Reinforcement Learning for Intrusion Detection: More Model Longness and Fewer Updates

Roger R. dos Santos, Eduardo K. Viegas, *Member, IEEE*, Altair O. Santin, *Member, IEEE*, and Vinicius V. Cogo

***Abstract*—Several works have used machine learning techniques for network-based intrusion detection over the past few years. While proposed schemes have been able to provide high detection accuracies, they do not adequately handle the changes in network traffic behavior as time passes. Researchers often assume that model updates can be performed periodically as needed, although this is not easily feasible in real-world scenarios. This paper proposes a new intrusion detection model based on a reinforcement learning approach that aims to support extended periods without model updates. The proposal is divided into two strategies. First, it applies machine learning scheme as a reinforcement learning task to long-term learning - maintaining high reliability and high classification accuracies over time. Second, model updates are performed using a transfer learning technique coped with a sliding window mechanism that significantly decreases the need for computational resources and human intervention. Experiments performed using a new dataset spanning 8TB of data and four years of real network traffic indicate that current approaches in the literature cannot handle the evolving behavior of network traffic. Nevertheless, the proposed technique without periodic model updates achieves similar accuracy rates to traditional detection schemes implemented with semestral updates. In the case of performing periodic updates on our proposed model, it decreases the false positives up to 8%, false negatives up to 34%, with an accuracy variation up to only 6%, while demanding only seven days of training data and almost five times fewer computational resources when compared to traditional approaches.**

***Index Terms*—Intrusion detection, reinforcement learning, network traffic, machine learning.**

I. Introduction

THE NUMBER of cyberattacks has increased significantly, currently accounting for almost a fifth of worldwide network traffic [1]. According to a recent security report, the number of network-based attacks in the first quarter of 2022 increased 4.5 times compared to the same period in the previous year [2]. Network administrators must access security solutions that can reliably detect this growing number of network attacks. Intrusion detection systems (IDS) are widely deployed to monitor and identify network attacks, classify malicious activities, and neutralize them in a given environment [3]. Solutions from the literature often rely upon two main approaches to accomplish this intrusion detection task [4]. On one hand, *misuse-based* approaches aggregate well-known attack patterns and signatures for identifying them in the passing network traffic. However, they only detect previously known signatures [5], leaving systems unprotected against zero-day attacks, for instance. On the other hand, *Behavior-based* approaches analyze the players' conduct within a given network environment to signal misconducts, for instance, by applying a machine learning (ML) model. Solutions adopting this approach usually can detect new intrusions, but only if they behave likewise those previously known attacks used to build the behavioral model [6], [7].

The ever-increasing number of newly identified attacks has motivated the proposal of several works (e.g., [3], [8], [9]) for *behavior-based* intrusion detection, usually making use of ML through pattern recognition techniques. These solutions usually build their ML model based on massive datasets containing billions of events that must dependably represent the expected behavior of the respective production environments [6]. Although the extracted ML model can signal events associated with previously unknown attacks, using these massive datasets implies in computationally-expensive training phases and several challenges related to the labeling of such events [10].

The behavior of network environments can vary considerably over time due to the exploitation of new attacks, and the emergence of new services, among other reasons [6], [7]. This non-stationary nature of network production environments quickly renders the ML model outdated, requiring systems administrators to perform frequent and expensive model updates [6]. The rationale for this requirement is that an outdated ML model cannot maintain the accuracy rates obtained during the testing phase, becoming unreliable and putting production environments at risk [7]. In practice, the IDS alerts will escalate and become primarily false positives, motivating administrators to disregard them while an updated ML model is not yet available. In the literature, authors often assume periodic model updates will be performed, but they

Manuscript received 10 February 2022; revised 10 June 2022 and 30 August 2022; accepted 5 September 2022. Date of publication 15 September 2022; date of current version 6 July 2023. This work was partially sponsored by the Brazilian National Council for Scientific and Technological Development (CNPq), grant n° 304990/2021-3, and the FCT through the LASIGE Research Unit, ref. UIDB/00408/2020 and ref. UIDP/00408/2020. The associate editor coordinating the review of this article and approving it for publication was N. Z.-Heywood. *(Corresponding author: Altair O. Santin.)*

Roger R. dos Santos and Altair O. Santin are with the Graduate Program in Computer Science, Pontificia Universidade Catolica do Parana, Curitiba 80215-901, Brazil (e-mail: roger.robson@ppgia.pucpr.br; santin@ppgia.pucpr.br).

Eduardo K. Viegas is with the Graduate Program in Computer Science, Pontificia Universidade Catolica do Parana, Curitiba 80215-901, Brazil, and also with the Secure Systems Research Center, Technology Innovation Institute, Abu Dhabi, UAE (e-mail: eduardo@ssrc.tii.ae).

Vinicius V. Cogo is with the LASIGE, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, 1649-004 Lisbon, Portugal (e-mail: vvcogo@fc.ul.pt).

Digital Object Identifier 10.1109/TNSM.2022.3207094

either consider it an orthogonal problem or overlook the challenges posed by model retraining task [11]. Model updates for pattern recognition demand the collection of up-to-date events, expert assistance for event labeling, and the mentioned computationally-expensive model retraining [3].

All procedures mentioned above can take weeks or even months of expert assistance—which may be unavailable in some organizations or entail high costs to them [12]. Therefore, providing an ML model that can reliably withstand long periods is a must, as model updates are neither easy nor cheap.

Traditional pattern recognition and classification approaches usually are not designed to last long-term [7]. Previous work in general seeks higher classification accuracies, with significant tradeoffs on the model's generalizability, resulting in severe decreases in its detection performance [6]. Unfortunately, ML intrusion detection techniques in the literature still neglect to evaluate the reliability of their ML model as time goes [13].

An outdated ML model must be updated as soon as possible. However, identifying expired models is a challenging task [11] as the network administrator must manually evaluate whether the current model's accuracy still meets the accuracy measured at the test phase. In general, proposed approaches for such tasks rely upon supervised settings (e.g., drift detection mechanisms) that assume the proper event label is always available [14]. The problem is that, contrary to the test phase, the label of events is not previously known in production. Current approaches either neglect when the current ML model will become outdated, or the impact of their model's lifespan, presuming that periodic model updates are performed without considering their costs [6].

Ideally, model updates should demand the lowest computational resources as well as the minimum amount of data due to the high costs associated with the event labeling task [6]. In contrast, traditional pattern recognition solutions discard their active model and build new ones based on the newly obtained training data at each model update procedure. The execution of such a task demands significant amounts of data to reliably extract the new ML model while also consuming excessive processing resources. The training data size must be as big as the original (despite prior knowledge from the outdated discarded model) [3]. As a result, ML-based intrusion detection schemes remains mostly as a research topic, as they are rarely used in production due to network traffic's evolving behavior and the challenges it incurs to deployed ML-based techniques.

This paper proposes a new intrusion detection model based on reinforcement learning (RL) that aims to extend the model's longness to increase its lifespan, reduce the accuracy variation as time passes, and facilitate model updates. The first goal is achieved by exploring the insight that a long-lasting model can be reached if the model training pursues high accuracy and correctness. Our proposal measures the model correctness according to the classification confidence distance to the proper event label. The main assumption is that the classifier classification confidence can be used to attest to the correctness of the performed classification. We thus build our first objective, with the obtained model aiming for higher classification correctness across all classified events rather than improving it only on a subset of those in the training phase. Our second goal is reached because the model will be more accurate and also more reliable, decreasing its accuracy variation over time. The third goal is achieved by performing model updates considering a sliding time window to decrease the number of labeled data that must be provided when the active model becomes outdated. Notwithstanding our proposed model update procedure leverages prior knowledge from the outdated model making use of its parameters (weights), i.e., instead of discarding it entirely, we follow a transfer learning-based approach that decreases the computational resources needed for such a task. The proposed scheme can significantly extend the model longness and reduce the computational resources and the human expert intervention needed to perform model updates when the active model becomes less effective.

More specifically, the main contributions of this paper are threefold:

- A new publicly available intrusion *dataset* composed of four years of real, valid, and labeled network traffic provided on a daily basis. The dataset is the first of its kind and encompasses more than 8TB of data, composed of more than seven billion of network flows. The dataset enables researchers to evaluate proposed ML-based intrusion detection schemes on how changes in the network traffic behavior over time affect the classification accuracy and how model updates can be conducted to address such a challenge;
- An evaluation of the classification accuracy, variation, and longness of widely-used ML-based techniques for intrusion detection. Our experiments indicate that current ML-based approaches in the literature cannot cope with changes in the network traffic behavior over time, increasing their error rate by up to 38% only a year after the training period while significantly varying their accuracy as time passes. Notwithstanding, we show that periodic model updates can be used to provide reliability in intrusion detection. However, it must be conducted based on the newly occurring network traffic characteristics, demanding a higher frequency of model updates as soon as a higher attack frequency is evidenced. The experiments show that current ML-based approaches demand frequent and computational expensive model updates that make them hardly feasible for production deployments;
- A new reinforcement learning model for intrusion detection that significantly improves classification reliability over time. The proposed scheme aims to build and update the ML model through a fine-tuning procedure that aims for higher model confidence values rather than only improving their obtained accuracies as time passes. As a result, the proposed model can improve the model lifespan by up to two years when no model updates are performed. If done so, it can leverage the outdated ML model, demanding only 21% of computational costs while improving the false-negative rates by up to 34% and decreasing the accuracy variation to only 6%.

The remainder of this paper is organized as follows. Section II contextualize machine and reinforcement learning techniques for intrusion detection. Section III reviews the

TABLE I
FEATURES SET EXTRACTED AT THE NETWORK LEVEL FOR EACH FEATURE GROUPING IN A TIME WINDOW INTERVAL OF 15S

Grouping Features	Collected Features
Src. IP Addresses, Src. IP and Dst. IP Addresses, Src. and Dst. Service Ports	Number of Packets
	Number of Bytes
	Average Packet Size
	Percentage of Packets (PSH Flag)
	Percentage of Packets (SYN and FIN Flags)
	Percentage of Packets (FIN Flag)
	Percentage of Packets (SYN Flag)
	Percentage of Packets (ACK Flag)
	Percentage of Packets (RST Flag)
	Percentage of Packets (ICMP Redirect Flag)
	Percentage of Packets (ICMP Time Exceeded Flag)
	Percentage of Packets (ICMP Unreachable Flag)
	Percentage of Packets (ICMP Other Types Flag)

related works. Section IV introduces our novel dataset and presents an evaluation of how widely-used ML algorithms for intrusion detection perform on it over time. Section V describes our proposed reinforcement learning model for intrusion detection, while Section VI evaluates it. Section VII presents the final remarks to conclude this paper.

