A Reliable Semi-Supervised Intrusion Detection Model: One Year of Network Traffic Anomalies

Eduardo K. Viegas*, Altair O. Santin*, Vinicius V. Cogo**, Vilmar Abreu*
*Graduate Program in Computer Science - Pontifical Catholic University of Parana, Brazil
{eduardo.viegas, santin, vilmar.abreu}@ppgia.pucpr.br
**LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
vielmo@lasige.di.fc.ul.pt

***Abstract*—Despite the promising results of machine learning for network-based intrusion detection, current techniques are not widely deployed in real-world environments. In general, proposed detection models quickly become obsolete, thus, generating unreliable classifications over time. In this paper, we propose a new reliable model for semi-supervised intrusion detection that uses a verification technique to provide reliable classifications over time, even in the absence of model updates. Additionally, we cope with this verification technique with semi-supervised learning to autonomously update the underlying machine learning models without human assistance. Our experiments consider a full year of real network traffic and demonstrate that our solution maintains the accuracy rate over time without model updates while rejecting only 10.6% of instances on average. Moreover, when autonomous (non-human-assisted) model updates are performed, the average rejection rate drops to just 3.2% without affecting the accuracy of our solution.**

***Index Terms*—Intrusion Detection, Machine Learning, Classification Reliability, Semi-Supervised Co-Learning.**

I. Introduction

In 2019, network-based attacks occurred 84% more frequently [1], while also significantly increased their bandwidth consumption, in some cases reaching hundreds of GB/s [2]. For instance, in 2018, GitHub was a target of a Distributed-Denial-of-Service (DDoS) attack that peaked 1.35 Tb/s [3]. As a consequence, service operators need access to efficient solutions that enable real-time analysis of malicious content from these massive network attacks.

Network-based Intrusion Detection Systems (NIDS) [4] is a commonly employed approach to detect network attacks. Many recent works have proposed highly accurate *behavior-based* intrusion detection schemes, from which most of them rely on pattern recognition approaches [5]. The usual methodology in this type of solution is to build a machine learning (ML) model from an intrusion dataset of expected behaviors in the system and detect intrusion attempts using the modeled behavior extracted from this training dataset [5].

Contrarily, the behavior in real-world network environments changes daily, either due to the integration of new services in the system or the discovery of new attacks [6]. These behavior changes in the network traffic make ML models unreliable and require them to be retrained [7]. The main reason is that the model is obtained analyzing the behavior of a training dataset that no longer represents the current network behavior [8]. As a result, the accuracy rates in production are no longer the same as the ones obtained during the test phase [6].

Retraining the ML model is not an easily achieved task. First, new network events must be collected to compose a new training dataset [4]. Second, these events must be individually and correctly labeled either as normal traffic or an intrusion attempt. In addition, the labeling of network events is usually only achieved through human assistance, which limits the sample to only a subset of the collected events [9]. After obtaining an up-to-date training dataset, one can finally retrain the ML model through a computationally expensive process, measure its accuracy rates, and deploy the new ML model. As a result, a ML model can even become outdated before its actual deployment in production [10]. Therefore, any detection scheme must reliably detect intrusion attempts for long periods, regardless of the model update frequency [7].

Although important, updating ML models is a task still overlooked in the literature [11]. In general, authors assume that periodic model updates will be made in their systems, but they do not specify how it is done nor take into consideration the challenges it brings to ML-based NIDS deployment in production. Therefore, despite the promising reported results, ML-based NIDS remains mostly a research topic rarely deployed in real-world environments [6].

This gap is an opportunity we address in this paper by proposing a novel reliable semi-supervised intrusion detection model able to autonomously withstand long periods without requiring human assistance or model updates. It is based on two main insights. First, an autonomous verification technique assess the reliability of the classification of network traffic over time. It ensures that only highly confident classifications are accepted and, consequently, maintains (over time) the accuracy rates obtained in the testing phase—even without updating the model. Second, we leverage this verification technique to cope with a reliable semi-supervised co-learning to enable autonomous (non-human-assisted) updates of the underlying ML model. In summary, our main contributions are two-fold:

- We evaluate several state-of-the-art ML-based NIDS using a one-year-long real network traffic dataset. Our results attest that current supervised and semi-supervised ML-based approaches do not cope with the inherent changes in the network traffic behavior;

- We propose and evaluate a novel reliable semi-supervised intrusion detection model that withstands long periods without human assistance. It does not require experts labeling unknown network traffic (i.e., traffic new to the ML model) neither requires the ML model to be updated. Nevertheless, it is prepared to update the ML model autonomously, maintaining its accuracy over time.

