Facing the Unknown: a Stream Learning Intrusion Detection System for Reliable Model Updates

Eduardo K. Viegas, Altair O. Santin, Vinicius V. Cogo and Vilmar Abreu

Abstract Current machine learning approaches for network-based intrusion detection do not cope with new network traffic behavior, which requires periodic computationally and time-consuming model updates. This paper proposes a novel stream learning intrusion detection model that maintains system accuracy, even in the presence of unknown traffic behavior. It also facilitates the process of updating the model, gradually incorporating new knowledge into the machine learning model. Our experiments were performed using a recent realistic dataset of network behaviors and they have shown that the proposed technique detects potentially unreliable classifications. Moreover, the proposed model can incorporate the new network traffic behavior from model updates to improve the system accuracy while maintaining its reliability.

1 Introduction

Over the last years, machine learning (ML) techniques have been extensively used for the design of network-based intrusion detection systems (NIDS) [1]. However, despite the promising reported results, ML-based NIDS are rarely deployed in real-world production environments [7]. This is because the reported accuracy rates are often not achieved when the proposed detection schemes are used in production.

Eduardo K. Viegas
Pontifical Catholic University of Parana, Brazil (PUCPR), e-mail: eduardo.viegas@ppgia.pucpr.br

Altair O. Santin
Pontifical Catholic University of Parana, Brazil (PUCPR), e-mail: santin@ppgia.pucpr.br

Vinicius V. Cogo
LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal e-mail: vvcogo@fc.ul.pt

Vilmar Abreu
Pontifical Catholic University of Parana, Brazil (PUCPR), e-mail: vilmar.abreu@ppgia.pucpr.br

As a consequence, proposed ML-based NIDS are unreliable for real-world environments, remaining mostly as a research topic.

In NIDS, ML is mostly achieved by pattern recognition means [4]. In such a case, a behavioral model is built by the means of a training dataset, in general, made of network flows. The dataset comprises both normal and malicious activities, that are expected to be evidenced in production environments. As a result, the built ML model is able to classify new events, as long as they present a behavior similar to the one observed in the training dataset [7]. On the other hand, networked environments are highly variable, therefore, the building of a realistic training dataset is often not an easily feasible task [3]. In addition, even if a proper training dataset is built, over time, new attacks will occur, and new services will be provided. Hence, the built ML model will become unreliable since the behavior evidenced in the training dataset will no longer be observed in the production environment [1].

In the literature, due to the evolving behavior of network traffic, authors often assume that periodic model updates are performed [3]. However, the model update task is not easily achieved since a new training dataset must be built in each model update. This task requires the collection of new network data, its proper prior labeling (i.e., classifying an event as either normal or attack), and the execution of a computationally expensive process of model training. As a result, the model retraining task may demand several days or even weeks before an updated model is available [1]. This delay in updating the model leaves the system unprotected and can bring significant losses to it. Hence, proposed ML-based NIDS must be able to perform reliable classifications, even in the presence of unknown network traffic behavior [4].

Alternatively, current proposed ML-based NIDS often aim only to improve their accuracy in a given dataset [5]. Consequently, proposed techniques are unreliable for production environments, lowering their accuracies over time, thus, demanding the execution of periodic model retraining. In general, in the literature, to ensure the reliability of ML model classifications, authors often resort to rejection techniques [12]. Rejection-based approaches evaluate the ML model confidence on their classifications before a decision can be made. However, in networked environments, due to the behavior change over time, the ML model confidence values cannot be used reliably [3]. This limitation happens because the confidence values are computed according to the behavior from the training dataset, which due to the environment behavior change over time, does not present the current behavior from production environment [2]. In addition, even if the reliability of classifications can be ensured, the periodic retraining of the model, required by the constant behavior changes in network environments, produces techniques not feasible for proper deployment in the production environment [1] [7] [4].