II. Background and Context

A. Machine Learning for Intrusion Detection

Generally, the approaches from prior work perform intrusion detection through four sequential modules, namely *Data Acquisition*, *Feature Extraction*, *Classification*, and *Alert*. The *Data Acquisition* module is responsible for monitoring the environment to collect events (e.g., the network packets from a network interface card). The collected data are forwarded to the *Feature Extraction* module, which extracts a set of event behavioral features. Generally, the network's data behavior is analyzed based on its flow – communication history.

Table I shows an example of network flows usage for the feature values computation in intrusion detection [10]. In this case, a set of thirteen features is extracted according to the communication history (e.g., a fifteen-second interval) between hosts, host *vs.* host, and service *vs.* service. The resulting feature vector is the collected features organized through the three groups of features (left column of the table), yielding a total of thirty-nine features for each network flow. The extracted set of features composes a feature vector used as input to the *Classification* module, which classifies it as *normal* or *attack*. Several approaches can be used to perform the classification task, in which ML through pattern recognition techniques is typically used [3]. Finally, if a given event is classified as an *attack*, the *Alert* module properly reports it to the network administrator.

Machine learning techniques have been successfully applied in several domains, including fraud detection [15], medical diagnosis [16], and optical character recognition [17]. However, despite the promising results reported by prior works, ML-based intrusion detection techniques are rarely deployed in production environments [6]. Networked environments pose more hurdles when compared to domains where ML has succeeded. The behavior in network environments can vary considerably over time due to the exploitation of new attacks, the emergence of new services or even changes in the communication link [6], [7], [12].

Building a reliable ML model in network-based intrusion detection is challenging and often overlooked task. Training datasets must encompass events for as many expected behaviors of production environments as possible, which is not always feasible due to networked environments' highly variable nature. The ML-based classification model must properly generalize the behavior experienced in the training phase [3], which is often achieved only through losses in classification accuracy.

The literature often neglects or omits the performance of their solutions in production environments, highlighting only high accuracies obtained in the test phase – at the cost of decreasing the model generalization. Additionally, even if a ML model with high generalization capabilities is built, it will become unreliable as time goes on [6] since the training dataset assumes a static environment that is highly variable in reality. Therefore, the ML model must be generalizable and updated as soon as it decreases its expected classification rates. Despite the lack of adoption in production environments, current literature approaches still neglect network traffic's highly variable nature. At the same time, they also assume that periodic model updates can be easily performed, while the challenges involved in such a process remain mostly overlooked [6], [7], [13].

B. Reinforcement Learning

In contrast to traditional ML techniques, reinforcement learning (RL) aims to find an optimal learning policy strategy during the model-building phase [18]. RL-based techniques are performed through an entity called the *agent*. The *agent* performs its action on a given environment through *perception* and *action* mechanisms. The former is used by the *agent* to collect the *environment* current *state*, while the *action* enable the *agent* to act in the given *environment*.

Usually, an RL-based approach is executed iteratively over the given environment at the training phase. At each iteration, the *agent* receives as input the current environment *state* as measured through the feature vector. One can be noticed that the *agent* acts through an *action* over the environment, as defined by its policy (e.g., applying an ML model over the environment *state*), which changes its state and generates a new *reward* as output. Therefore, the training procedure goal is to find an *agent* that performs *actions* that increase its obtained rewards over time.

Traditional ML relies upon an input and output pair, where an ML model receives a given event feature vector as input and outputs its estimated class value. RL-based approaches are significantly different from traditional ML-based ones [18]. RL-based techniques are not given the event class value [3]. Instead, after an *action* is performed, the *agent* only receives the *reward* for its *action* and the subsequent environment *state*. The *agent* learns the optimal decision threshold in the long term, given that it optimizes its rewards over time.

Noticeably, intrusion detection through RL techniques is still being developed [8], but they are very promising. Authors from literature solutions generally pursue higher classification accuracy by modeling the intrusion detection field as an RL-based environment.

III. Related Work

A. Machine Learning for Intrusion Detection

Machine learning reached a strong relation with the intrusion detection domain since both communities benefit from optimizing ML algorithms in many dimensions (e.g., maximizing accuracy, minimizing false alarms) while addressing real-world problems. Despite the immediate primary goal of many proposals being to provide the highest accuracy possible in attack detection (e.g., [19], [20], [21]), other objectives arose since these attacks (and the network traffic) evolved. Some prominent examples of additional objectives addressed by related works include speeding up model updates with feature selection (e.g., [22], [23]), reducing the dependence on humans to label new events (e.g., [24]), avoiding discarding the whole previously employed (even outdated) models (e.g., [25]), among others.

Mauro *et al.* [26] evaluated several deep learning techniques on a variety of publicly available intrusion datasets, showing that the selected approaches' accuracy does not significantly vary, regardless of the selected deep learning architecture. Unfortunately, the authors did not evaluate how the network traffic behavior changes affect the classification accuracy. Intrusion detection techniques aiming at higher detection accuracies are a widely addressed challenge in the literature. Upadhyay *et al.* [27] proposed a feature selection technique to increase the accuracy of ML-based intrusion detection in a single dataset. The author proposed scheme selected the most suited features to proactively increase accuracy. The authors have neglected the proposal's generalization capacity and the impact of network traffic behavior changes over time.

Some related works (e.g., [27]) mention that adopting an ensemble of classifiers may be able to increase the chances of always employing the most accurate of their models for detection. In such a context, Das *et al.* [28] evaluates the classification accuracy of several ensemble ML techniques in publicly available intrusion datasets, showing that such approaches can achieve higher detection accuracies when compared to the single classifiers approach. However, if all adopted models decrease the detection accuracy immediately after the testing phase, this ensemble classifier may not be very helpful. Again, the impact of network traffic changes is not even evaluated, leaving an opportunity to address such a challenge in the literature.

Several works have resorted to deep learning techniques to increase accuracy in their proposed schemes in recent years. Longari *et al.* [29] have proposed an intrusion detection scheme based on long short-term memory (LSTM) and autoencoder to identify intrusion attempts as anomalies. Their proposed scheme was able to increase detection accuracy when compared to traditional ML techniques. However, the authors neglect the challenges related to model updates. Another LSTM-based approach was proposed by Imrana *et al.* [30] as an attempt to decrease the number of false alarms. Their proposed model increased accuracy in a multi-class setting while being evaluated in an outdated dataset. Again, the authors neglected the evolving behavior of network traffic, and traditional ML evaluation was used.

B. Changes in Network Traffic

Although many of these works achieve high detection accuracy at the testing phase, none evaluates the mentioned metric over time. The main challenge is the creation of a realistic intrusion dataset that makes it possible to evaluate the accuracy of the designed techniques as time passes [10]. To achieve such a goal, the network administrator must perform the proper data collection and the labeling of the network traffic in a long interval, which is often unfeasible in production environments [6]. As a consequence, authors generally resort to publicly available datasets, which often do not provide the expected properties from production environments [7].

Network traffic behavior changes over time are hardly considered in the literature. Alshammari and Zincir-Heywood [31], [32] evaluated the performance of ML-based classifiers used in a different environment for the classification of application-level network traffic. The authors found that new network traffic features, such as those caused due to distinct periods, may significantly impact flow-based classification systems. Liang and Ma [33] recently mentioned the detection rates of IDSs gradually decaying with the emergence of new attacks. However, they only address the incentives for collaborative model retraining by assigning more blockchain-based tokens to transactions that classify unknown suspect packets. Their framework must download the latest database and retrain the whole model without using outdated ML model knowledge. Additionally, they did not benchmark the frequency of these updates required to maintain the training detection accuracy in production environments.

The evolving behavior of network traffic was also considered by Hsu and Matsuoka [34]. The authors created a *"simulated"* network traffic behavior change by creating several chunks in intrusion detection datasets and evaluating their technique according to each fragment. The authors noticed the accuracy variation in their evaluation. However, the changes in network traffic behavior were created in a controlled setting, hence, not reproducing the actual behavior of real-world environments.

C. Reinforcement Learning for Intrusion Detection

Recently, a promising approach in intrusion detection has relied on reinforcement learning (RL). Suwannalai and Polprasert [35] proposed a deep RL scheme to increase accuracy in a single dataset. The authors provided higher detection accuracy compared to traditional ML techniques while neglecting the challenges related to the actual deployment of their proposed scheme in production environments, such as the behavior changes in network traffic. Benaddi *et al.* [36] proposed a RL scheme to detect malicious adversaries in wireless sensor networks. Their proposed approach was able to increase accuracy when compared to traditional ML techniques. However, model updates are also overlooked.

Another RL technique was proposed by Servin and Kudenko [37] for distributed learning in intrusion detection. Their proposed model addresses model updates in a distributed manner following a hierarchical-based strategy. Unfortunately, the easing of the model update burden is overlooked as they assume that the label of events can be requested as needed. Lopez-Martin *et al.* [8] proposed a more practical application of deep RL. In their work, the authors propose a new training procedure for RL that replaces the live environment with mini-batches of a training dataset. Thus, it is feasible for model updates that follow such a procedure and leverage the outdated model. Unfortunately, the authors overlook the model updates in their evaluation and how the model update procedure can be eased in the face of new network traffic behavior.

D. Discussion

It can be noted that a plethora of works has been proposed to increase detection accuracy in the field of network-based intrusion detection. However, despite the promising reported results, such as those obtained by traditional machine learning [27], [28], deep learning [29], [30], or even through deep reinforcement learning techniques [8], [35], [36], [37], their actual deployment in the production environment remains low. The challenges of networked production environments remain neglected by the literature, and the traditional ML evaluation procedure occurs.

Proposed schemes deployed in real-world settings will be subject to a highly variable behavior of the underlying network, making it unfeasible to be reproduced in a training dataset – demanding that the proposed scheme generalize the behavior of the network traffic [6], [7]. In practice, the network traffic behavior will change over time [33], [36], demanding model updates to be performed periodically. However, model updates are also a challenging task that often demands huge amounts of labeled training data to be provided.