II. Preliminaries

A. Machine Learning for Network-based Intrusion Detection

The development of highly accurate ML-based NIDS has been an active research topic in recent years [12]. In general, proposed techniques are composed of four sequential modules: *Data Acquisition*, *Feature Extraction*, *Classification*, and *Alert*. First, the *data acquisition* module collects network events to be classified—e.g., network packets. Then, the *feature extraction* module extracts behavioral features from the collected events and builds a feature vector that will be used for classification. For instance, in NIDS, events are often summarized through network flows, which comprises the exchanged data from hosts over the network in a time window. After the feature extraction, the resulting feature vector is forwarded to the *classification* module, which applies a ML model to establish the network flow class—e.g., normal or intrusion. Finally, when an intrusion attempt is detected, the *alert* module triggers the respective alert.

Therefore, the classification task is achieved by applying a ML model to classify a feature vector, which represents the current network behavior. ML models are built using training datasets, which contain a set of network flows comprising the expect network environment behavior from both normal and attack events. Therefore, the training dataset must have been correctly labeled to allow the ML algorithm to infer a behavior from the collected data. However, labeling network events is a challenging task that usually requires human-assistance [9]. Nonetheless, as the classification is achieved through the behavior analysis, the used ML model may misclassify some instances, e.g., when a network attack mimics normal events. Therefore, in the testing phase, the ML model accuracy rates are measured, such as the True Positive (TP) and True Negative (TN) rates. TP denotes the ratio of intrusion attempts correctly classified as attacks, whereas TN denotes the ratio of regular events correctly classified as normal activities.

B. Semi-supervised Machine Learning

In general, ML in NIDS is applied in a supervised learning setting, where authors often assume the availability of properly labeled network traffic [12]. However, the settings of real network environments rarely are labeled, which translates into only a small subset of events' labels being previously known. With this constraint in mind, a reliable ML-based NIDS must be able to operate in a semi-supervised setting [9]. In doing so, both training and update tasks must be performed with only a subset of event labels.

In the literature, there are several approaches to achieve such a goal. One of the most popular approaches to increase the number of labeled events (i.e., self-labeling) relies upon co-learning techniques, which apply a set of ML models to label events for other models [9]. From the many co-learning techniques that have been proposed, Tri-Training [13] provides promising accuracy results. It relies on three classifiers for both classification and labeling tasks. The former is achieved by a simple majority voting, whereas the latter is performed for each classifier. The event label is assigned according to the majority voting of the other two classifiers. Although widely used in several fields, self-labeling techniques have their applicability to be verified on the evolving behavior of network traffic classification.

III. A Year of Network Traffic Anomalies

In this section, we describe a new dataset based on real network traffic [7] and use it to evaluate the accuracy of common supervised and semi-supervised techniques.

A. A year of network traffic anomalies: MAWIFlow dataset

One of the biggest challenges in building and evaluating ML algorithms for NIDS is the lack of a properly built training dataset. Such a dataset should be composed of network data with events (i.e., network packets) that are real, valid, variable, publicly available, and correctly labeled. One must record real data to provide such an enriched dataset, which renders unfeasible sharing it due to privacy concerns. Additionally, evaluating a model lifespan is even more difficult since data must be recorded for long periods.

