A Stream Learning Intrusion Detection System for Concept Drifting Network Traffic

Pedro Horchulhack*, Eduardo K. Viegas†, Martin Andreoni Lopez†
*Pontifícia Universidade Católica do Paraná (PUCPR) — Graduate Program in Computer Science (PPGIa)
Curitiba, Paraná, 80215-901, Brazil.
pedro.horchulhack@ppgia.pucpr.br
†Secure Systems Research Center, Technology Innovation Institute, Abu Dhabi 9639, United Arab Emirates.
{*eduardo, martin*}*@ssrc.tii.ae*

2022 6th Cyber Security in Networking Conference (CSNet) | 979-8-3503-9722-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CSNET56116.2022.9955620

***Abstract*—Network-based intrusion detection is a widely explored topic in the literature. Yet, despite the promising reported results, designed schemes are rarely used in production environments. Apart from evolving as time passes, the behavior of network traffic varies significantly, rendering proposed schemes unreliable for real-world application. This paper proposes a new stream learning intrusion detection aiming for feasible model updates, implemented in three phases. First, intrusion detection is performed through a stream learning classifier, enabling incremental model updates to be performed. Second, new network traffic behavior is identified through a one-class learner. Third, identified new network traffic is incrementally incorporated into our system. Experiments performed on a dataset containing evolving network traffic behavior have shown our proposal feasibility, reaching up to 96% of accuracy while demanding only 48% of labeled events to be provided.**

***Index Terms*—Intrusion Detection, Stream Learning, Concept Drift, Machine Learning.**

I. Introduction

Over the past years, the number of reported cyberattacks are increasing and is still on the rise, showing that current security solutions have been ineffective to protect computational systems. According to a Kaspersky report, only in 2021, more than 15% of all Internet users were targeted by a cyberattack, representing a 50% increase compared to its precedent year [1]. Consequently, the provision of security solutions able to detect this growing number of network threats becomes a must. In practice, network operators resort to network-based intrusion detection systems (NIDS) to detect network attacks, typically implemented through two main approaches [2]. On one hand, *misuse-based* techniques signal network attacks based on previously known attack patterns, performing the detection task through a pattern matching approach. Nevertheless, this method can only detect previously known threats, leaving systems unprotected against new kinds of attacks [3]. On the other hand, *behavior-based* techniques perform the detection task according to the behavior of the analyzed event, signaling misconduct according to a previously modeled behavior. Therefore, *behavior-based* approaches can detect new (zero-day) attacks, assuming that these new attacks behave similarly to previously modeled ones.

Due to the ever-increasing number of network-based attacks, several works have been proposed for *behavior-based* detection of network attacks [4, 3]. Most proposed approaches perform the detection task through machine learning (ML) techniques, typically using pattern recognition approaches. To conduct such a procedure, an ML model is built through a computationally demanding model training task. In such a case, the behavior of a training dataset, composed of huge amounts of labeled production environment samples, is evaluated. Finally, the performance of the built ML model is estimated through a test dataset, which is then assumed to be evidenced when the system is deployed in a production environment.

The behavior of networked environments, unfortunately, is highly variable as well as evolves as time passes [5]. As time goes on, new network attacks can be identified, and new network services can be provided. The non-stationary network traffic behavior poses a great challenge to current *behavior-based* approaches proposed in the literature. The reliability of designed ML techniques is bound to the behavior used during the training phase [6]. As a result, changes in network traffic demand the provision of a new ML model, which can only be obtained after the collection of new network traffic, its labeling, and the execution of the model training task [5]. Therefore, model updates pose a high cost, and may demand several days or even weeks to be conducted, leaving systems unprotected while an updated model is not yet available. Surprisingly, the non-stationary behavior of network traffic is often neglected in the literature, wherein authors assume that model updates can be executed as needed, despite the challenges related to the data collection and labeling. Consequently, despite the promising reported results in the literature, *behavior-based* intrusion detection remains mostly a research topic, rarely being deployed in production environments.