In this paper it is been proposed a novel reliable stream learning intrusion detection model aiming the reliability of classifications in the presence of unknown traffic behavior over time and the easiness of ML model updates. To achieve such a goal, our proposal is twofold. First, our proposal addresses the challenge of providing reliable classifications of unknown traffic behavior by leveraging a stream learning outlier detection mechanism, which ensures that only known classifications are per-

formed. Second, we perform reliable intrusion detection through stream learning classifiers, coped with our proposed outlier detection scheme, to easiness the model update task by incrementally updating the underlying stream learning models. As a result, our proposal is able to guarantee the reliability of the classifications, even when unknown network traffic is classified, incrementally updating the underlying flow learning model. Therefore, reliably incorporating the new behavior in our system. In summary, this article presents the following contributions: In summary, this paper presents the following contributions:

- An anomaly-based stream learning technique that ensures the reliability of classifications over time. Our proposed scheme is able to assess the classifications performed by the underlying ML model, and is able to incorporate new network traffic behavior into it;
- A reliable stream learning intrusion detection model able to reliably incorporate new network traffic behavior, easing the model update task. Our proposed scheme is able to perform reliable classifications, even for unknown traffic behavior, while also easier the inclusion of new network traffic behavior over time.

2 Preliminaries

2.1 Network-based Intrusion Detection

A network-based intrusion detection system (NIDS) aims at finding malicious activities in a networked environment [4]. Over the last years, several detection approaches were proposed to fulfill such task, in which machine learning (ML) techniques have yield promising reported results [1]. To achieve such a goal, a ML-based NIDS is typically composed of four sequential modules, namely *Data Acquisition*, *Feature Extraction*, *Classification* and *Alert*. First, the *Data Acquisition* module gathers network data from the environment, e.g., network packets. Then, the *Feature Extraction* module extracts a set of features to compound a behavioral description of the event, namely feature vector. In general, in NIDS, the event behavior is represented as a network flow, which comprises the exchanged data between network entities in a given time-window. Through the extracted feature vector, the *Classification* module applies a ML model to classify the input as either normal or malicious. Finally, if a malicious event is found, the *Alert* module reports it.

Several ML-based approaches were proposed for the classification task, in which pattern recognition techniques have yield promising reported results [7]. In pattern recognition, a ML model is built according to the behavior extracted from a training dataset. Consequently, the training dataset must comprise the expected production environment behavior. However, the building of a proper training dataset in NIDS is not easily achieved. This is because the network traffic behavior is highly variable, while also changes over time [3]. As a result, even if a "*perfect*" training dataset is built, it would fail in considering the network traffic as static. Therefore, pattern

recognition techniques for NIDS demand periodic model updates, which is also not easily feasible in production environments.

2.2 Stream Learning for Network-based Intrusion Detection

Over the last years, several stream learning techniques have been proposed for environments in which the behavior changes over time [6]. A stream learning approach differs from traditional pattern recognition techniques by allowing the underlying model to be incrementally updated over time. As a result, the model update task is significantly improved, as the current model is not discarded [3]. On the other hand, new behaviors can be incorporated into the model incrementally, significantly decreasing the computational and time requirements.

However, traditional stream learning techniques assume a supervised scenario, in which the event label is available [6]. In other words, it requires that the current event label is known in order to incorporate the new behavior into the underlying model [10]. On the other hand, in NIDS, the appropriate event label is not always available. However, the event label usually requires specialized assistance. As a consequence, only a small subset of properly labeled events can be provided [1]. Therefore, the applicability of stream learning techniques in NIDS is still in its beginnings.