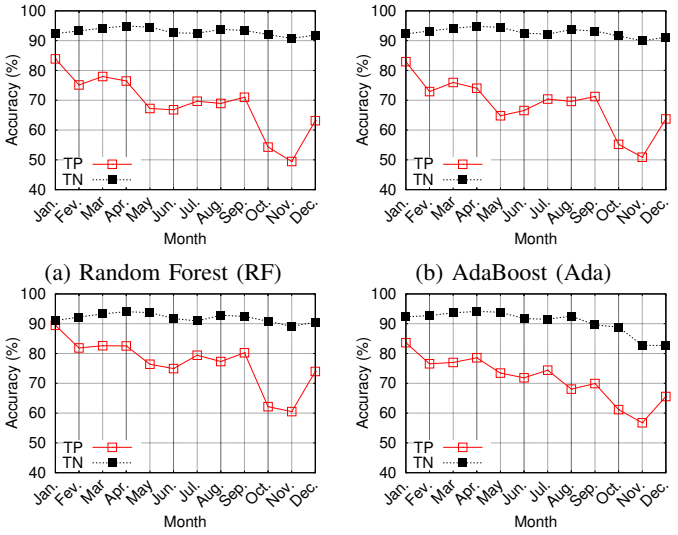
We use the MAWI network traffic [14] to overcome the mentioned challenges. More specifically, we used the MAWI Samplepoint-F in MAWI archive, which is a dataset made of real network traffic from a transit link between Japan and the USA collected for a 15-min-long interval daily. We selected the network traffic of the whole year of 2016, which allows one to evaluate models' lifespan and accuracy degradation. The resulting dataset contains more than 5TB of network data composed of 60 billion of network packets.

An unsupervised ML technique from MAWILab [15] was employed to automatically label the events—i.e., tag them as either normal or attack. MAWILab executes several unsupervised (i.e., no need for event labels) machine learning algorithms to find anomalies in MAWI data. These anomalies are labeled as attacks, while the remaining data is assumed to be normal events. The feature extraction module [7] groups events in intervals of 15 seconds and extracts 22 features from the work of Orunada *et al.* [16].

B. Accuracy behavior of ML-based NIDS over time

We evaluate the accuracy behavior of ML-based NIDS, both with supervised and semi-supervised ML techniques. Three supervised techniques were selected: Random Forest [17], Adaboost [18], and Decision Tree [19]. The semi-supervised technique was evaluated through the Tri-Training [13] algorithm using the three mentioned supervised classifiers.

Random Forest and Adaboost use a base-learner of 100 decision trees, while the Decision Tree is implemented through



(a) Random Forest (RF) (b) AdaBoost (Ada)

(c) Decision Tree (DT) (d) Tri-Training

Fig. 1. Accuracy behavior of traditional machine learning techniques on the *MAWIFlow* dataset. The Random Forest, AdaBoost and Decision Tree classifiers are not updated throughout time. The Tri-Training is updated monthly, according to the labels obtained by its self-assigned labeling technique.

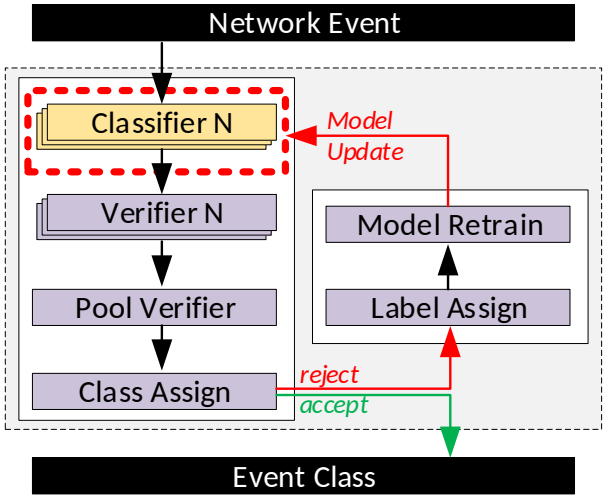


Fig. 2. Proposed reliable semi-supervised intrusion detection model.

the C4.5 algorithm with a confidence factor value of 0.25. We used the first week of January as the training dataset for the supervised techniques and evaluated them throughout the year without model updates. In the semi-supervised model, we updated the model monthly according to the self-assigned label, as obtained through the Tri-Training algorithm. The majority of the events are normal due to the imbalanced nature of the dataset. We conducted a stratification procedure in the training dataset to balance the classes occurrences and implemented the classifiers using the Weka API [20].

Figure 1 shows the accuracy obtained with the supervised and semi-supervised techniques over time. The first noticeable result is the significant accuracy degradation of all evaluated approaches. More specifically, the evaluated techniques significantly decrease the TP rates as soon as a few weeks after the training. For instance, the supervised Random Forest classifier decreased the TP rate by 8.75% in the month following the training, while reached a TP rate of only 49.44% in November (a significant decrease of 34.44% from its TP rate in January). Nevertheless, the semi-supervised approach, even with monthly updates, has decreased its accuracy rates. Similarly to supervised approaches, the traditional semi-supervised technique decreased the TP rate by 7.08% a month after the training, while reached its worst TP rate also in November.