A popular approach used in the literature to address environments with non-stationary behaviors' relies on stream learning techniques [7]. In contrast to traditional pattern recognition approaches, stream learning enables incremental model updates to be performed as time passes, significantly decreasing the computational costs associated with the model-building task. Opposite to traditional approaches, the current model is not discarded during model updates, in the contrary, it is performed for each evaluated event in an incremental manner.

This paper proposes a new stream learning intrusion detec-

tion model for concept drifting network traffic, implemented through three phases. First, intrusion detection is performed through a stream learning detector, able to incorporate new network traffic behavior as time passes. Second, new network traffic behavior is identified by making use of a one-class stream learning algorithm. The algorithm is trained with instances used during the training phase, thus, signaling unknown behaviors that should be used for model updates. Therefore, it enables operators to proactively identify when model updates must be performed, as well as which network events are unknown to our deployed scheme. Third, identified new network traffic is incrementally incorporated by our model, decreasing the computational costs of model updates.

In summary, our paper's main contributions are as follows:

- A new stream learning intrusion detection model for concept drifting network traffic. The proposed model can incorporate new network traffic behavior's into the deployed model.
- A model for detection of new network traffic that must be incorporated into the intrusion detection model in unsupervised settings.

The remainder of this paper is organized as follows. Section II introduces the background related to our paper. Section III presents related works on intrusion detection. Section IV describes our proposed model, and Section V evaluates our scheme. Finally, Section VI concludes our work.

II. Network-based Intrusion Detection and Stream Learning

Network-based intrusion detection systems (NIDS) are commonly implemented through four sequential modules [8], namely *Data Acquisition*, *Feature Extraction*, *Classification*, and *Alert*. The *Data Acquisition* module collects the data from the monitored environment, typically reading the passing network packets from a network interface card. The collected data is fed as input to a *Feature Extraction* module, which performs the associated data preprocessing and the extraction of the proper behavioral features. Usually, in NIDS, network traffic is classified according to the network flow, which depicts the behavior of network traffic in a given time interval. For instance, the number of sent/received network packets between two given hosts over the last 15 seconds. The extracted set of features is classified by the *Classification* module, which establishes the network flow label as either *normal* or *attack*, signaling identified threats through the *Alert* module.

Several highly accurate machine learning (ML) techniques have been proposed for the network flow classification task [2]. Prior work generally resorts to pattern recognition approaches, yielding highly accurate intrusion detection accuracies. To achieve such a goal, researchers use three distinct datasets, namely *training*, *validation*, and *test*. The first is used for extracting the behavioral ML model, thus, must be composed of a representative number of network samples. The *validation* dataset is often used during the model building phase, enabling the model finetuning to be executed, such as the selection of features and the model hyperparameters [9]. Finally, after the proper model building, the *test* dataset is used to validate the final model accuracy. Consequently, the accuracy measured through the test dataset is assumed to be evidenced when the model is used in production environments.

Despite the benefits achieved through ML in several fields, such as image classification [10], performance monitoring [11], and fault detection, their actual deployment in the intrusion detection field remains a challenging task. The ML methods are mostly used as a research field, rarely deployed in real-world applications [12].

The behavior of networked environments is non-static due to the evolving characteristics of network traffic [13]. As a result, the accuracy rates measured through the *test* dataset are only reliably evidenced in production if the network traffic behavior remains unchanged. In contrast, network traffic changes daily, a situation caused due to the discovery of new attacks, the provision of new services, or even changes in the network traffic communication link.

These naturally occurring changes in network traffic behavior demand the execution of frequent and time-consuming model updates [5]. To achieve such a goal, a new training and testing dataset must be collected and adequately labeled, often only achieved through expert assistance. Notwithstanding, the model update must be executed from scratch, as traditional pattern recognition discards the outdated model during the model training phase. As a result, model updates are neither cheap nor easily feasible in network-based intrusion detection, in the contrary, their execution often poses a high cost, while also demanding several days or even weeks before an updated model is available, leaving systems unprotected during such a period [14]. Surprisingly, the literature still neglects such a challenge, assuming that model updates can be easily executed as time passes.