Such open opportunities motivate evaluating traditional methods with more extended periods and studying their accuracy variation and model longness to draw more general conclusions about this downward trend in accuracy rate for network traffic classification tasks. These open problems/opportunities contribute to the previously addressed goals to provide security administrators with more trustworthy and easy-to-maintain IDS solutions.

IV. Problem Statement

This section investigates the main aspects that make network-based intrusion detection challenging for ML-based techniques. More specifically, we accomplish this task by proposing a new intrusion dataset and evaluating the performance of widely used ML-based intrusion detection approaches.

A. MAWIFlow Dataset

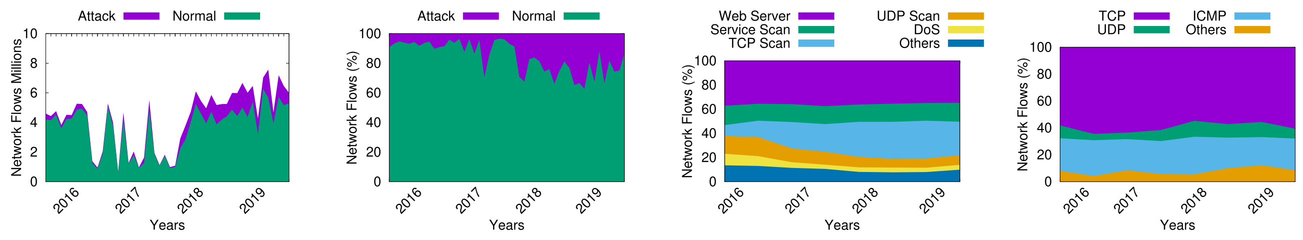
The reliability of network-based intrusion detection mechanisms relies upon using a realistic training dataset. However, current approaches in the literature often build their techniques on top of outdated datasets with several known flaws [38]. Previous work is rarely used in production environments, despite the promising results, such as a high-accuracy ML model [6], [7]. In practice, a realistic training dataset [12] must contain *real* network traffic that can be experienced in production environments with *valid* and *correct* network traffic communication protocols implementation.

The network traffic has to be *highly diverse* in terms of used protocols and network behavior, ensuring the *completeness* availability for the network traffic protocols in the production environment as well as their variability. The network events must be *prior labeled* as belonging to a class (e.g., normal or attack) to enable the proper evaluation of the model and building of the ML classifier. Finally, the dataset must be *publicly available* in a *usable* format to enable the proper intrusion detection proposal benchmarking.

Usually, proposals in the literature collect network traffic from a controlled testbed or monitor the production environment behavior to support the mentioned requirements [10]. The former enables the appropriate data sharing and labeling of events due to its controlled nature. However, they lack a highly diverse network content of the production environment and generate unrealistic network traffic. In contrast, the latter produces real and valid network data that is highly diverse but hinders the proper labeling of events and the public sharing of the collected data (e.g., due to the privacy concerns associated with real network traffic). Regardless of the approach adopted for building the datasets, works from the literature have not adequately considered network traffic's evolving behavior. It means that even the best datasets will fail as the available network traffic remains unmodified in such data over time.

Our new dataset, named *MAWIFlow*, comprises real, valid, and prior labeled network traffic collected from production environments for long periods. More specifically, its network flows are extracted from the MAWI network packet traces [39] (SamplepointF in MAWI archive), which are collected daily with a 15min-long interval from a network transit link connected between Japan and the USA. During the network recording period, the samplepoint was made of a 1Gbps network traffic link. Additionally, the network packet payloads are removed, and sensitive network packet header fields are anonymized.

The labeling of network traffic is a challenging task, which is typically only achieved through human assistance [3]. Unfortunately, using an administrator for the labeling of network events becomes unfeasible due to the huge number of network events available in our dataset ($\approx$ 7.23 billion). In general, the literature performs the labeling task through misuse-based techniques (e.g., intrusion signatures) [6], [10], which may introduce labeling flaws, rendering the classification task easy to be performed. The ML classifier may learn the underlying used set of signatures, rather than the network traffic behavior. Consequently, our dataset was labeled using MAWILab [40] work, which label the daily anomalous events (network flows) from MAWI making use of a variety of unsupervised anomaly detectors. Although such a labeling procedure is prone to errors, it copes with the behavior variability of the analyzed network traffic, as the labeling task is performed in a daily basis. It maintains the variability of network



(a) Number of network flows over time. (b) Distribution of network flows over time. (c) Distribution of attack network traffic over time. (d) Distribution of normal network traffic over time.

Fig. 1. *MAWIFlow* network flow distribution throughout the 4 selected years.

TABLE II
MAWIFlow DATASET STATISTICS

Property	***Value***
Average Daily Network Packets	120 Millions
Average Daily Network Flows	23 Millions
Average Daily Throughput	720 Mbps
Average Daily Anomalous Flows	2.1 Millions
Average Daily Dataset Size	23.2GB
Total Network Packets	32.36 Billions
Total Network Flows	7.23 Billions
Total Dataset Size	8.3TB

traffic behavior and the challenges of production environment network traffic.

In this work, we consider the whole network traffic available for a four-year interval, ranging from 2016 to 2019. Network anomalies are classified according to their attack types as labeled by MAWILab. In such a case, the network anomalies can be of several types, including Service Scan, TCP Scan, and denial-of-service, among other network-level attacks.

Due to its large scale, over 8TB of data composed of over 7 billion network flows, the *MAWIFlow* dataset was built using the BigFlow [10] feature extraction algorithm, which is implemented in a Big Data processing framework. The feature extraction algorithm extracts, for each network flow, 39 features in a 15s window interval, as listed in Table I. Each network flow is previously labeled as normal or attack as established by the MAWILab [40] unsupervised algorithms output. Table II shows the *MAWIFlow* statistics throughout the evaluated period of 4 years.

The built dataset provides the expected properties from a realistic network traffic dataset used for the benchmark of intrusion detection techniques [12]. The dataset provides real, valid, and correct network traffic as it was collected from a real network transit link, maintaining the characteristics of production environments. The available network traffic is highly diverse, with a completeness behavior of communication protocols due to the extended recording period, considering a 4-year long time window. Notwithstanding, usability is ensured as the dataset is provided in a network flow format in a publicly available setting.

B. Reliability of ML for Intrusion Detection

The present evaluation aims to answer the following research questions:

- (**RQ1**) *What is the behavior of widely used ML-based approaches in terms of accuracy over time when no model updates are performed*?
- (**RQ2**) *What is the impact of periodic model updates on the accuracy of widely used ML-based approaches*?

In the following, we elaborate more on the evaluations carried out and our findings.

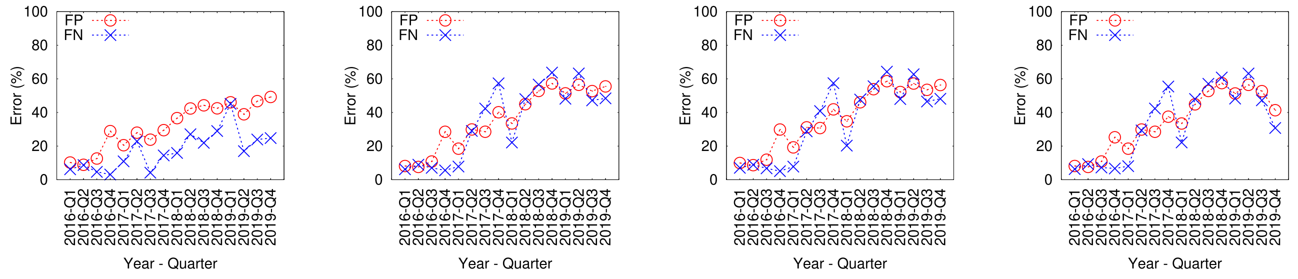
To evaluate widely used ML-based approaches, we select four commonly used ML-based techniques for intrusion detection, namely Long Short-Term Memory (LSTM), Random Forest (RF), Adaboosting (Ada), and Gradient Boosting (GBT).

The ensemble classifiers (RF, Ada, and GBT) were implemented with 100 decision trees as their base-learners, where each one of them also uses *gini* as the node split quality metric. The GBT classifier relies upon a 0.1 learning rate value, with *deviance* as the loss function. The Ada classifier uses the *SAMME* as boosting algorithm and 1.0 as the learning rate. The ensemble classifiers were implemented through *scikit-learn* API *v*0.24.

The LSTM was implemented with the following architecture: *(I) Input:* The 39 flow-based network features are fed as an input to the LSTM; *(II) LSTM:* Two LSTM layers with 256 and 128 units respectively. *(III) Output:* Two *dense* layers implemented with a *relu* activation function, with 512 and 1 units respectively. The implemented LSTM architecture used the *categorical crossentropy* as *loss* using *adam* optimizer. For the model building procedure, 1,000 epochs are executed with a batch size of 512. It is important to note that the used parameters from LSTM were set from the related works, and no significant differences were found while varying them.

As shown in Figure 1(d), due to the unbalanced nature of the *MAWIFlow*, a random undersampling without replacement is used at each training procedure to balance the occurrence between the classes for both the ensemble and the LSTM classifiers. The classifiers were evaluated concerning their false-negative rates (FN), false-positive rates (FP), and F-Measure. The following classification performance metrics were used:

- *True-Positive (TP):* number of attack samples correctly classified as attack.
- *True-Negative (TN):* number of normal samples correctly classified as normal.
- *False-Positive (FP):* number of normal samples incorrectly classified as an attack.



(a) LSTM (b) Random Forest (c) Adaboosting (d) Gradient Boosting

Fig. 2. Accuracy performance, shown quarterly, for various ML algorithms on the entire *MAWIFlow* dataset. Classifiers are trained with January 2016 data and not updated throughout time.

- *False-Negative (FN):* number of attack samples incorrectly classified as normal.

The F-Measure was computed according to the harmonic mean of precision and recall values while considering attack samples as positive and normal samples as negative, as shown in Eq. (3).

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F\text{-}Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

C. Classification Accuracy Without Periodic Model Updates

The first experiment aims at answering $RQ1$ and performs the evaluation of the classification performance of the selected ML techniques when model updates are not performed over time. The selected ML classifiers are trained only with the data from January 2016, while they are used for evaluation throughout the remaining four years of the interval. The evaluation goal is to measure how network traffic's evolving behavior affects the classification performance over time if model updates are not periodically performed, as commonly made by related works.