The evaluation through the *MAWIFlow* dataset indicates that traditional ML-based techniques, supervised or semi-supervised ones, are unable to cope with the inherent changes of network traffic behavior over time. However, periodic model updates are unfeasible because they depend on updated models that may take several days or weeks to become available for deployment. Additionally, the dependency on correct event labels to update the model hardens the challenge of dealing with changes in the network traffic behavior over time.

IV. A Reliable Semi-Supervised Intrusion Detection Model

In this section, we present a novel semi-supervised and reliable intrusion detection model to address the evolving behavior of the network traffic mentioned above. It consists of two main steps (*verification* and *reliable semi-supervised update*, as shown in Figure 2) and aims to maintain (or even improve) the accuracy measurements obtained during the test phase, without human assistance.

The proposal considers a semi-supervised intrusion detection model composed of a pool of classifiers—for instance, the Tri-training algorithm. Classification starts with a to-be-classified network event that is forwarded to a pool of classifiers, where each classifier produces a (algorithm-dependent) classification confidence value. For instance, a Random Forest classifier computes this value based on the ratio of base-learners that classified the instance as belonging to a given class. The set of obtained classification confidence values are forwarded to the *verifier* module, which filters out values from classifiers that have not met a minimum threshold. Consequently, only highly confident classifications are accepted.

The *pool verifier* attests that the classified event has been accepted by at least a predefined number of classifiers, and this classification is accepted or rejected by the *class assign* module. Accepted classifications have high confidence values, which means they are more likely to be correctly classified since the pool of classifiers is more confident in their assigned labels. Contrarily, rejected instances are potentially misclassified events, since the used classifiers are uncertain on the assigned label. The system conservatively rejects potentially misclassified instances instead of accepting them.

Rejected instances are used by two components to update the models. First, the *label assign* module determines the proper event label according to the pool of classifiers, where each classifier receives the event according to the label assigned by the other classifiers. Then, the *model retrain* periodically and autonomously updates the pool of classifiers.

The next descriptions detail the verification and update stages of our proposal.

A. Verification

The goal of the *verifier* and *pool verifier* modules is to ensure that only highly confident classifications are accepted over time. Instances that are classified with highly confident values are known to the underlying classification models, which means they can be accepted without increasing the error rates. Additionally, such an assumption provides a reliable classification even in the lack of model updates since changes in the network behavior will affect the classifier confidence values and will be rejected by our model. Consequently, our system maintains its reliability even if the ML models are not updated over time, despite resulting in a higher rejection rate.

Our proposal leverages the confidence values of the underlying classifiers to evaluate the instance classification correctness. The *verifier* module rejects or accepts classifications based on predefined threshold values for each classifier, which are computed during the training to reach the desired level of reliability. Classifications accepted by the *verifier* are used by the *pool verifier* to establish whether it is reliable or not, based on the outcome of the pool of classifiers. The *pool verifier* accepts the evaluated instance if a majority of the classifiers have accepted their classifications, which also determines the assigned event label.

B. Reliable Semi-supervised Update

Classification verification helps to maintain system reliability over time. However, the lack of model updates would increase the proposal rejection rates. In addition, we seek to avoid the need for human intervention in this task, using our semi-supervised technique to leverage the verification module in the automatic labeling task. The update procedure receives the set of rejected instances as input, which means that instances that did not meet the confidence thresholds contribute to this step.

When a classifier rejects an instance, the *label assign* module tags the instance according to a majority vote of the other accepted classifications in the remaining classifiers. This self-labeled instance is stored until a periodic model update takes place (e.g., once a month). The *model retrain* component inserts all stored self-labeled instances to the initial training dataset, and a new pool of classifiers takes place.

C. Discussion

The proposed model addresses many challenges faced (and neglected) by ML-based NIDS in production. In real-world scenarios, the model update is not an easy task to accomplish, which requires the classification scheme to be reliable even with outdated models. Our proposal relies upon a verification technique (Section IV-A) to ensure that our scheme rejects potentially misclassified instances. Nonetheless, our proposal overcomes the update challenge through a reliable semi-supervised update technique (Section IV-B). Coping the update technique with the verification one allows our system to maintain (or even improve) the classification accuracy despite event rejections. Consequently, we provide ML model updates without human assistance while maintaining system reliability over time.