3 Related Works

Over the last years, several highly accurate ML-based approaches were proposed for NIDS [1]. In general, ML in NIDS is performed through pattern recognition techniques [4]. For instance, Amjad M. *et al.* [8] proposes the usage of several classifiers, each with an adapting class threshold for dealing with network traffic. In their work, the authors have shown that due to the variability of network traffic, the used ML models must be able to be optimized accordingly. However, the authors have not addressed the model update task, neither the reliability of classifications. In contrast, Pradeep S. *et al.* [9] builds a Random Forest classifier coped with a k-means clustering algorithm to identify new attacks. In their work, the authors apply the clustering approach to identify new behaviors, while training the Random Forest classifier in order to detect them. They do not address the challenge of model updates nor the classification reliability when new network traffic behavior is detected. Alternatively, some authors resort to clustering-based approaches to identify new behaviors. For instance, Kai Peng *et al.* [11] applies a clustering-based technique coped with a feature reduction approach to classify unknown attacks. However, their approach can be applied in an unsupervised setting, i.e., with the prior knowledge of the correct event label, their technique is computationally expensive, and cannot be applied in real-time.

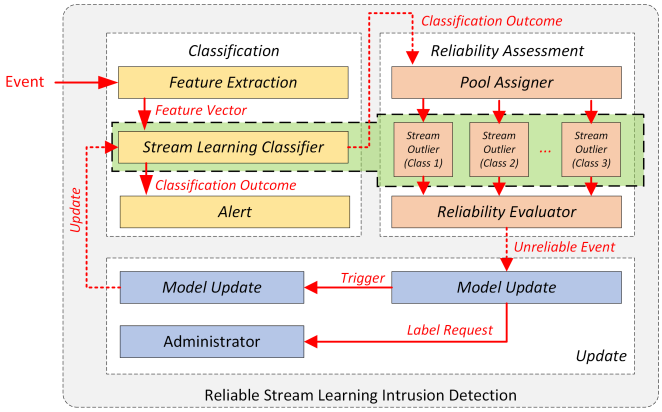


Fig. 1: Reliable Stream Learning Intrusion Detection architecture.

To overcome the challenge of applying ML techniques in environments with evolving behavior, in general, authors resort to stream learning techniques [3] [6]. Such approaches have been extensively used in other fields, such as financial, network communication, and even gaming [6]. However, their applicability in NIDS is still in its beginnings [4]. For instance, a prior work [14] applies a Hoeffding Tree classifier for NIDS. In their work, it is assumed that the event label is always known and can be used to update the model as desired. In contrast, Asmah M. *et al.* [13] proposes a distributed stream learning framework for intrusion detection. Similarly, the authors assume that the event label is previously known, and model updates can be performed as needed.

To the best of our knowledge, our work is the first one to address both model update task and reliability of classifications in a realistic setting. In other words, we assume that only a subset of event labels can be provided, and the model must remain reliable regardless of their update.

4 A Reliable Stream Learning Intrusion Detection Model

In this section, we propose a novel reliable stream learning intrusion detection model to address the challenge of classifying unknown network traffic behavior. It is composed of three main components, namely *Classification*, *Reliability Assessment*, and *Update* (as shown in Figure 1). It focuses on assessing the reliability of classifications over time, even when unknown network behavior is faced. In addition, it facilitates the procedure of updating the model, through flow learning techniques.

The *Classification* module aims at performing the proper event classification. To achieve this goal, our proposal uses flow learning classifiers to activate the incremental update of the model, facilitating the process of updating the model. The use

of the classification outcome is twofold. First, it enables the proper alert, if a malicious event is found. Second, it forwards its output to the *Reliability Assessment* module, which enables the evaluation of the classification reliability.

The *Reliability Assessment* module in turn, evaluates the classification reliability through a pool of stream outliers detectors. Each detector is incrementally updated, according to its related class. For instance, there is one outlier detector for normal class and another for attack class. The goal of each outlier flow detector is to allow an adequate evaluation of the classifications, also in an incremental and updated way. Thus, it is updated incrementally together with the flow learning classifiers. Each outlier detector enables to proper assess whether the current classified event presents a similar behavior to those used for training. If an unknown event is found (an outlier for a given classified event class), it is forwarded to the *Update* module.