Figure 2 shows the average classification error for every quarter of the selected ML-based intrusion detection techniques without periodic model updates. A significant impact on the accuracy performance can be noticed compared to its training period (i.e., January 2016). All evaluated intrusion detection approaches decrease their accuracy as the model longness increases. For instance, compared to the training phase, the RF classifier increased, on average, 7.5% and 0.9% the FP and FN rates in 2016. Besides, from 2017 to 2019, the RF classifier increased its FP and FN rates even further compared to the training period, presenting an average increase of 38.3% and 35.3% on their FP and FN rates, respectively.

In contrast, the LSTM classifier presented lower accuracy degradation as time passes. For instance, it reached an average FN rate of only 13% in 2017. The reduced LSTM accuracy degradation was reached due to the complexity increase in the used model, given that the traditional shallow classifiers (RF, Ada, and GBT) cannot depict all the complexities of the network traffic used during the training phase. Consequently, the resulting model reaches a higher generalization regarding network traffic classification as time passes.

The performance of traditional ML-based intrusion detection approaches is dependent on the used model longness. The average quarterly accuracy varies significantly, from 2017-Q4 to 2018-Q1 for all the evaluated classifiers. The non-stationary behavior of accuracy over time significantly degrades the reliability of the intrusion detection mechanism, assuming that the administrator will discard signaled alerts as soon as higher FP and FN ratios are experienced. Even if an outdated model reaches high detection accuracy (e.g., those achieved by the RF classifier in 2018-Q1), the system alerts will be discarded, as its accuracy has already degraded in previous circumstances.

The accuracy degradation of the selected classifiers is caused by changes in the network traffic behavior as time passes. It can be seen in Figure 1(b), which shows the distribution of attack occurrence on the *MAWIFlow* dataset, that the attack occurrence remains stable throughout 2016. Similarly, the model detection accuracies are also not significantly affected (see Figure 2). In contrast, the attack occurrence significantly changes in 2017, caused by a significant increase in TCP Scan, followed by a decrease in UDP Scan and DoS attacks, resulting in the accuracy degradation of the evaluated techniques. Notwithstanding, the normal traffic occurrence also changes as time passes, as shown in Figure 1(c).

Over time, new network services will be provided, and new network attacks will be discovered, thus changing the underlying behavior of the network. However, identifying such changes in the underlying network traffic (which affects the deployed ML model) is difficult, as the administrator must (often manually) evaluate the current model accuracy, as defined by the ratio of correctly classified network samples. In contrast, the proper event label is unavailable in production settings, making the accuracy evaluation significantly more challenging. As a result, the deployed ML model must be regularly updated to address such network traffic changes.

To further investigate the accuracy variability over time when no model updates are performed, we compare the accuracy distribution of the accuracies obtained at the training (January 2016) to those obtained in the remaining period. Figure 3 shows the accuracy distribution (*boxplot*) of evaluated classifiers without model updates over time.

Regardless of the detection technique, the error ranges significantly increase as time passes when compared to those

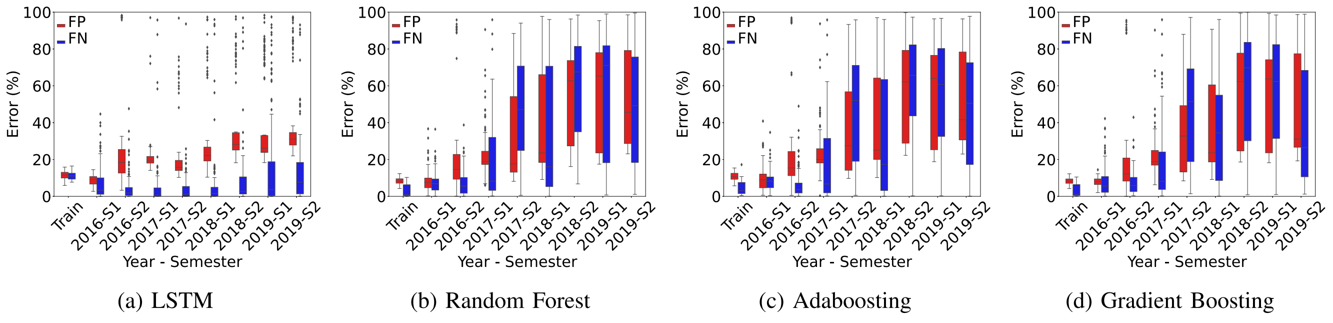


Fig. 3. Accuracy distribution, shown semiannually, for various ML algorithms on the entire *MAWIFlow* dataset. Classifiers are trained with January 2016 data and not updated throughout time.

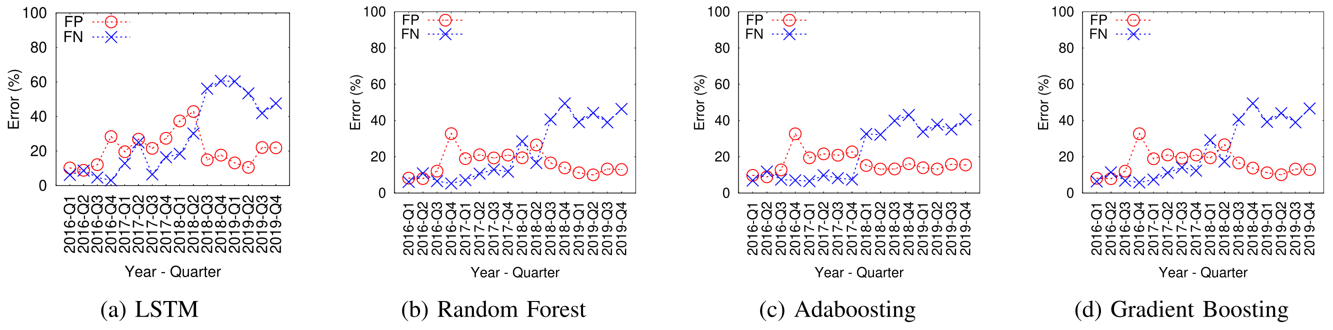


Fig. 4. Accuracy performance, shown quarterly, for various ML algorithms on the entire *MAWIFlow* dataset. Classifiers are updated every 6-month interval, with 1 month of training data.

measured during the training period. For example, on average, the RF interquartile error range increases by 3.4% and 4.0% for FP and FN rates every six months of additional model longness, thus significantly increasing its accuracy variation over time. The evaluated techniques cannot provide reliability in intrusion detection months after the model classifier training phase. Nonetheless, the accuracy distribution increases as time passes, as noted by an increase in the number of outliers (Figure 3, 2016 and 2017) and also the interquartile ranges over time (Figure 3, 2018 and 2019).

One can notice that the evolving behavior of network traffic, which varies as time passes, poses a significant challenge to ML-based techniques. It becomes unfeasible for the network administrator to generate a training dataset with all possible variations. Thus, the deployed ML model will present a high accuracy variation if it is not designed to consider the generalization capabilities. It should be noted that a high variation in the error rate over time for the intrusion detection mechanism significantly affects the system administrator's perception of system reliability.

This lack of reliability is because, even if the intrusion detection mechanism can reach high accuracy rates with an outdated ML model, the administrator will suppress alerts as soon as a high error rate is experienced. Intrusion detection mechanisms must reach high accuracy rates and present low accuracy variability over time for reliable production deployment. However, if the ML model is not updated, the evaluated ML classifiers is not able to cope with the network traffic's evolving behavior.

The techniques become unreliable in the immediate months following the training period, as noted by a higher error rates (Figure 2) and higher variability in the measured error rates (Figure 3). Therefore, intrusion detection approaches based on ML must be regularly updated to maintain classification reliability.

D. *Classification Accuracy With Model Updates*

To answer *RQ*2, we further investigate the impact that periodic model updates cause on the classification accuracy of the underlying ML model. We first consider a model longness of 6 months using a 1 month range as training data, which means we update the ML model every semester using the *MAWIFlow* data that occurred a month earlier. The evaluated classifiers are updated at the end of every January and July at any given year – using the data that occurred in the last 30 days at that given period. Notwithstanding, it is essential to mention that the frequency of model updates must be established according to the administrator's needs.

Figure 4 shows the accuracy performance throughout *MAWIFlow* data when model updates are performed every 6 months. One can notice that, in general, the evaluated classifiers' accuracy performance is improved. For instance, the periodically updated RF classifier improved its average FP rate from 35.1% to 16.5% while improving its average FN rate from 36.0% to 23.4% compared to the results without model updates. The model longness varies over time, as noted by higher FP and FN rates in 2018 and 2019. The updated LSTM could not significantly improve the obtained accuracy rates compared to the shallow classifiers. In contrast, it decreased the average accuracy rates as soon as a higher ratio of attacks is evidenced (Figure 1(a), 2018 and 2019). This decrease happens

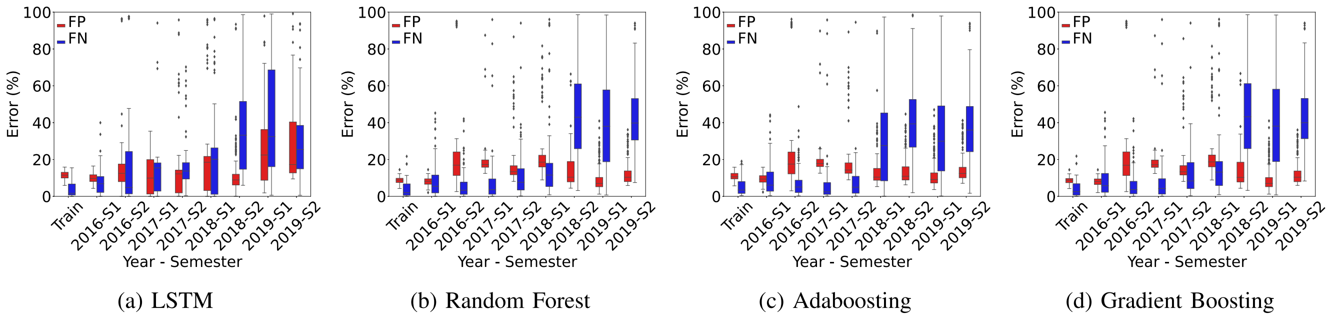


Fig. 5. Accuracy distribution, shown semiannually, for various ML algorithms on the entire *MAWIFlow* dataset. Classifiers are updated every 6-month interval, with 1 month worth of training data.