A suitable approach used in scenarios wherein the environment behavior may change as time passes resorts to stream learning techniques. Contrary to traditional ML approaches, stream learning aims to enable incremental model updates to be performed as time passes, incorporating new behaviors into a previously existing model. More specifically, stream learning deals with a potentially unbounded infinite sequence of data items that must be processed over time. The incoming data items must be processed (i.e. classified) while considering potential stream changes with very costly data labeling. Notwithstanding, due to the potentially infinite nature of processed streaming, designed algorithms must be able to be incrementally updated without requiring the data storing, while also considering a resource-constrained scenario in terms of processing and memory.

Stream Learning differs from conventional batch processing in: i) the data elements in the stream arrive online; ii) the system has no control over the order in which the data elements arrive to be processed; iii) stream data are potentially unlimited in size; iv) once an element of a data stream has been processed, it is discarded or archived and cannot be easily retrieved unless it is explicitly stored in memory, which is usually small relative to the size of the data streams. Further,

the latency of stream processing is better than micro-batch, since messages are processed immediately after arrival [15].

A stream ψ is an unbounded set of data, $\psi = \{D_t | 0 \leq t\}$ where a point D_t is a set of attributes with a explicit or implicit time stamp. Formally one data point is $D_t = (\mathbf{V}, \tau_t)$, where $\mathbf{V}$ is a p-tuple, in which each value corresponds to an attribute, and τ_t is the time stamp for the t-th sample.

Hoeffding Tree introduces the decision tree rationale in stream learning settings. The algorithm takes advantage of the Hoeffding Bound to enable it to learn stream patterns without storing the input data samples for future processing. Thus, it provides the same asymptotically guarantees when compared with traditional batch learners, as explores the statistical distribution of samples, significantly decreasing the memory needs.

Stream learning algorithms can provide benefits that can address the model update task in intrusion detection. Apart from significantly decreasing the computational costs of model updates, stream learning algorithms must be able to address concept drift, a situation wherein the behavior of the processed environment changes.

III. Related Works

Intrusion detection through machine learning (ML) techniques has been a subject of a plethora of works in the literature. Yet, despite the promising reported results, IDS are rarely deployed in production environments, as the network traffic behavior changes as time passes, and prior works neglect model update tasks [16].

For instance, N. Moustafa *et al.* [17] proposes an ensemble-based intrusion detection model for the internet of things aiming for higher accuracy rates. Their proposed scheme makes use of new flow-based features for network traffic classification and achieves higher accuracies when compared to other approaches. Similarly, the authors assume that model updates will be performed periodically, despite the challenges it introduces. Recently, W. Lunardi *et al.* [18] proposed an unsupervised anomaly-based deep learning technique for network-based intrusion detection. Their proposed model was able to improve detection accuracy when compared to traditional techniques in a normal-only training setting through a predefined threshold to identify anomalous network traffic. However, their proposed scheme overlooks model updates, assuming that only attack-related network traffic is subject to changes as time passes. Y. Zhou *et al.* [19] addressed intrusion detection through an ensemble-based approach coped with a feature selection scheme. Their proposal was able to significantly improve classification accuracy when compared to the traditional approach, however, overlooked changes in network traffic.

The evolving nature of network traffic is hardly considered in the literature. E. Viegas *et al.* [9] addressed network traffic changes by aiming the building of machine learning models with a higher lifespan. To achieve such a goal, the authors perform a multi-objective feature selection that introduces a measure for model longevity. Similarly, R. dos Santos *et al.* [20] aimed higher model lifespan considering the classification reliability during model training in a reinforcement learning setting. Both approaches can decrease the frequency in which model updates are necessary, however, does not address how such a task can be eased when performed.

Stream learning techniques for intrusion detection task is not widely evidenced in the literature, despite the benefits it introduces to proposed schemes. P. Horchulhack *et al.* [5] uses stream learning algorithms to perform incremental model updates in intrusion detection. The authors incrementally adapt the stream learning algorithm based on the confidence score of the used approaches. However, if the confidence score of used stream learners is biased, the need for model updates will not be identified. N. Martindale *et al.* [21] evaluates an ensemble-based approach using a variety of stream learning algorithms. The authors' approach addressed intrusion detection by incrementally adapting their used model, however, the identification of new network traffic was not addressed.