V. Evaluation

The present evaluation focuses on answering the following four research questions: ($Q1$) *Does the verification technique enable the proper assessment of the classification reliability?* ($Q2$) *Does the verification technique provide a reliable classification over time, even in the absence of model updates?* ($Q3$) *Can the proposed reliable semi-supervised update technique execute model updates without human assistance?* ($Q4$) *What are the tradeoffs between the model update periodicity, accuracy, and rejection rates?*

The next items describe how do we build the model and how does it perform when facing the *MAWIFlow* dataset.

A. Model Building

In this section, we build the proposed reliable semi-supervised intrusion detection model using the Tri-Training algorithm and its co-learning technique. We use the same three classifiers evaluated in Section III-B (i.e., Random Forest, Adaboost, and Decision Tree) as our proposal individual models (*classifier*, Figure 2). Each classifier is built using the first week of January of the *MAWIFlow* dataset and is evaluated throughout the year.

B. Verification of ML Model Classifications

The first experiment relates to Question $Q1$ and evaluates each classifier verification technique and finds the optimal rejection threshold for each one. We apply the Class-Related-Threshold (CRT) [21] technique on each classifier using the second week of January of our dataset. We find two thresholds: one used to accept or reject normal classified events and another one for events classified as attacks. Figure 3 presents the relation between rejection and error rates for each classifier.

A direct correlation between the rejection and the error rates is observed, which means that the *verifier* improves system reliability even without model updates.

The second experiment relates to Question $Q2$ and applies the verification technique over the whole *MAWIFlow* dataset. We select individual operation points (i.e., class thresholds) for each classifier. Operation points are selected when a 10% error rate is achieved in the second week of January (see *operation point* in Figure 3). Figure 4 presents the accuracy and rejection rates throughout the whole *MAWIFlow* dataset of our model with the verification technique and no model updates. A first observation is that the rejection rate increases

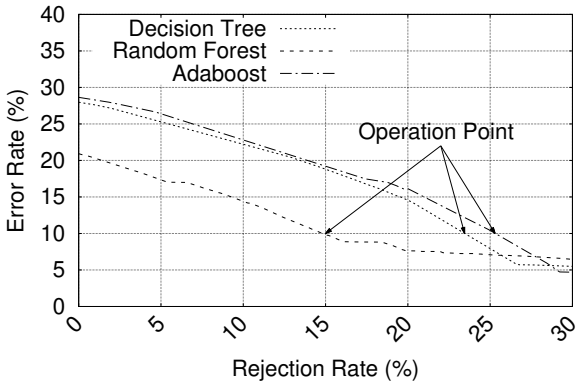


Fig. 3. Tradeoff between the error rate of the verification technique and the rejection rate for each individual classifier in the second week of January in *MAWIFlow*.

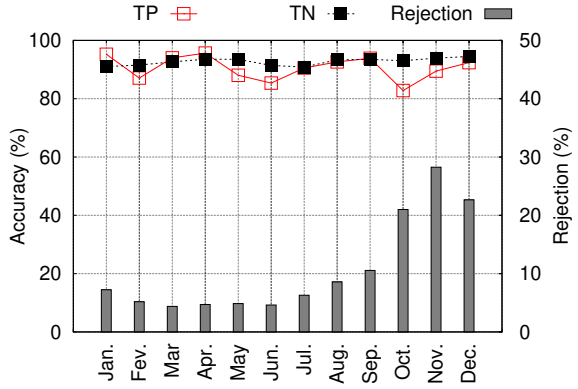


Fig. 4. Proposed reliable semi-supervised intrusion detection model with *verifier*, without periodic model updates.

over time (reaching 28.26% in November), which is a side-effect caused by the lack of model updates. Nonetheless, the verifier approach is able to maintain the accuracy rates while rejecting in average 10.6% of instances. However, the main result is the indication that the *verifier* maintains (or even improve) accuracy rates throughout the year, even in the absence of model updates.