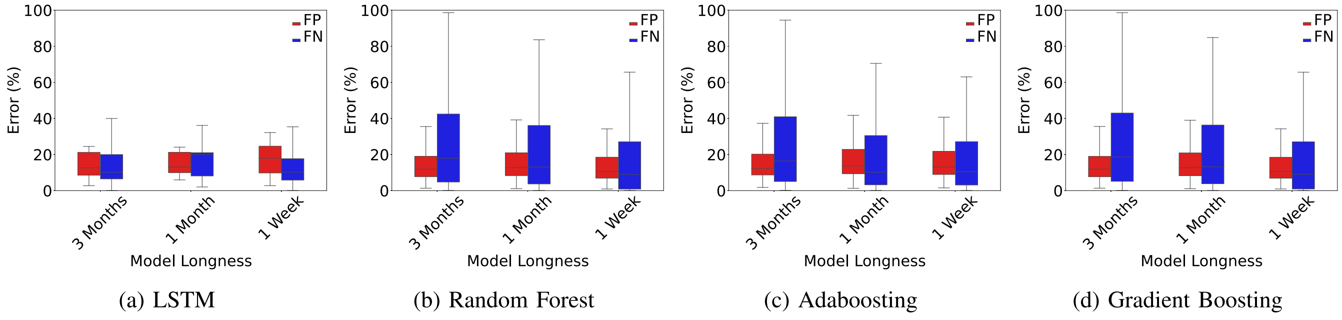


Fig. 6. Model longness and average error rate tradeoff in *MAWIFlow*. Model longness establishes the model update frequency, while average error rate is measured as the average of both FP and FN rates throughout the entire *MAWIFlow* data.

because as the complexity of the LSTM model increases when compared to traditional shallow classifiers, a higher number of training samples must be provided to ensure the proper model generalization, as occurred with its no-update counterpart.

The classifiers evaluation should have been updated more frequently to increase reliability in such a case. Due to the difficulties related to identifying model obsolescence, the administrator will increase the model update periodicity, regardless of whether the current model is still reliable, as occurred from 2016 to 2017. The sudden increase in network attacks shows the need for a more frequent model update. As shown in Figure 1(a), the attack occurrence increases from $\approx 6\%$ throughout 2016 and 2017 to $\approx 24\%$ in 2018 and 2019.

The changes in the occurrence of network attacks significantly affect the classification accuracy of the periodically updated classifiers, as noted by the degradation of the attack classification performance compared to the normal classification performance (Figure 4, the significant increase in FN when compared to FP). As the attack occurrence increases, so does the complexity of detecting attacks on the network, thus requiring more frequent updates of the ML model. The updated classifiers can provide high detection accuracy to previously known threats, considering that older attacks are still frequent as time passes (see Figure 1(b)).

We also investigate the accuracy variability when model updates are performed. Figure 5 shows the accuracy distribution (*boxplot*) of evaluated ML classifiers when model updates are performed every semester. In contrast to the approach without model updates (Figure 3), periodic model updates also decrease the accuracy variation over time, as noted by a lower interquartile range in both 2016 and 2017. Noteworthy, in 2018 and 2019 (the period that demands more frequent model updates), the interquartile range increases up to its counterpart with no periodic updates. Even considering a 6-month longness, an outdated ML model may significantly decrease and vary its accuracy, impacting the intrusion detection reliability.

We further investigate how the model update periodicity can affect the classification performance of the selected ML classifiers. Figure 6 shows the model accuracies according to the model longness, i.e., the frequency in which model updates are performed. It is possible to note that increasing the model update periodicity, in most cases, improves the accuracies of the obtained model. For instance, the FN interquartile range of the weekly updated RF decreased by 38% when compared to its counterpart updated every trimester. The LSTM classifier is not as significantly affected by the increase in the model update periodicity. More specifically, the 1-month model longness LSTM decreased the FN interquartile range by only 4.1% when compared to its 3-month model longness counterpart. This is because, as discussed previously, the higher complexity of the LSTM, compared to the shallow classifiers, can increase the model's longness.

E. Discussion

In general, previous works assume a static network traffic behavior, using datasets that do not represent the dynamic nature of production environments, resulting in imprecise and unrealistic detection schemes. Unlike the ones in the literature, our built dataset *MAWIFlow* is the first of its kind and is composed of more than 8TB of data that spans four years.

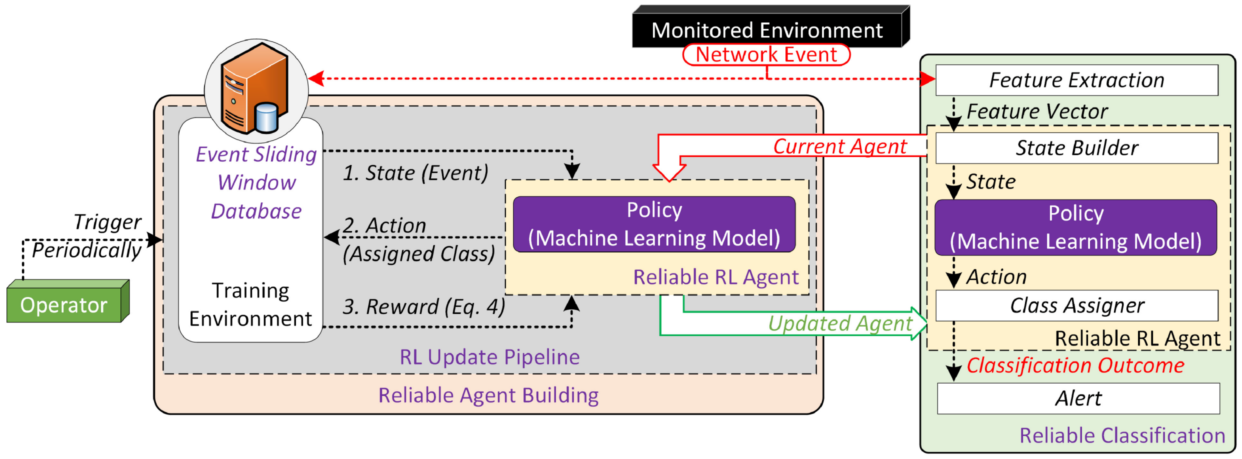


Fig. 7. Overview of the reliable reinforcement learning intrusion detection model classification and update pipelines.

The dataset enabled us to evaluate the current approaches in the literature regarding their reliability over time. The experiments have shown that widely used ML-based techniques cannot cope with the evolving behavior of network traffic. As time passes, their detection accuracy decreases while the accuracy variation increases.

The execution of periodic model updates can positively impact the accuracy and decrease its variance. However, it should be made according to the network traffic distribution. Current literature approaches must be used with small model longness (e.g., a few days or weeks) to provide reliability over time. Due to the small model-longness expectation, previously proposed techniques are rarely used in production environments. The main reason is that the ML model with such a small lifespan can not be updated as frequently as needed to ensure system reliability.

V. Reliable Reinforcement Learning Intrusion Detection Model

We present in this section a novel reliable reinforcement learning model for intrusion detection to address the aforementioned evolving behavior of network traffic. Our primary goal is to maintain the reliability in the system classification for more extended periods without model updates. Additionally, our solution aims to decrease the human and computational resources required to conduct such a task when model updates are performed. Figure 7 shows an overview of our proposed model, which is organized into two main stages, the *Reliable Classification* and the *Reliable Agent Building*.

Reliable Classification is performed through a reinforcement learning classification pipeline. Its purpose is to classify network events over time in production deployment either as normal or attack. The network events are depicted as an environment state, while the reinforcement learning agent outputs an action representing the classified event class according to the underlying RL policy.

The goal of the *Reliable Agent Building* is twofold: the building of the proposed scheme at the initial deployment and the update of an already deployed outdated agent. The agent's first deployment is executed in a reinforcement learning approach with an agent built from scratch. The model update decreases the computational costs by leveraging the current outdated agent deployed in production, following a transfer learning rationale. In such a case, a reduced set of events can be used while leveraging prior knowledge of the current outdated agent, thus, it applies an event sliding window mechanism to the production environment—e.g., updating the agent with events from the last week.

Our scheme can build agent policies through a novel reinforcement learning algorithm that extends the obtained model longness. Our proposal insight is that one can increase the model longness through model correctness rather than only focusing on model accuracy. Our scheme represents the model correctness difference between the model output classification confidence and the correct event label. As a result, the model training aims for higher accuracy and approximation to the event label, considering its distance to the correct event class.

In the following subsections, we detail our proposal, present the architecture of the modules that implement it, and describe its main components.

A. Reliable Agent Building

Reinforcement learning techniques have produced promising results in several fields where agent rewards can be autonomously collected as a direct consequence of a given action. In contrast, intrusion detection systems may not easily obtain the correct event label in production deployments, thus, rewards cannot be given in an appropriate time. In other words, an action may be taken, but the rewards may not be given adequately. Intrusion detection systems must have prior access to the label of the events during the model building, enabling appropriate rewards.

Our proposed model's primary goal is to leverage the reinforcement learning technique to build an agent policy with an increased longness in production, i.e., presenting a longer lifespan. The *Reliable Agent Building* approach deals with intrusion detection as a reinforcement learning task, making use of the following relations:

- *Environment:* The intrusion detection training dataset which contains a set of network events expected to occur in production deployment.
- *State:* Current *environment* event that the intrusion detection agent should analyze. It can be made of a *normal* or an *attack* event.
- *Action:* The *agent* classification outcome of a given *environment state* either as *normal* or *attack*.
- *Reward:* The reward that an agent receives by a given *action* output according to the current *environment state*.

The rationale of our intrusion detection as a reinforcement learning task is shown in Figure 7 (*Reliable Agent Building*). One can notice that an agent receives a training dataset environment state (i.e., an event) as input. It acts using a class assigned per event, receiving the corresponding reward. The training procedure can be executed using a new agent (e.g., the first deployment model) or an outdated one (e.g., a model update). The operator is responsible for periodically triggering the reliable agent-building process (e.g., every semester).

One should note that the model update periodicity must be defined according to the administrator's needs. Our proposal relies on an event sliding window to provide easiness on model updates in such a case. We assume that a fewer number of events can be used at model updates if previous knowledge of the historical environment behavior is available. Therefore, we leverage the outdated agent to represent the environment's historical behavior while updating it through a transfer learning rationale.

As a result, model updates can be performed as a simple agent policy fine-tuning instead of building it again from scratch. A model update that can be performed with fewer events will require less human intervention for the labeling procedure and less computational costs.