Andreoni Lopez *et al.* [22, 23] use the stream learning platform for the intrusion detection system. The system uses Apache Spark while running a traditional decision tree algorithm. The proposal can detect concept drift on network traffic. When a concept drift is detected, the model is updated incrementally. Nevertheless, the proposal could not perform model updates in an unsupervised manner.

It is possible to note that intrusion detection tasks through machine learning techniques have been widely explored in the literature [2, 18], yet, their use in production environments remains low. Most proposed schemes overlook the challenges related to network traffic behavior changes, whilst assuming that model updates will be performed periodically [3].

IV. A Stream Learning Intrusion Detection System for Concept Drifting Network Traffic

In light of this, we propose a new stream learning intrusion detection system for concept drifting network traffic implemented in two phases, namely *Classification Pipeline* and *Update Pipeline*.

First, the *Classification Pipeline* addresses intrusion detection as a stream learning task, enabling model updates to be performed incrementally. Our main insight is to employ stream learning algorithms to easiness the model update task in face of new network traffic behavior. As a result, when new network traffic is identified, its behavior can be adequately incorporated by our proposed scheme.

Second, we address model updates by using a one-class stream learning scheme. Our proposal contribution is to identify new network traffic, that must be used for model updates, without the assistance of an expert. To achieve such a goal, we build a one-class stream learning algorithm based on the network events previously used for the training phase. Our main insight is to use the one-class algorithm to identify network behaviors not previously used for training, hence, new network flows must be incorporated into our scheme. Consequently, our proposed scheme can significantly decrease expert intervention to address new network traffic. Finally, new

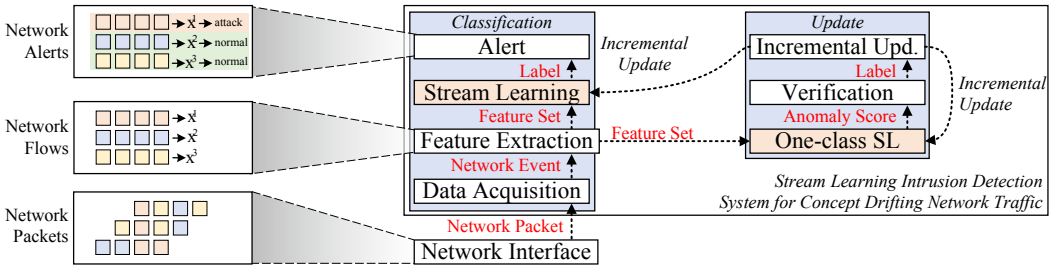


Fig. 1: Overview of our proposed stream learning intrusion detection model for concept drifting network traffic. Classification is performed through a stream learning scheme, while new network traffic is identified by using a one-class stream learner.

network traffic behavior is incrementally incorporated by our model.

The next subsections further describe our proposed model, shown in Figure 1, including the modules that implement it.

A. Classification Pipeline

Our proposed model takes into account a traditional network-based intrusion detection scheme implemented by making use of a *behavior-based* detection approach. Contrary to most prior works, our scheme performs the network traffic classification through a stream learning model.

The classification starts with a to-be-classified network event collected by a *Data Acquisition* module (see Fig. 1), e.g. a network packet. The collected event is used as input by a *Feature Extraction* module which extracts a set of behavioral flow features compounding a feature set, used twofold in our approach. On one hand, the extracted behavioral features are fed to a stream learning classifier, which outputs a corresponding classification outcome. The classification triggers alerts if a network attack is identified, signaling to the operator the identified network intrusions. On the other hand, the feature set is also fed as input to our proposed *Update Pipeline* to identify newly occurring network traffic in an unsupervised fashion.