C. Reliable Semi-supervised Updates

The third experiment relates to Question $Q3$ and evaluates the impact of monthly model updates on accuracy and rejection rates. These updates are autonomous and occur through the *label assign* module (see Figure 2). Figure 5 presents the accuracy and rejection rates of the proposed model over time, considering the *verifier* module is applied with monthly updates. The monthly updates incurred a significant reduction in the rejection rate, decreasing it from 10.6% to only 3.2%. However, the accuracy rate has not significantly improved, which shows that the verification technique maintains the system reliability regardless of the presence of model updates.

Finally, the fourth experiment relates to Question $Q4$ and measures the relation between model update periodicity, accuracy, and rejection rates over time. Figure 6 presents the

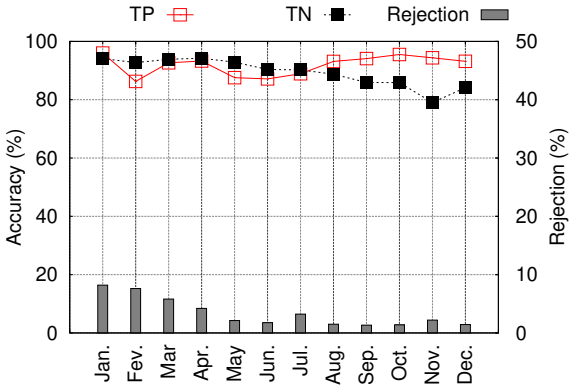


Fig. 5. Proposed reliable semi-supervised intrusion detection model with *verifier*, and monthly model updates without human-assistance.

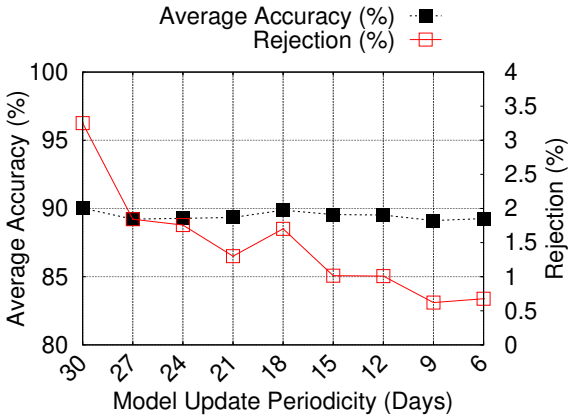


Fig. 6. Relation between model update periodicity, accuracy and rejection tradeoff for the proposed reliable semi-supervised intrusion detection model.

balance between these three properties. The more frequent the model updates, the lower the rejection rate is. However, the accuracy rate does not significantly varies, which corroborates the hypothesis that the proposed verification technique does not require model updates to provide reliability over time.

D. Discussion

The proposed semi-supervised intrusion detection model overcomes the challenges that changes in network traffic behavior bring to ML-based NIDS. The fact that only high confidence events are accepted by our model results in the proposed verification technique being able to maintain system reliability even in the absence of model updates. Additionally, our approach provides reliable semi-supervised model updates by leveraging the verification technique, which significantly the rejection rate with no side-effects on accuracy. In summary, the proposed model provides a reliable ML-based NIDS classification since it performs reliable classifications over time and provide updated ML models autonomously, i.e., without human intervention.

VI. Related Work

Over the last years, several works have proposed highly accurate ML-based NIDS for network traffic classification [12]. However, despite their promising results, their actual deployment in real-world environments remains scarce [6]. It is essential to overcome the task of updating the underlying ML models [7] to overcome this practical gap in ML-based NIDS.

Published related works usually assume that model updates are executed periodically [7]. For instance, Xu *et al.* [22] proposed a semi-supervised intrusion detection model that applies a one-class support-vector-machine to detect network anomalies. However, their approach demands human-assistance for the model update task. Zhou *et al.* [23] uses a multi-view technique for self-labeling classification. However, they do not specify how the model update is performed.

Another common assumption in the literature is the availability of event labels for the update task. For instance, Correa *et al.* [24] applied supervised stream learning techniques for the intrusion detection task. They incrementally update the underlying ML model with the new instance. However, they assume that the correct event instance label is available, which is not always true in production environments. The verification of the classification output has been extensively applied in other fields, such as Optical Character Recognition (OCR), fault detection, and medical diagnosis analysis. Surprisingly, its applicability in intrusion detection remains scarce. In a previous work [25], we used a single classifier to verify the detection of unknown behavior of network traffic. It detects behaviors similar to those in the training dataset to overcome the flaws of building a proper intrusion dataset.