The building of a new training dataset in our model is performed through a sliding window of production environment events. In such a case, the network data that occurred N days before the model update task must be stored and properly labeled (Figure 7, *Event Sliding Window Database*). The administrator may use unsupervised ML techniques, misuse-based detection approaches, or even store the network data for extended periods until the proper attack disclosure in public domains. Even if the model update procedure demands an extended period for its execution, the proposed model will keep its reliability, as it is also designed to increase the model longness, i.e., its lifespan. Besides the easiness of model updates, the model-longness challenge must also be addressed.

Our proposal uses the classification correctness metrics as rewards in the training phase to extend the ML model's longness. The computation of our agent reward is shown in Equation (4), where $confidence^{inst_i}$ depicts the underlying ML model classification confidence on a given instance $inst_i$, FP denotes false positives, and FN denotes false negatives.

$$reward = \begin{cases} 1 - confidence^{inst_i} & \text{if } FP \text{ or FN} \\ confidence^{inst_i} & \text{otherwise} \end{cases} \quad (4)$$

Confident classifications on misclassified events receive fewer rewards, while correctly classified ones receive their classification confidence as a reward. Thus, rewards are computed as a classifier confidence approximation measure from the correct event class. The classifier confidence values are agnostic to the used classifier algorithm. For instance, the Random Forest classifier outputs its confidence values as the ratio of its base decision trees that classifies a given instance as the event-assigned class.

Our proposal's insight is that we can increase the model longness while decreasing accuracy variation using the classification correctness, as measured through the classifier confidence values, rather than only pursuing a higher accuracy. As a result, the built ML model will provide better classification correctness in all input events instead of increasing the accuracy in a subset (resulting in less longevity caused by overfitting in the data training model).

To implement our novel reinforcement learning model for intrusion detection, using the proposed reward measure, we develop a version of the well-known reinforcement learning $Q - Learning$ algorithm [41]. The implementation, executed at each model training or update round (Figure 7, *Reliable Agent Building*), is shown in Algorithm 1.

The algorithm receives a training dataset, similar to the traditional $Q - Learning$ algorithm, which acts as the environment states (S). It initializes an agent either arbitrarily, in the model's first deployment, or with an outdated agent, in a model update process that follows the transfer learning rationale. The algorithm is executed until it has converged by reaching, for instance, an expected accuracy level or a predefined number of iterations. In brief, it randomly selects at each iteration a state (i.e., an instance) from the training dataset (i.e., an environment), outputs a related action (i.e., confidence), and receives a reward accordingly (Eq. (4)). Thus, it approximates the underlying classification model confidence values to the correct label of each event to maximize its rewards over time, improving its generalization capacity over the whole training data (environment).

B. Reliable Classification

The operation of our proposed scheme for the classification task in production deployments is shown in Figure 7 (*Reliable Classification*). It receives a network traffic event as input from the monitored environment (e.g., a network packet). The collected data is then represented as a feature vector, extracted by a feature extraction module (e.g., extraction of several flow-based features as shown in Table I). The feature vector is fed as input to the reliable reinforcement learning agent deployed in production. The agent then depicts the feature vector as the environment state (Figure 7, *State Builder*) and applies the agent policy with its underlying ML model. The policy outputs the action (confidence values) and the classification outcome can be established (Figure 7, *Class Assigner*). Finally, an alert can be generated if the event is classified as an attack.

C. Discussion

Network traffic's evolving behavior is challenging for ML-based intrusion detection (see Section IV). Our proposal aims

Algorithm 1 Proposed Intrusion Detection Q-Learning Algorithm for Reliable Reinforcement Learning Agent

```
Require:
  States $S = \{instance_1, \ldots, instance_n\}$                    ▷ Training dataset
  Actions $A = \{normal, attack\}$                                   ▷ ML NIDS Classes
  Reward function $R{:}S \times A \rightarrow \mathbb{R}$            ▷ Reward function as computed through Eq. (4)
  Transition function $R : S \times A \rightarrow \mathbb{S}$
  Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$
  Discounting factor $\gamma \in [0, 1]$
  procedure QLearning($S$, $A$, $R$, $T$, $\alpha$, $\gamma$)
    Initialize $Q : S \times A \rightarrow \mathbb{R}$ arbitrarily, or outdated agent
    while $Q$ is not converged do
      Start in random state $s \in S$
      while $s$ is not terminal do
        Calculate $\pi$ according to Q and exploration strategy (e.g. $\pi(x) \leftarrow arg\,max_a Q(x, a)$)
        $a \leftarrow \pi(s)$                                         ▷ Establishes state classification confidence
        $r \leftarrow RewardFunction(s, a)$                           ▷ Receive the reward as computed through Eq. (4)
        $s' \leftarrow GetInstance(instance_i + 1)$                   ▷ Receive the new state
        $Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$
        $s \leftarrow s'$
      end while
    end while
    return $Q$
  end procedure
```

to address three main aspects related to network traffic behavior changes over time: the *model updates*, *model longness*, and *accuracy variation*. Our proposal leverages prior knowledge about the environment to facilitate model updates using a transfer learning rationale. The assumption is that the outdated agent can be used in the model update task to decrease the computational costs, the amount of needed training data, and the costs associated with the event labeling task. The proposed model correctness pursues longer model longness. We assume that one should seek the model approximation to all training events to increase the model's longness. Therefore our proposal does not favor only a subset of events, as commonly made in the literature by accuracy-based approaches. The accuracy variation is decreased by the model longness – trained to aim for higher model correctness' over all the training data.

VI. Evaluation

The evaluation of our proposed model aims at answering the following additional research questions:

- (**RQ3**) *How does our proposed model perform without model updates*?
- (**RQ4**) *What is the impact of model updates in our proposal*?
- (**RQ5**) *Can our proposed model provide higher reliability than traditional techniques*?

The following sections describe how we build our model and its performance on *MAWIFlow* dataset.

A. Model Building

Our reinforcement learning proposal for intrusion detection (Algorithm 1) was implemented on top of OpenAI Gym API [42]. At each model training or update, the algorithm creates a testbed environment that reproduces the proposed training procedure (see Section V-A). The proposal relies on a Multi-Layer Perceptron (MLP) as a reinforcement learning policy (i.e., ML model).

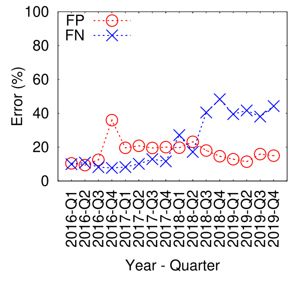
The algorithm enables our proposed model update procedure (Section V-A) to be executed on top of the older MLP neurons through a transfer learning procedure. The data from January 2016 was used (30 days) for the first model training to enable the subsequent evaluation of our model updates, which will rely upon the outdated agent. The model update procedure relies on only 7 days worth of data (Figure 7, *Event Sliding Window Database*). Thus, we perform model updates using less data than evaluated previously (see Section IV, 7 *vs.* 30 days of training data) to properly evaluate how our model performs with less training data during model updates.

The MLP with our proposed algorithm is executed at each model update with a learning rate of 0.3, a momentum rate for the backpropagation algorithm of 0.2, and uses the TensorFlow API [43]. The algorithm executes 5.000 epochs to build the model, where each epoch performs 100 turns. Each turn computes the proposed algorithm policy gradients according to the obtained rewards (see Eq. (4)) from the classification of 10 thousand training instances.

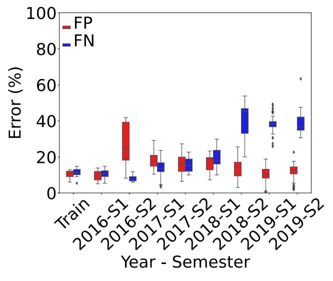
As a convergence criterion during the model update, the algorithm either executes 5.000 epochs or reaches 90% of accuracy. These parameters were identified empirically, resulting in similar classification results.

B. Classification Accuracy Without Updates

The first experiment aims at answering *RQ*3 and evaluates our proposed model performance when no model updates are



(a) Model accuracy over time



(b) Daily accuracies distribution

Fig. 8. The over time proposal accuracy performance, run on *MAWIFlow* dataset. The proposal is trained with January 2016 data and not updated throughout the time.

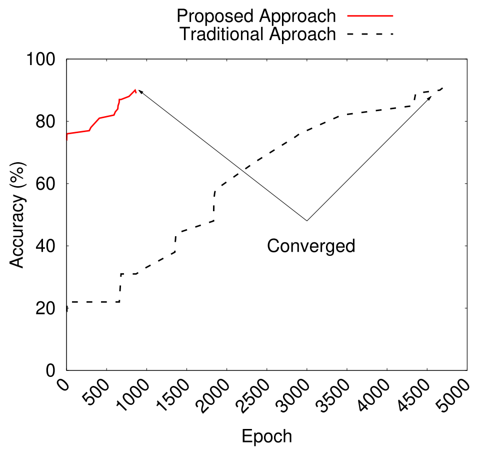


Fig. 9. Training convergence of our proposal at the 2nd semester of 2016, considering an outdated agent used in the model update process and its comparison with the retraining from scratch. The accuracy was measured as the ratio of total events correctly classified. Similar results were found at every model update procedure execution.

performed as time passes. Similar to what was previously made, we apply our proposed algorithm to create an agent policy using data from January 2016 of *MAWIFlow* as the training environment (Figure 7). The obtained agent is used throughout the whole *MAWIFlow* data without model updates. Figure 8(a) shows the average error performance considering both FP and FN rates of our proposed model on a three-month periodicity.

Our proposed model kept its reliability for extended periods, maintaining its FP rates closer to those measured at the classifier training phase throughout the four years. For instance, our proposed technique presented an average error rate of 18.9% and 8.8% for FP and FN, considering the first deployment year (i.e., 2016). In our proposal, each month after the training period, on average, increases 4.2% and 0.3% in FP and FN rates, respectively, considering a model longness of 1 year. Additionally, the proposed model presented average FP and FN rates throughout the four years of 16.4% and 23.5%.

Our model reached similar accuracy rates obtained by traditional techniques, such as the RF classifier with a 6-month longness (Figure 4), which presented 16.5% and 23.4% of FP and FN rates. Therefore, our model provided high detection accuracy even when no model updates occurred, with similar accuracy rates obtained by the literature's techniques with a 6-month model longness.