B. Update Pipeline

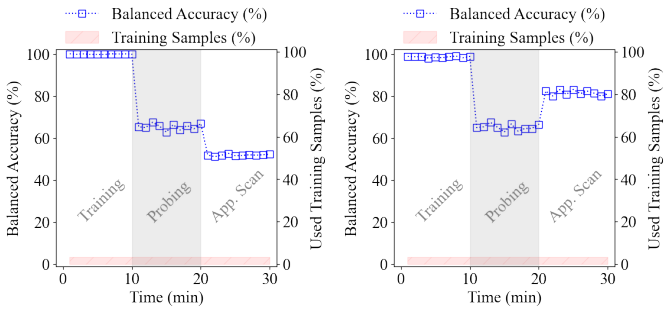
The behavior of network traffic changes as time passes, demanding model updates to be performed periodically. However, model updates pose significant challenges, mainly due to the identification of behavior changes in network traffic, as well as the labeling task of the newly occurring events, often only achievable by expert assistance.

To address such a challenge, our proposed model makes use of a one-class stream learning model to proactively identify network events that are unknown to the stream learning classifier. The one-class model is trained based on the events used for stream learning model training, therefore, it identifies as anomalies network behaviors not similar to those used for training purposes. The newly occurring network traffic (anomalies) is then used for incremental model updates over the current stream learning classifier and the used one-class stream learner. The update pipeline procedure is shown in Figure 1 and is executed for every new event that our scheme will classify. The algorithm applies our proposed one-class detector to identify new network traffic. Signaled events are fed as input to a *Verification* module that adequately labels the new network traffic, e.g. through human assistance. Finally, the labeled event is fed as input to the *Incremental Update* module, which performs the model update on both the stream learning classifier and our one-class stream learner.

Algorithm 1 Proposal classification and update scheme.

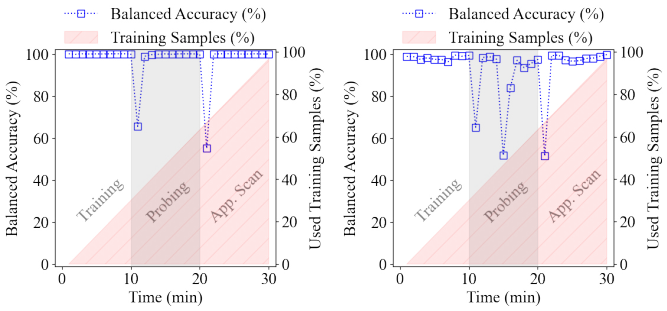
Require:
- Learner $l \leftarrow f(x) = y$ ▷ Stream Learner
- One-class Learner $o \leftarrow f(x) = v*$ ▷ One-class
- Anomaly Threshold $t \in [0, 1]$, typically $t = 0.1$
- Stream $S \leftarrow \{event_1, \ldots, event_\infty\}$
- Start in initial state $s \in S$
- **while** s is not final **do**
  - FeatureVector x = extractFeatures(s)
  - Classification c = predict(l, x)
  - **if** c is $attack$ **then**
    - $alert(e)$
  - **end if**
  - **if** $anomalyScore(o, x) \geq t$ **then**
    - Label $label \leftarrow requestLabel(e)$
    - $l \leftarrow incrementalUpdate(l, x, label)$
    - $o \leftarrow incrementalUpdate(o, x)$
  - **end if**
  - $s \leftarrow s'$ ▷ Next network event
- **end while**

Algorithm 1 shows our proposal classification and update procedure. It receives as input a stream learner (l), a one-class learner (o), with an associated anomaly threshold (t), and the stream source (s). The model update is executed for every classified event. The scheme continuously reads an unbound sequence of network events from a given stream (s). The behavior of the event is extracted compounding a feature vector (x), subsequently classified by a given stream learner (l). If the classification outcome (c) is deemed as an attack, it is properly reported. After the event classification, the one-class learner (o) computes an anomaly score and evaluates it with a given anomaly threshold (t). Anomalous events are used for incremental model updates on both our used stream learner and our one-class learner.



(a) Gradient Boosting (b) Hoeffding Tree

Fig. 2: Accuracy performance of selected classifiers on FGD dataset, in a 1-minute periodicity. Classifiers are trained using the initial minute of the dataset in the *known* scenario in a streaming manner. No model updates are performed as time passes.