Finally, the *Update* module, in turn, takes advantage of events classified in an unreliable way to adapt incrementally to our model, significantly facilitating the model update procedure. The module requests an appropriate label for unreliable events from an administrator, who in turn can manually evaluate the event or use known security tools to properly label it. As the label event is known, the module incrementally updates the flow learning classifier and the related flow outlier detector, keeping the system reliable as new network behaviors are faced over time. The next subsections further describe each of these three stages.

4.1 Classification

In ML-based NIDS, to classify the network activity it is necessary to apply a ML model using a feature vector, which describes the event behavior. However, currently used techniques are unable to cope with unknown network traffic behavior, considering the production environment network traffic. As a consequence, proposed techniques demand constant model updates.

To tackle the model update challenge, our proposal resort to stream learning classifiers, which enables the easiness of the model update process by allowing to incorporate new behaviors into the current ML model. The classification process starts with a to-be-classified network event, which is forwarded to the feature extraction module to generate a feature vector to be classified into a class. The classified event is used twofold. First, if an malicious event is found, it generates a related alert. Second, it is forwarded to the *Reliability Assessment* module, which will evaluate the classification reliability, to define if a model update process is needed or not.

Therefore, by relying on stream learning classifiers, our model is able to significantly improve the update process. This is because we are able to incrementally update the ML model, instead of executing the whole computationally expensive process of model retraining task. As a result, our model is able to significantly ease the process of providing updated classification models.

4.2 Reliability Assessment

Regardless of the current environment behavior, the underlying ML model will perform a decision. As a result, as soon as the environment behavior changes, the classifications become unreliable, increasing the error rates and resulting in alarms that the operator no longer trusts. Therefore, the goal of the *Reliability Assessment* module is twofold. First, it ensures that the performed classification is reliable. In other words, it evaluates whether the classification was made over a known behavior, or it is a new unreliable behavior. Second, it defines if the current event should be used by our scheme to incrementally adapt it. Thus, enabling our model to reliably incorporate a new behavior.

The module receives as input the classifications performed by the *Classification* module. Then, the pool assigner module checks the assigned event label, in order to properly forward it to the corresponding stream outlier detector. The module holds a set of streams outliers detectors, in which each is trained for each considered label, i.e., one for normal and one for attack. As a consequence, each stream outlier detector is able to evaluate the event outlier degree against its corresponding and known behavior. Therefore, the module is able to assign whether the event is known (used for training purposes) or unknown. Finally, according to the outlier degree, the module defines if an event is reliable or unreliable.

Unreliable events can be used for update purposes by the *Update* module, and have corresponding alert suppressed. Therefore, maintaining the system reliability by the operator, who believe that only reliable events generate alerts over time. This process enables to choose the events that should be used for update purposes, instead of requiring a complete new training process.

4.3 Update

Traffic behavior in networked environments changes over time, requiring built ML models to be updated accordingly. Model updates may take days or even weeks to be completed since building a new training dataset is not easily feasible.

Therefore, the goal of the *Update* module is to update the proposed scheme incrementally and reliably. To achieve such a goal, the module receives as input the unreliable events, as established by the *Reliability Assessment* module. Then, the unreliable event correct label is requested to an administrator. The administrator may either manually inspect the event or use well-known tools for the labeling task. The properly labeled event is then used for triggering the model update, which incrementally updates both the stream learning classifier (*Classification*, Section 4.1) and the stream outlier detectors (*Reliability Assessment*, Section 4.2).

Consequently, the update procedure is able to ensure that only properly labeled events are incorporated into the system knowledge. Therefore, our model is able to incrementally and reliably adapt to the environment behavior changes over time.

4.4 Discussion

The insights of the proposed reliable flow learning intrusion detection model is to take advantage of flow learning techniques to facilitate the update process. We can measure whether the current behavior of the environment is known by our underlying ML model. This allows our model to assess the reliability of the classification, considering that unknown behaviors are potentially classification errors. In addition, by using flow learning techniques, we can incorporate unknown behaviors found in our model, significantly facilitating the model update process in a reliable manner. As a result, our model is able to adapt to changes in the behavior of the production environment, maintaining its reliability for the system administrator.