We also investigate how our proposed model's accuracy varies over time without model updates, as shown in Figure 8(b). The model accuracy rate variation is significantly lower when compared to traditional ML approaches. The proposed method presented average interquartile ranges in 2016-S1 of only 3.4% and 1.9% of FP and FN, respectively. In contrast, traditional approaches (e.g., RF) presented an average interquartile range in 2016-S1 of 3.1% and 6.1% of FP and FN rates, respectively. Additionally, if the whole *MAWIFlow* 4 years of traffic is considered, our proposal presents an average interquartile range of 13.5% and 13.3% of FP and FN rates, while the RF classifier presents 29.3% and 36.6% of interquartile range (Figure 3), respectively in both cases. Consequently, our proposed approach increases the intrusion detection schemes' longness by improving the systems' accuracy and reducing the accuracy variation over time.

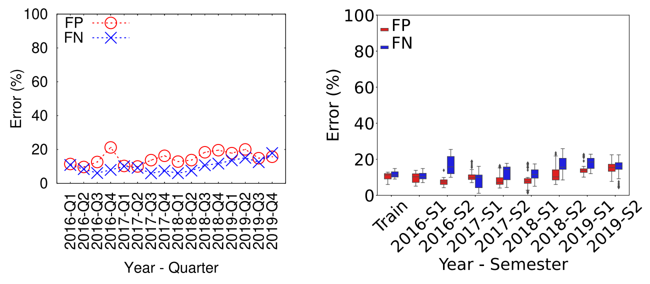
C. Classification Accuracy With Updates

To answer *RQ*4, we perform periodic model updates on our proposed scheme. In such a case, similarly to the experiments performed in Section IV, we execute our proposed update procedure every semester. However, only taking into account the events that occurred over the last seven days (Figure 7, *Event Sliding Window Database*). As our model leverages the previous knowledge through the outdated model in a transfer learning fashion (Section V-A), we first evaluate how the outdated ML model can ease the model update procedure.

Figure 9 shows the proposal convergence according to the model update epoch, while considering if the outdated model is used or not in the second semester of 2016. The proposed approach that leverages the outdated model converges demanding significantly fewer epochs to be executed, reaching 90% of accuracy rate with only 970 epochs even with only 1-week of training data. In contrast, if the outdated model is not used, 4610 epochs must be executed. Similar results were found regardless of the period in which the model update occurred.

The proposed approach that takes advantage of the outdated model can significantly decrease the computational needs in the training procedure due to the decrease in the number of epochs. It also decreases the number of events that must be labeled due to a higher model longness, hence, demanding fewer model updates. On average, the proposed approach that leverages the outdated model converged with only 21.1% of epochs while relying on only a week of training data compared to the traditional approach that performs model updates from scratch.

Figure 10(a) shows the proposed approach accuracy over time with a 6-month model update periodicity. In such a case, the error rates are significantly lower and more stable as time passes. The periodicity of model updates enabled our proposal to reach high accuracy rates throughout the four years of *MAWIFlow* data. The proposed technique presented an average of 14.5% and 10.0% of FP and FN rates over time, respectively. Figure 10(b) shows the accuracy variation over time of our proposed scheme with periodic model updates.



(a) Model accuracy over time (b) Daily accuracies distribution

Fig. 10. The proposal accuracy performance over time, run throughout *MAWIFlow* dataset. The proposal is trained with January 2016 data and updated every semester with 1-week data.

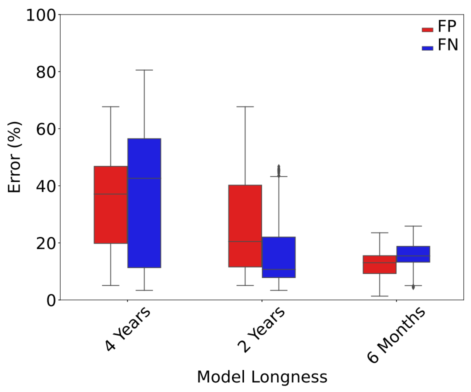


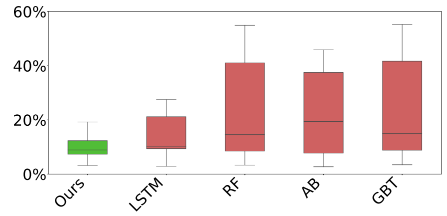
Fig. 11. The longness and average error rate tradeoff in the *MAWIFlow* proposed dataset. Model longness establishes its update frequency, while the average error rate is measured as the mean of both FP and FN rates throughout the entire *MAWIFlow* data.

The accuracy variation significantly decreases, presenting an average interquartile range of only 4.4% and 8.5% of FP and FN rates, while the traditional ML techniques (e.g., RF) (Figure 5) present an average of 16.9% and 5.3% (i.e., 3.84× more FP for only a 0.37× reduction in FN), respectively in all cases. Therefore, our proposed model significantly increases the intrusion detection scheme's reliability over time by extending the model longness and decreasing the accuracy variation while maintaining the accuracy over time.

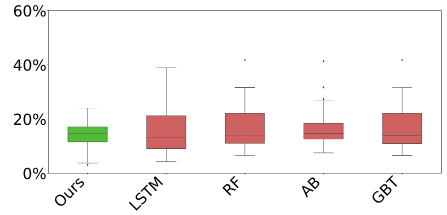
We further investigate how the model update periodicity impacts our proposal's accuracy. Figure 11 shows the relation between model update periodicity and accuracy rates of our scheme. Our proposal reaches significantly higher accuracy rates even when the longness of the model is considered. More specifically, even with a 2-year long model longness, it reaches better classification accuracy than traditional techniques that consider a 1-month model longness value (Figure 6).

In summary, our proposal increases the model longness and decreases the model update periodicity without degrading the accuracy. Nonetheless, it achieves accuracy rates higher than traditional techniques even when no model updates occur.

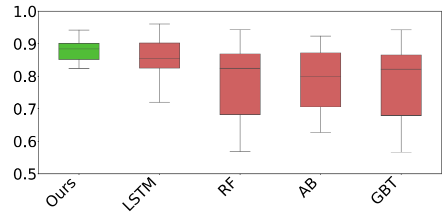
Finally, to answer the question *RQ*5, we compare the accuracies obtained by our model and those from traditional ML techniques. Figure 12 shows the monthly distribution of accuracy rates of each evaluated classification scheme with a 6-month model longness. In such a case, our proposed model



(a) False Negative



(b) False Positive



(c) F-Measure

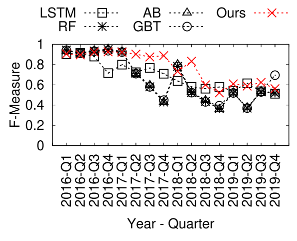
Fig. 12. Monthly accuracies for several classification techniques considering a 6-month model update periodicity, run on four years of *MAWIFlow* data. The proposed model can provide lower interquartile ranges and achieve higher median accuracy than other evaluated techniques.

median was 14.3%, 8.4%, and 0.92 of FP, FN, and F-Measure, respectively. In contrast, the traditional techniques presented a median F-Measure of 0.83, 0.81, 0.78, and 0.81 for the LSTM, RF, Ada, and GBT respectively. Thus, our proposed scheme could also provide higher detection accuracies than deep learning schemes, such as the evaluated LSTM, improving the F-Measure by 0.09 while demanding only 20% of the required training data and significantly fewer computational resources for model updates.

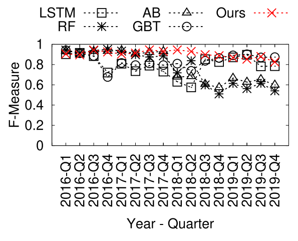
Figure 13 shows a performance comparison of our proposed model with the previously evaluated approaches. It is possible to note that our proposed scheme can significantly improve the model's longness (see Figure 13(a)). For instance, our proposal provides an average F-Measure throughout 2016 and 2017 of 0.91 without periodic model updates, while the RF classifier reaches only 0.78 (i.e., an improvement of 0.13). Additionally, if periodic model updates are performed, our proposed scheme reaches an average F-Measure of 0.92 throughout the 4 years of *MAWIFlow*, while the RF classifier, as an example, provides an average F-Measure of only 0.80. As a result, our proposed model can provide higher classification accuracies most of the time (months), improving the average FP rate by up to 8.0% and the average FN rate by up to 34.6% when compared to traditional techniques.

D. Limitations and Open Challenges

Network-based intrusion detection through ML techniques in the literature overlooks the challenges related to the



(a) Without updates



(b) With updates

Fig. 13. Comparison of the accuracy performance of the proposal over time with and without periodic model updates, run on the entire *MAWIFlow* dataset.

evolving behavior of network traffic, focusing their research efforts on obtaining higher accuracies, which often is only achieved with a tradeoff in the model lifespan. The proposed scheme significantly improved the ML model lifespan, providing higher accuracy after the training period than traditional techniques. However, despite the significant improvement in the average accuracy, the obtained FP and FN rates, due to the huge amount of network traffic, can still pose a significant challenge for deploying the proposed scheme in production environments. Several techniques can be used to decrease the FP or FN rates. For instance, the administrator may change the classifier operation point to increase the model update periodicity. She can use an ensemble of classifiers and correlate the triggered alarms according to the host source, as each network flow may trigger a corresponding alarm. Thus, a single attack may generate thousands of alarms, e.g., network-based denial-of-service.

VII. Conclusion

Novel approaches for intrusion detection through machine learning techniques have been extensively proposed in the scientific literature recently, while only a few were deployed in production. We showed in the paper that researchers incorrectly adopt traditional machine learning assumptions in the intrusion detection domain, such as assuming a static behavior of network traffic.

This paper proposed evolving these intrusion detection assumptions regarding network traffic behavior. To the best of our knowledge, the proposal is the first to build a dataset that evaluates intrusion detection schemes' reliability over time. Nonetheless, we proposed a novel reinforcement learning technique for intrusion detection with longer model lifespan, lower accuracy variation, and more accessible model updates. The proposed technique provided higher detection accuracy even when no model updates occur, and if made, the proposal significantly further decreases the accuracy variation while also improving the detection accuracy.

As future works, the researchers are encouraged to use our built dataset to increase system accuracy while providing a higher model lifespan. Thus, research conducted in our dataset should consider the tradeoff between model accuracy, model lifespan, model update periodicity, and the number of training samples. The built dataset used throughout this paper's experiments is publicly available for download at `https://secplab.ppgia.pucpr.br/reinforcemawiflow`.