(a) Gradient Boosting (b) Hoeffding Tree

Fig. 3: Accuracy performance of selected classifiers on FGD dataset, in a 1-minute periodicity. Classifiers are updated every minute using all the data that occurred before the model update procedure.

As a result, our proposed model's main contribution is to enable the selection of which events are unknown to the underlying stream learning model. Thus, the number of events that an expert must label is significantly decreased, given that not all events will be used for model updates, significantly easing the conduction of model updates as time passes.

V. Evaluation

Our proposed model was evaluated considering the following research questions (RQ):

- (***RQ1***) *How do the traditional classifiers performs with no updates?*
- (***RQ2***) *How does the updated classifiers counterpart performs?*
- (***RQ3***) *What is the performance of our proposed model?*

The next subsections further describe our performed evaluations including the model building and used dataset.

A. Model Building

Our evaluation is performed based on the *Fine-grained Intrusion Dataset* (FGD) [24]. It is made of network flows that present the network variation which designed *behavior-based* NIDS will experience in production environments. The dataset contains events in three situations, as follows:

- ***Training***. Network traffic remains unchanged as time passes. Normal traffic depicts 5 different protocols, generated by 100 clients [25]. Attacker traffic depics port-scanning attacks.
- ***Probing***. The behavior of attacker network traffic shifts to application-level scanning. Normal network traffic remains unchanged.
- ***App. Scan***. Attacker generates application-level pentesting. Normal network traffic does not change.

Therefore, during the testbed execution, the current scenario behavior varies from *Training*, *Probing*, and *App. Scan* in a 10-minute window interval each, as further described in [25]. The feature extraction algorithm grouped events in intervals of 2-seconds while extracting 49 flow-based features from Viegas *et al.* [9] work.

The selected classification schemes were evaluated concerning their false-negative rates (FN), false-positive rates (FP), true-positive rates (TP), and true-negative rate (TN), as follows:

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

To evaluate selected techniques concerning the detection of both normal and attack samples we compute the balanced accuracy, as shown in Eq. 1.

$$BalancedAccuracy = \frac{\overbrace{TP/(TP+FN)}^{\text{sensitivity}} + \overbrace{TN/(TN+FP)}^{\text{specificity}}}{2} \quad (1)$$

We evaluate two widely used machine learning and stream learning techniques. The first was implemented through the Gradient Boosting (GBT) classifier. The latter was implemented through the Very Fast Hoeffding Tree (VFHT) stream learner. The GBT was evaluated with 100 decision trees as its base-learner, where each one of them uses $gini$ as the node split quality metric, the classifier relies upon a 0.3 learning rate value, with $deviance$ as the loss function. The VFHT stream learner was evaluated using information gain as the node split criterion, a grace period of 200, and naive Bayes adaptive as the leaf node prediction. Our proposed one-class stream learner was implemented using a Half-Space Tree (HST). The one-class learner is built using all the instances used for the stream learning training procedure. The HST makes use of a window size of 15, 25 estimators, while the anomaly threshold varies for each experiment (Alg. 1, *Anomaly Threshold*) The GBT was implemented through *scikit-learn* API $v0.24$, whereas the VFHT and the HST were implemented on top of *scikit-multiflow* API v. 0.5.3. The parameters of the selected techniques were empirically set.

B. Traditional Behavior-based Intrusion Detection

Our first experiment aims at answering RQ1 and evaluates the classification performance of traditional *behavior-based* intrusion detection techniques without performing updates as time passes. In such a case, the selected classifiers are built using the data that occurred during the first 5 minutes of our dataset (*Training* behavior), and no model updates are performed. Figure 2 shows the classification performance of selected techniques in our dataset. It is possible to note that the classification accuracy remains significantly high during the *training* behavior, decreasing when a new behavior is evidenced (*Probing* and *App. Scan*). Consequently, model updates must address the non-stationary behavior of the network traffic.