5 Evaluation

The present evaluation focuses on answering three research questions: ($Q1$) *Is the proposed assessment technique able to detect unreliable classifications?* ($Q2$) *Does the proposed reliability assessment technique aids at improving the system accuracy?* ($Q3$) *Is the proposed update technique able to incorporate unknown behaviors into the underlying model?*

The next subsections describe the data set used, the model building process and how it performs on the dataset.

5.1 A Fine-Grained Intrusion Dataset

This paper is based on the *Fine-grained Intrusion Dataset* (FGD) [7]. The data set comprises a series of network behaviors (flows) that can appear in the network variations that ML models must process in production environments. The properties are defined regardless of the inherent changes in network traffic behavior over time. In other words, the FGD dataset comprises the natural variability of network traffic, due to the design limitations of the intrusion datasets. This allows for the proper assessment of ML models according to each possible variation of network behavior in the production environment, including service content, type of service and attack. The dataset contains events in three situations: *known* (behaviors used in training), *similar* (behavior similar to training data), and *new* (not available during the training time; thus, with a different behavior). The similarity must be defined according to the network administrator's discretion, considering that she operates the network and comprehends the possible traffic variations. To provide such fine-grained data control, FGD was created in a local environment, through a Honeypot and well-known attack traffic generation tools, that is further described in [7]. Moreover, five services were used to generate normal traffic. Importantly, the client's and attacker's behaviors significantly vary during the network environment monitoring period [7].

The built dataset enables the fine-grained evaluation of proposed ML-baed NIDS detection schemes with respect to their reliability when facing production environment properties. The testbed was executed for 10 hours, in which 100 nodes were used as clients to generate the benign traffic, while 10 nodes were used as malicious to generate attack traffic. During the testbed execution, the current scenario behavior was varied from *known*, *similar*, and *new* in a 30-minute window interval, as further described in [7]. At total, the dataset comprises 165 GB of data, with 560 million network packets, in which 740 thousand of them are attacks. The feature extraction algorithm grouped events in intervals of 2-seconds while extracting 49 flow-based features from Viegas *et al.* [7] work.

5.2 Model Building

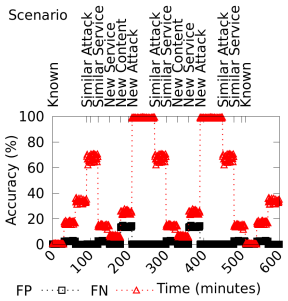
Due to the imbalanced nature of the dataset (only approximately 2% of network flows are classified as attacks), a random undersampling without replacement was performed in the training data, yielding an even distribution between the classes. To properly evaluate the network traffic behavior impact on ML models, the classifiers were trained through the *Known* behavior, while being evaluated throughout the remaining of FGD dataset. For the evaluation of our proposed model we have used the well-known Hoeffding Tree stream learning classifier in the *Classification* module. In contrast, the *Reliability Assessment* module holds two stream outlier detectors, one for normal class and one for attack class. The stream outlier detector was implemented through the streaming half-space tree one-class classifier [16], with 25 trees, a max depth of 15, and an anomaly threshold of 0.5. The stream learning classifiers were implemented using the MOA API [15]. Each classifier was evaluated for their false-negative (FN) and false-positive (FP) rates. The stream learning algorithms were trained only with the data from the known scenario, while being evaluated throughout the whole FGD dataset, with or without the *Update* module.