To best of our knowledge, this is the first proposal to address the reliability of model classifications and updates without requiring human assistance. We provide autonomously updated classification models that maintain reliability over time.

VII. Conclusions

Several recent works have been addressing network-based intrusion detection through machine learning techniques. Despite their promising results, they left open a gap in transferring their solutions to deployments in production. We have proposed a novel reliable semi-supervised intrusion detection model to overcome the challenge of network traffic behavior changing over time. We apply a verification technique to evaluate the reliability of ML model classifications and deal with semi-supervised machine learning to autonomously update the underlying ML models. The results of the proposal indicate that the solution maintains the reliability of the system along of time and provides updated classification models without human intervention.

Acknowledgment

This work was partially supported by the CNPq (National Council for Scientific and Technological Development) grant 315322/2018-7 and by the FCT through the LASIGE Research Unit, ref. UIDB/00408/2020.

References

[1] Symantec Corporation, "Internet security threat report," 2019, pp. 1–61. [Online]. Available: https://www.symantec.com/security-center/threat-report

[2] Kaspersky Lab, "A DDoS storm has come: number of attacks grows after long period of decline," 2019. [Online]. Available: https://www.kaspersky.com/about/press-releases/2019_a-ddos-storm-has-come-number-of-attacks-grows-after-of-decline

[3] L. H. Newman, "GitHub survived the biggest DDoS attack ever recorded," 2018. [Online]. Available: https://www.wired.com/story/github-ddos-memcached/

[4] M. Ring *et al.*, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[5] 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, 2017.

[6] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. of the 31st IEEE Symposium on Security and Privacy*, 2010.

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

[8] C. Vicentini, A. Santin, E. Viegas, and V. Abreu, "SDN-based and multitenant-aware resource provisioning mechanism for cloud-based big data streaming," *Journal of Network and Computer Applications*, vol. 126, pp. 133–149, Jan. 2019.

[9] I. Triguero, S. García, and F. Herrera, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," *Knowledge and Information Systems*, vol. 42, no. 2, pp. 245–284, 2013.

[10] F. Maggi, W. Robertson, C. Kruegel, and G. Vigna, "Protecting a moving target: Addressing web application concept drift," in *Springer Lecture Notes in Computer Science*, 2009, pp. 21–40.

[11] 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.

[12] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[13] Z.-H. Zhou and M. Li, "Tri-training: exploiting unlabeled data using three classifiers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, 2005.

[14] MAWI, "MAWI Working Group Traffic Archive - Samplepoint F," 2019. [Online]. Available: https://mawi.wide.ad.jp/mawi/

[15] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proc. of the 6th Int. Conf. on emerging Networking EXperiments and Technologies (CoNEXT)*, 2010.

[16] J. Dromard, G. Roudiere, and P. Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 34–47, 2017.

[17] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[18] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class AdaBoost," *Statistics and Its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[19] S. Ruggieri, "Efficient C4.5 [classification algorithm]," *IEEE Trans. on Knowledge and Data Engineering*, vol. 14, no. 2, pp. 438–444, 2002.

[20] Weka, "Weka 3: Machine learning software in Java," 2019. [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/

[21] G. Fumera, F. Roli, and G. Giacinto, "Reject option with multiple thresholds," *Pattern Recognition*, vol. 33, no. 12, pp. 2099–2101, 2000.

[22] S. Xu, Y. Qian, and R. Q. Hu, "A semi-supervised learning approach for network anomaly detection in fog computing," in *Proc. of the Int. Conference on Communications (ICC)*, 2019, pp. 1–6.

[23] M. Zhou *et al.*, "SCTM: A multi-view detecting approach against industrial control systems attacks," in *Proc. of the Int. Conference on Communications (ICC)*, 2019, pp. 1–6.

[24] D. G. Correa, F. Enembreck, and C. N. Silla, "An investigation of the hoeffding adaptive tree for the problem of network intrusion detection," in *Proc. of the Int. Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 4065–4072.

[25] E. Viegas *et al.*, "A reliable and energy-efficient classifier combination scheme for intrusion detection in embedded systems," *Computers & Security*, vol. 78, pp. 16–32, 2018.