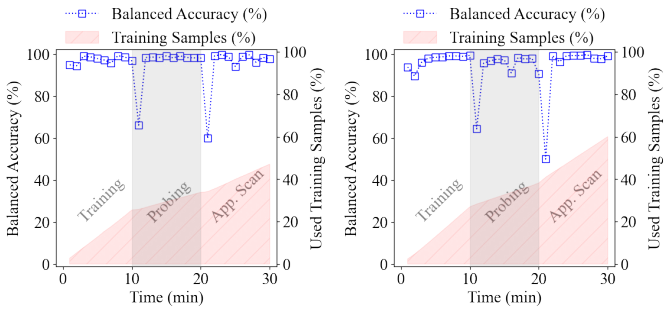
Our second experiment answers RQ2 and evaluates how model updates can be used to address new network traffic behavior. To achieve such a goal, we conduct periodic model updates every minute in our dataset using the network events that occurred over the last 1 minute. More specifically, we perform periodic model updates in a 1 minute periodicity with 1 minute worth of data.

Figure 3 shows the classification performance of the selected techniques when model updates are periodically performed. In contrast to their no-update counterpart, periodically-updated classifiers were able to address the non-stationary behavior of network traffic. It is possible to note that classification accuracies are increased as soon as the model update is performed. For instance, in a *Probing* scenario, the classification accuracy is initially degraded (Fig. 3, 10^{th} minute), while after the model is adequately updated the accuracy increases again (Fig. 3, 11^{th} minute onward). However, conducting such an update procedure in production environments is not easily feasible, due to the huge amounts of network traffic that must be adequately labeled. Consequently, model updates must be performed demanding as few as possible labeled network events.

C. Stream Learning for Concept Drifting Network Traffic

Finally, we evaluate the performance of our proposed scheme to address the evolving behavior of network traffic. To achieve such a goal, we evaluate our scheme while varying the anomaly threshold used by our one-class learner (Alg. 1, *Anomaly Threshold*). The threshold must be defined based on the operator's needs, as a higher value will demand more events to be adequately labeled, but will, as an assumption, provide higher model accuracies. Therefore, we select two operation points (thresholds), namely *High*, and *Avg.* using a 0.85, and 0.45 anomaly thresholds respectively.

The two selected anomaly threshold operation points are used to evaluate the performance of our model. Recalling that we use the mentioned threshold to incrementally update our proposed scheme (see Alg. 1) Figures 4a, and 4b show the classification accuracy of our model according to the used anomaly threshold. Our proposed model was able to improve accuracy while demanding less events to be provided during model updates (Fig. 3b *vs.* 4) For instance, our scheme at a high operation point (Fig 4a) reaches an average accuracy



(a) Performance (High Operation Point)

(b) Performance (Avg. Operation Point)

Fig. 4: Accuracy performance of our proposed model on FGD dataset, in a 1-minute periodicity. Our proposal is updated every minute according to the identified anomalous samples on the preceding minute.

of 95.49%, while demanding only a total of 48% of labeled events to be provided as time passes. Notwithstanding, our proposed scheme can achieve high classification accuracies even when fewer events are provided for model update purposes. For example, the Avg. operation point (Fig. 4b) reaches an average accuracy of 94.46%, while demanding a total of 60% of labeled events to be provided over time.

VI. Conclusion

The behavior of network traffic varies over time, significantly affecting the reliability of designed *behavior-based* intrusion detection schemes. This paper proposed a novel approach for feasible model updates through a stream learning technique coped with a one-class learner. The first can incorporate new network traffic behavior as time passes into the deployed model. The latter detects new network traffic that must be used for model update purposes. The experiments shown in the paper demonstrated our proposal's feasibility. For future works, we plan to evaluate our scheme on more datasets.

References

[1] *Kaspersky Security Bulletin 2022. Statistics*, 2022. [Online]. Available: https://go.kaspersky.com/rs/802-IJN-240/images/KSB_statistics_2021_eng.pdf

[2] 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. on Network and Service Management*, vol. 17, no. 4, pp. 2451–2479, Dec. 2020.

[3] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE Symposium on Security and Privacy*. IEEE, 2010.

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

[5] P. Horchulhack, E. K. Viegas, and A. O. Santin, "Toward feasible machine learning model updates in network-based intrusion detection," *Computer Networks*, vol. 202, p. 108618, Jan. 2022.