5.3 Reliability Assessment

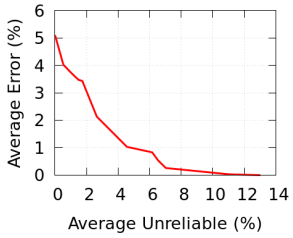
We first evaluate the traditional approach accuracy behavior in FGD, using the same stream learning classifier, Hoeffding Tree, with the same set of parameters throughout the dataset. In the Figure 2a is possible to note a significant increase in both FP and FN rates when classifying *similar* and *new* behaviors.

The first proposal evaluation aims at answering Question $Q1$ and evaluates if the proposed *Reliability Assessment* technique is able to detect unreliable classifications, i.e., detect misclassifications. For that, we select an operation point of unreliable threshold which reaches 0.5% of error rate in the *known* and *similar* scenarios.

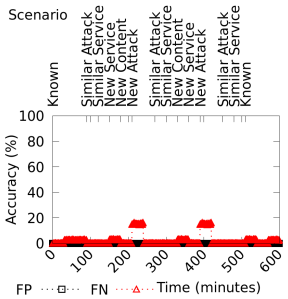
Figure 2b shows the relation between unreliable classifications and error rate, when only reliable classifications are considered. Using the selected operation point



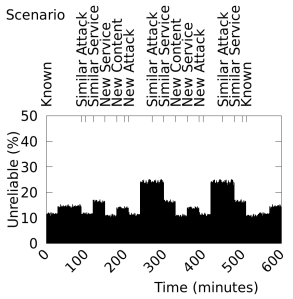
(a) Traditional approach accuracy over time without our proposal.



(b) Unreliable decision curve on *known* and *similar* scenario.



(c) Proposal accuracy over time over reliable classifications.

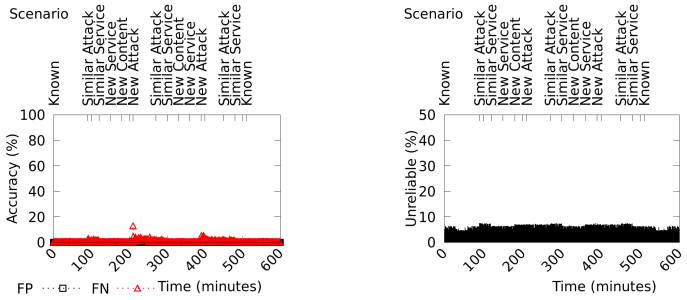


(d) Proposal unreliable classifications over time.

Fig. 2: Accuracy behavior with and without our model without incremental model updates.

(0.5% of error rate, reached at 10% of unreliable classifications), we apply our proposed *Reliability Assessment* technique throughout the whole FGD dataset without incremental model updates. Figure 2d shows the unreliable rating rate over time, when the selected operating point is used in FGD. Our model can adequately guarantee the model's reliability, considering that its unreliable classification rate increases when the behavior of unknown traffic is encountered.

To answer question $Q2$, we evaluate the obtained error rate, when only reliable classifications are considered. Figure 2c shows the related error rate, without incremental model updates, when only reliable classifications are considered for the accuracy computation. It is possible to note that our proposed model is able to maintain the system reliability (accuracy rate), even when facing unknown traffic behavior, despite a higher unreliable classifications rate.



(a) Accuracy over time with incremental model updates.

(b) Unreliable classifications over time.

Fig. 3: Accuracy behavior of our model with incremental model updates over unreliable classifications.

5.4 Stream Learning

Finally, to answer question $Q3$, we incrementally update our model according to the unreliable instances. Figure 3a shows the obtained accuracy, while figure 3b shows the unreliable rate when incremental model updates are performed. It is possible to note that our proposed model was able to incorporate the unknown behavior into the underlying model. Nonetheless, when incremental updates of the model are performed, it was possible to consider other instances as reliable, showing that our model is capable of facilitating the process of updating the model.

6 Conclusion

Current ML-based approaches for NIDS are unable to detect new network behaviors without requiring a computationally and time demanding process of model update. The model proposed in this paper addressed the reliability of classifications, when facing new network traffic behavior, while it also facilitated model updates. Our proposed model used flow learning techniques to incrementally incorporate new traffic behaviors into the underlying model. We assess the reliability of the classification using outlier flow detectors.

