 2019. [Online]. Available: https://qosient.com/argus/index.shtml
[14] CICFlowMeter. *UNB CIC Network Traffic Flow Generator (Formerly SCXflowmeter)*. Accessed: Jan. 23, 2019. [Online]. Available: http://www.unb.ca/cic/datasets/flowmeter.html
[15] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Proc. 19th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2006, pp. 801–808. [Online]. Available: http://dl.acm.org/citation.cfm?id=2976456.2976557
[16] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 227–261, Nov. 2000.
[17] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. 2nd IEEE Int. Conf. Comput. Intell. Secur. Defense Appl. (CISDA)*, Jul. 2009, pp. 53–58.
[18] S. Elhag, A. Fernândez, A. Bawakid, S. Alshomrani, and F. Herrera, "On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems," *Expert Syst. Appl.*, vol. 42, no. 1, pp. 193–202, 2015.
[19] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Dendron: Genetic trees driven rule induction for network intrusion detection systems," *Future Gener. Comput. Syst.*, vol. 79, pp. 558–574, Feb. 2018.
[20] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (Formerly BIONETICS)*, 2016, pp. 21–26.
[21] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, 2016.
[22] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007.
[23] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[24] L. F. Maimó, A. H. Celdrán, M. G. Pérez, F. J. G. Clemente, and G. Martínez-Pérez, "Dynamic management of a deep learning-based anomaly detection system for 5G networks," *J. Ambient Intell. Hum. Comput.*, pp. 1–15, May 2018. [Online]. Available: https://link.springer.com/article/10.1007/s12652-018-0813-4
[25] Y. Imamverdiyev and F. Abdullayeva, "Deep learning method for denial of service attack detection based on restricted boltzmann machine," *Big Data*, vol. 6, no. 2, pp. 159–169, 2018.
[26] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Secur. Commun. Netw.*, vol. 2017, Nov. 2017, Art. no. 4184196.
[27] 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.
[28] C. Kolias, V. Kolias, and G. Kambourakis, "TermID: A distributed swarm intelligence-based approach for wireless intrusion detection," *Int. J. Inf. Secur.*, vol. 16, no. 4, pp. 401–416, Aug. 2017.
[29] D. Papamartzivanos and F. G. Mármol, "Intrusion detection and prevention system and method for generating detection rules and taking countermeasures," U.S. Patent 10 084 822, Sep. 25, 2018.
[30] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.
[31] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

DIMITRIOS PAPAMARTZIVANOS received the M.Sc. degree in information and communications systems security of the Department of Information and Communication Systems Engineering, University of the Aegean, Greece. He is currently pursuing the Ph.D. degree with the Info-Sec-Lab, Department of Information and Communication Systems Engineering, University of the Aegean, Greece. His research concentrates on the field of intrusion detection systems, assisted by machine learning technologies, dealing both with misuse and anomaly detection systems. His research interests include VoIP and the IoT security, and adaptive intrusion detection and response systems. More info at: http://www.icsd.aegean.gr/dpapamartz.

FÉLIX GÓMEZ MÁRMOL received the M.Sc. and Ph.D. degrees in computer engineering from the University of Murcia. He is currently a Researcher with the Department of Information and Communications Engineering, University of Murcia, Spain. His research interests include cybersecurity, the Internet of Things, machine learning, and bio-inspired algorithms. More info at: https://webs.um.es/felixgm.

GEORGIOS KAMBOURAKIS received the Ph.D. degree in information and communication systems engineering from the Department of Information and Communications Systems Engineering, University of the Aegean, Greece, where he is currently an Associate Professor, and also the Director of the Info-Sec-Lab. His research interests are in the fields of mobile and wireless networks security and privacy. He has over 120 refereed publications in the above areas. More info at: http://www.icsd.aegean.gr/gkamb.