[6] J. V. V. Silva, N. R. de Oliveira, D. S. V. Medeiros, M. Andreoni Lopez, and D. M. F. Mattos, "A statistical analysis of intrinsic bias of network security datasets for training machine learning mechanisms," *Annals of Telecommunications*, Feb.

[7] E. Viegas, A. Santin, V. Abreu, and L. S. Oliveira, "Stream learning and anomaly-based intrusion detection in the adversarial settings," in *2017 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, Jul. 2017. [Online]. Available: https://doi.org/10.1109/iscc.2017.8024621

[8] D. Chou and M. Jiang, "A survey on data-driven network intrusion detection," *ACM Comput. Surv.*, vol. 54, no. 9, oct 2021. [Online]. Available: https://doi.org/10.1145/3472753

[9] E. Viegas, A. O. Santin, and V. A. Jr, "Machine learning intrusion detection in big data era: A multi-objective approach for longer model lifespans," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 366–376, Jan. 2021.

[10] J. Mallmann, A. O. Santin, E. K. Viegas, R. R. dos Santos, and J. Geremias, "PPCensor: Architecture for real-time pornography detection in video streaming," *Future Generation Computer Systems*, vol. 112, pp. 945–955, Nov. 2020.

[11] F. Ramos, E. Viegas, A. Santin, P. Horchulhack, R. R. dos Santos, and A. Espindola, "A machine learning model for detection of docker-based APP overbooking on kubernetes," in *IEEE International Conf. on Communications (ICC)*, 2021.

[12] J. Gu and S. Lu, "An effective intrusion detection approach using SVM with naïve bayes feature embedding," *Computers &amp Security*, vol. 103, p. 102158, Apr. 2021. [Online]. Available: https://doi.org/10.1016/j.cose.2020.102158

[13] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 3971–3988. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/arp

[14] R. R. dos Santos, E. K. Viegas, and A. O. Santin, "A reminiscent intrusion detection model based on deep autoencoders and transfer learning," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Dec. 2021. [Online]. Available: https://doi.org/10.1109/globecom46510.2021.9685724

[15] M. Andreoni Lopez, A. G. P. Lobato, and O. C. M. Duarte, "A performance comparison of open-source stream processing platforms," in *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[16] R. R. dos Santos, E. K. Viegas, A. O. Santin, and V. V. Cogo, "Reinforcement learning for intrusion detection: More model longness and fewer updates," *IEEE Transactions on Network and Service Management*, pp. 1–17, 2022. [Online]. Available: https://doi.org/10.1109/tnsm.2022.3207094

[17] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019.

[18] W. T. Lunardi, M. Andreoni Lopez, and J. Giacalone, "Arcade: Adversarially regularized convolutional autoencoder for network anomaly detection," 2022.

[19] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Computer Networks*, vol. 174, p. 107247, Jun. 2020.

[20] R. R. dos Santos, E. K. Viegas, A. Santin, and V. V. Cogo, "A long-lasting reinforcement learning intrusion detection model," in *Advanced Information Networking and Applications*. Springer International Publishing, 2020, pp. 1437–1448.

[21] N. Martindale, M. Ismail, and D. A. Talbert, "Ensemble-based online machine learning algorithms for network intrusion detection systems using streaming data," *Information*, vol. 11, no. 6, p. 315, Jun. 2020.

[22] M. Andreoni Lopez, D. M. Mattos, O. C. M. Duarte, and G. Pujolle, "Toward a monitoring and threat detection system based on stream processing as a virtual network function for big data," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 20, p. e5344, 2019.

[23] A. G. P. Lobato, M. Andreoni Lopez, A. A. Cardenas, O. C. M. Duarte, and G. Pujolle, "A fast and accurate threat detection and prevention architecture using stream processing," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 3, p. e6561, 2022.

[24] R. R. dos Santos, E. K. Viegas, and A. O. Santin, "Improving intrusion detection confidence through a moving target defense strategy," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[25] E. K. Viegas, A. O. Santin, and L. S. Oliveira, "Toward a reliable anomaly-based intrusion detection in real-world environments," *Computer Networks*, vol. 127, pp. 200–216, Nov. 2017.