References

[1] AO Kaspersky Lab. "Kaspersky Security Bulletin 2019. Statistics," 2019. [Online]. Available: https://securelist.com/kaspersky-security-bulletin-2019-statistics/95475/

[2] "Kaspersky Security Bulletin 2022. Statistics." 2022. [Online]. Available: https://securelist.com/ddos-attacks-in-q1-2022/106358/

[3] B. Molina-Coronado, U. Mori, A. Mendiburu, and J. Miguel-Alonso, "Survey of network intrusion detection methods from the perspective of the knowledge discovery in databases process," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2451–2479, Dec. 2020.

[4] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Security*, vol. 86, pp. 147–167, Sep. 2019. [Online]. Available: https://doi.org/10.1016/j.cose.2019.06.005

[5] N. Hubballi and V. Suryanarayanan, "False alarm minimization techniques in signature-based intrusion detection systems: A survey," *Comput. Commun.*, vol. 49, pp. 1–17, Aug. 2014. [Online]. Available: https://doi.org/10.1016/j.comcom.2014.04.012

[6] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 305–316. [Online]. Available: https://doi.org/10.1109/sp.2010.25

[7] C. Gates and C. Taylor, "Challenging the anomaly detection paradigm: A provocative discussion," in *Proc. Workshop New Security Paradigms (NSPW)*, 2006, pp. 21–29.

[8] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Exp. Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112963. [Online]. Available: https://doi.org/10.1016/j.eswa.2019.112963

[9] M. Injadat, A. Moubayed, A. B. Nassif, and A. Shami, "Multi-stage Optimized machine learning framework for network intrusion detection," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1803–1816, Jun. 2021. [Online]. Available: https://doi.org/10.1109/tnsm.2020.3014929

[10] E. Viegas, A. Santin, A. Bessani, and N. Neves, "BigFlow: Real-time and reliable anomaly-based intrusion detection for high-speed networks," *Future Gener. Comput. Syst.*, vol. 93, pp. 473–485, Apr. 2019.

[11] G. Folino, F. S. Pisani, and L. Pontieri, "A GP-based ensemble classification framework for time-changing streams of intrusion detection data," *Soft Comput.*, vol. 24, no. 23, pp. 17541–17560, Aug. 2020.

[12] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 5, pp. 516–524, Sep. 2010. [Online]. Available: https://doi.org/10.1109/tsmcc.2010.2048428

[13] D. Arp *et al.*, "DoS and dont's of machine learning in computer security," in *Proc. 31st USENIX Security Symp. (USENIX Security)*, Aug. 2022, pp. 3971–3988. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/arp

[14] F. Pinagé, E. M. dos Santos, and J. Gama, "A drift detection method based on dynamic classifier selection," *Data Min. Knowl. Disc.*, vol. 34, no. 1, pp. 50–74, Oct. 2019. [Online]. Available: https://doi.org/10.1007/s10618-019-00656-w

[15] E. Kim *et al.*, "Champion-challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning," *Exp. Syst. Appl.*, vol. 128, pp. 214–224, Aug. 2019. [Online]. Available: https://doi.org/10.1016/j.eswa.2019.03.042

[16] V. Dremin *et al.*, "Skin complications of diabetes mellitus revealed by polarized hyperspectral imaging and machine learning," *IEEE Trans. Med. Imag.*, vol. 40, no. 4, pp. 1207–1216, Apr. 2021. [Online]. Available: https://doi.org/10.1109/tmi.2021.3049591

[17] J. Memon, M. Sami, R. A. Khan, and M. Uddin, "Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR)," *IEEE Access*, vol. 8, pp. 142642–142668, 2020. [Online]. Available: https://doi.org/10.1109/access.2020.3012542

[18] S. Wassermann, T. Cuvelier, P. Mulinka, and P. Casas, "Adaptive and reinforcement learning approaches for online network monitoring and analysis," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1832–1849, Jun. 2021.

[19] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a hidden Naïve Bayes multiclass classifier," *Exp. Syst. Appl.*, vol. 39, no. 18, pp. 13492–13500, 2012.

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

[21] Z. Wang, Y. Liu, D. He, and S. Chan, “Intrusion detection methods based on integrated deep learning model,” *Comput. Security*, vol. 103, Apr. 2021, Art. no. 102177. [Online]. Available: https://doi.org/10.1016/j.cose.2021.102177

[22] Akashdeep, I. Manzoor, and N. Kumar, “A feature reduced intrusion detection system using ANN classifier,” *Exp. Syst. Appl.*, vol. 88, pp. 249–257, Dec. 2017.

[23] T. Hamed, R. Dara, and S. C. Kremer, “Network intrusion detection system based on recursive feature addition and bigram technique,” *Comput. Security*, vol. 73, pp. 137–155, Mar. 2018.

[24] E. K. Viegas, A. O. Santin, V. V. Cogo, and V. Abreu, “A reliable semi-supervised intrusion detection model: One year of network traffic anomalies,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.

[25] G. Caminero, M. Lopez-Martin, and B. Carro, “Adversarial environment reinforcement learning algorithm for intrusion detection,” *Comput. Netw.*, vol. 159, pp. 96–109, Aug. 2019.

[26] M. D. Mauro, G. Galatro, and A. Liotta, “Experimental review of neural-based approaches for network intrusion management,” *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2480–2495, Dec. 2020. [Online]. Available: https://doi.org/10.1109/tnsm.2020.3024225

[27] D. Upadhyay, J. Manero, M. Zaman, and S. Sampalli, “Gradient boosting feature selection with machine learning classifiers for intrusion detection on power grids,” *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 1104–1116, Mar. 2021. [Online]. Available: https://doi.org/10.1109/tnsm.2020.3032618

[28] S. Das *et al.*, “Network intrusion detection and comparative analysis using ensemble machine learning and feature selection,” *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 1104–1116, Mar. 2021. [Online]. Available: https://doi.org/10.1109/tnsm.2021.3138457

[29] S. Longari, D. H. N. Valcarcel, M. Zago, M. Carminati, and S. Zanero, “CANnolo: An anomaly detection system based on LSTM Autoencoders for controller area network,” *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1913–1924, Jun. 2021. [Online]. Available: https://doi.org/10.1109/tnsm.2020.3038991

[30] Y. Imrana, Y. Xiang, L. Ali, and Z. Abdul-Rauf, “A bidirectional LSTM deep learning approach for intrusion detection,” *Exp. Syst. Appl.*, vol. 185, Dec. 2021, Art. no. 115524. [Online]. Available: https://doi.org/10.1016/j.eswa.2021.115524

[31] R. Alshammari and A. N. Zincir-Heywood, “The impact of evasion on the generalization of machine learning algorithms to classify VoIP traffic,” in *Proc. 21st Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2012, pp. 1–8. [Online]. Available: https://doi.org/10.1109/icccn.2012.6289243

[32] R. Alshammari and A. N. Zincir-Heywood, “Machine learning based encrypted traffic classification: Identifying SSH and Skype,” in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, Jul. 2009, pp. 1–8. [Online]. Available: https://doi.org/10.1109/cisda.2009.5356534

[33] J. Liang and M. Ma, “Co-maintained database based on blockchain for IDSs: A lifetime learning framework,” *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1629–1645, Jun. 2021.

[34] Y.-F. Hsu and M. Matsuoka, “A deep reinforcement learning approach for anomaly network intrusion detection system,” in *Proc. IEEE 9th Int. Conf. Cloud Netw. (CloudNet)*, 2020, pp. 1–6.

[35] E. Suwannalai and C. Polprasert, “Network intrusion detection systems using adversarial reinforcement learning with deep Q-network,” in *Proc. 18th Int. Conf. ICT Knowl. Eng. (ICT KE)*, Nov. 2020, pp. 1–9. [Online]. Available: https://doi.org/10.1109/ictke50349.2020.9289884

[36] H. Benaddi, K. Ibrahimi, A. Benslimane, and J. Qadir, *A Deep Reinforcement Learning Based Intrusion Detection System (DRL-IDS) for Securing Wireless Sensor Networks and Internet of Things* (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering). Cham, Switzerland: Springer Int., 2020, pp. 73–87. [Online]. Available: https://doi.org/10.1007/978-3-030-52988-8_7

[37] A. Servin and D. Kudenko, “Multi-agent reinforcement learning for intrusion detection,” in *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*. Heidelberg, Germany: Springer, 2008, pp. 211–223. [Online]. Available: https://doi.org/10.1007/978-3-540-77949-0_15

[38] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, Jul. 2009, pp. 1–6. [Online]. Available: https://doi.org/10.1109/cisda.2009.5356528

[39] MAWI. “MAWI working group traffic archive—Samplepoint F.” 2021. [Online]. Available: https://mawi.wide.ad.jp/mawi/

[40] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, “MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking,” in *Proc. 6th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2010, pp. 1–8.

[41] C. J. C. H. Watkins and P. Dayan, “*Q*-learning,” *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/bf00992698

[42] OpenAI. “OpenAI—Gym.” 2022. [Online]. Available: https://gym.openai.com/

[43] TensorFlow. “TensorFlow API.” 2022. [Online]. Available: https://www.tensorflow.org/

Roger R. dos Santos received the B.S. degree in information systems from the Pontificia Universidade Catolica do Parana (PUCPR) in 2014, and the M.Sc. degree in computer science in 2020. He is currently pursuing the Ph.D. degree with PUCPR. His research interests include machine learning and computer security.

Eduardo K. Viegas (Member, IEEE) received the B.S. and M.Sc. degrees in computer science and the Ph.D. degree from the Pontificia Universidade Catolica do Parana in 2013, 2016, and 2018, respectively. His research interests include machine learning, network analytics, and computer security.

Altair O. Santin (Member, IEEE) received the B.S. degree in computer engineering from the Pontificia Universidade Catolica do Parana (PUCPR) in 1992, the M.Sc. degree from UTFPR in 1996, and the Ph.D. degree from UFSC in 2004. He is a Full Professor of Graduate Program in Computer Science and the Head of Security and Privacy Lab, PUCPR. He is a member of ACM and the Brazilian Computer Society.

Vinicius V. Cogo received the B.Sc. degree in computer science from UFSM, Brazil, and the M.Sc. and Ph.D. degrees in informatics from Ciências/ULisboa. He is an Assistant Professor with the Department of Informatics, Faculty of Sciences, University of Lisbon, where he is an Integrated Researcher with LASIGE Research Laboratory. His research interests include distributed systems, dependability, fault tolerance, storage of critical data, and cloud computing.