The proposed model was able to detect the reliability of the classifications, maintaining the accuracy of the system, in addition to facilitating the process of updating the model, taking advantage of the flow learning techniques.

As a future work, we will focus on the scalability of our model in a distributed environment, maintaining the reliability of the system.

Acknowledgments

The authors thank CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for partial financial support (grant 430972/2018-0 and 315322/2018-7) and the FCT through the LASIGE Research Unit (ref. UIDB/00408/2020).

References

1. Sommer, R., Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *Proc. IEEE Symp. Secur. Priv. May, 305–316*
2. Abreu, V., Santin, A.O., Viegas, E.K., Stihler, M. (2017). A multi-domain role activation model. *IEEE Int. Conf. Commun. (ICC) 3–8*.
3. E., Viegas, Santin, A. , Bessan, A., Neves, N. (2019). BigFlow: Real-time and Reliable Anomaly-based Intrusion Detection for High-Speed Networks *Future Generation Computer Systems Volume 93, April 2019, Pages 473-485*
4. Gates, C., Taylor, C. (2007). Challenging the Anomaly Detection Paradigm: A Provocative Discussion *Proc. 2006 Work. New Secur. Paradig. 21–29*
5. Tavallaee, M., Stakhanova, N., Ghorbani, A. A. (2010). Toward credible evaluation of anomaly-based intrusion-detection methods *IEEE Trans. Syst. Man Cybern. 5, 516–524*
6. He, H., Chen, S., Li, K., Xu, X. (2011). Incremental learning from stream data *IEEE Trans. Neural Netw. 22, 12, 1901–14*
7. E., Viegas, Santin, A., Oliveira L. (2017). Toward a reliable anomaly-based intrusion detection in real-world environments *Comput. Networks. 127*
8. Al Tobi, A.M., Duncan, I.: (2019). Improving intrusion detection model prediction by threshold adaptation *Information, 10, 1-42*
9. Singh, P., Venkatesan, M.: (2018). Hybrid Approach for Intrusion Detection System. *Proc. 2018 Int. Conf. Curr. Trends Towar. Converging Technol. ICCTCT 2018. 1–5*
10. Viegas, E., Santin, A.O., Abreu V., Oliveira, L.S.. (2018) *Enabling Anomaly-based Intrusion Detection Through Model Generalization. IEEE Symposium on Computers and Communications (ISCC) Pages 934-939*
11. Peng, K., Leung, V., Huang, Q.: (2018). Clustering Approach Based on Mini Batch Kmeans for Intrusion Detection System over Big Data *IEEE Access. 6, 11897–11906*
12. Vicentini, C., Santin, A., Viegas, E., Abreu, V. (2019) SDN-based and multitenant-aware resource provisioning mechanism for cloud-based big data streaming. *J. Netw. Comput. Appl. 126, Pages 133–149*
13. Muallem, A., Shetty, S., Hong, L., Pan, J. (2019). TDDEHT: Threat Detection Using Distributed Ensembles of Hoeffding Trees on Streaming Cyber Datasets *Proc. - IEEE Mil. Commun. Conf. MILCOM, 219–224*
14. Viegas, E., Santin, A., Neves, N., Bessani, A., Abreu, V. (2017). A Resilient Stream Learning Intrusion Detection Mechanism for Real-time Analysis of Network Traffic. *IEEE Glob. Telecommun. Conf. GLOBECOM, p. 978-983*.
15. MOA. Available online: https://moa.cms.waikato.ac.nz/ *Accessed 10 december 2019*
16. S. C. Tan, K. M. Ting, T. F. Liu (2011). Fast anomaly detection for streaming data *IJCAI International Joint Conference on Artificial Intelligence, vol. 22, no. 1, pp. 1511–1516*