Improving Intrusion Detection Confidence Through a Moving Target Defense Strategy

Roger R. dos Santos, Eduardo K. Viegas, Altair O. Santin
Graduate Program in Computer Science (PPGIa)
Pontifical Catholic University of Parana (Pontifícia Universidade Católica do Paraná), Brazil
{robson.roger, eduardo.viegas, santin}@ppgia.pucpr.br

***Abstract*—Despite the promising results reported in the literature, the intrusion detection schemes cannot deal with new network traffic behaviors making such proposals unfeasible to be deployed in production environments. This paper presents an intrusion detection model that relies on a moving target defense strategy to face new network traffic behavior in a two stage process. First, the system select the most suitable classifiers set to assign a class (normal or attack) according to the current event behavior. Second, we evaluate if the performed classification is reliable by validating its confidence values. The goal is to ensure that only the higher confident classifications from the most suitable classifiers are used to trigger intrusion detection alerts, keeping the system reliable over time. Experiments performed on a dataset that spans over 97GB of data with seven categories of network traffic shows that current machine learning techniques cannot cope with novel traffic behavior, failing to detect up to four new traffic categories. In contrast, the proposed model can select the most confident classifiers, reducing the average false-negative rates by up to 39%, regardless of the current network traffic category.**

***Index Terms*—Intrusion Detection, Machine Learning, Moving Target Defense.**

I. Introduction

In general, administrators resort to Intrusion Detection Systems (IDS) to detect malicious activities within an environment, through either *misuse-based* or *behavior-based* approaches [1]. In one hand, *misuse-based* techniques searches for well-known attack patterns in its input data, thus, are only able to detect previously known attacks [2]. On the other hand, *behavior-based* approaches signal misconduct by analyzing deviations from the system's expected behavior. Therefore, it can detect new kinds of attacks as long as the extracted behavior significantly differs from benign ones [3].

Several approaches have been proposed for the *behavior-based* intrusion detection task, wherein pattern recognition through machine learning (ML) techniques have yield promising results [1]. In practice, a behavioral model is extracted from a training dataset in a computationally expensive process of model training [4]. The input data is expected to contain the labeled event behaviors from the production environment, considering both normal and attack patterns. Finally, the extracted model can be used for the classification of network traffic events in production. In the literature, many proposals have focused on building the most accurate ML model for intrusion detection on a given dataset, e.g., applying an ensemble of classifiers [5]. However, the behavior of networked environments is highly variable and changes over time [6]. Even if a *'perfect'* ML model is built, it will fail at considering the network traffic as an unchangeable environment, as measured through the training dataset [7]. Nonetheless, ML extracts its behavioral model by finding similarities in its input data rather than detecting new behavior.

It is known that the network traffic changes over time, either due to the discovery of new attacks or due to the provision of new services, the reliability of the intrusion detection scheme can not be ensured [8]. Thus, the occurrence of new attacks or services categories may affect the confidence of the underlying IDS ML model, increasing its signaled false alarms. Changes in production environment settings demand the model retraining task to be executed [2]. However, the ML model retraining requires labeled data to be provided, which can not be available or provided with human intervention with a high cost [6]. Additionally, the retraining task may demand several weeks or even months to provide an updated ML model. The current ML model in production must keep its confidence in intrusion detection for long periods, even in the presence of new attacks and service behaviors.

This paper proposes a new reliable intrusion detection model that leverages a moving target defense strategy to select the most suitable ML configuration for the behavior of the current environment, and the implementation is done in twofold. First, it chooses the ML configuration adaptation to consider the evaluated event behavior by assessing the classification confidence. The proposal insight is that the most suited classifier can be obtained by assessing the classification confidence values from a classifier pool. The proposal can choose the adaptation according to the event behavior without human assistance neither model updates. Second, it evaluates if the selected ML configuration can be reliably used to classify a given network traffic event. Thus, only highly confident classifications from the most suited classifiers are used for signaling intrusion alerts, increasing system stability.

In summary, the main contributions of this paper are:

- We evaluate commonly used ML-based intrusion detection schemes concerning their classification confidence in the face of new attack and service categories. Experiments performed over a dataset that span several distinct normal and attack categories shows that current approaches in the literature are unable to cope 4 out of 7 new traffic behaviors;

- We propose and evaluate a novel reliable intrusion detection model that follows a moving target defense strategy. The proposed model can select the most suited classifiers for classifying a given network traffic event while maintaining its confidence in the classification even in the presence of new network traffic.

II. Background

A. Machine Learning for Intrusion Detection

Intrusion Detection Systems (IDS) aims at finding malicious activities within a given environment [1]. A typical IDS is composed of four sequential modules. First, the *Data Acquisition* collects the environment events for further analysis, e.g., collect network packets from a network interface card (NIC). Then, the *Feature Extraction* module extracts a set of behavioral features from the collected data to compound a feature vector. In general, the behavior of network events is represented as a network flow, which summarizes network packet statistics in a given time window, e.g., the number of exchanged packets between hosts in 2-sec [9]. The extracted feature set is classified by a *Classification* module as *attack* or *normal*, e.g. by applying a ML model. Finally, if an *attack* is found, the *Alert* module signals the operator accordingly.

Recently, several ML approaches have been proposed for the classification task [10]. In general, authors mistakenly adopt the assumptions wherein ML has been successfully applied [6], [11]. The behavior of network environments is highly variable and changes over time. As a result, building a realistic training dataset with many network behaviors is a challenging task. Nonetheless, even if a realistic training dataset can be built, it will become outdated as time passes. Therefore, besides providing a highly accurate ML model, the used intrusion detection scheme must be able to cope with variations of known traffic and new traffic behaviors [2]. However, authors often overlook the challenges of network traffic classification, and traditional ML evaluation takes place [6].

B. Moving Target Defense

The Moving Target Defense (MTD) strategy aims to tackle the static nature of computational systems by constantly changing the system settings to reduce or move the attack surface that a malicious user can explore [12]. For instance, change a server set of software, service ports, IP address, or other components that can be a target of attacks [5]. Thus, MTD can hinder the attacker's goal by constantly increasing the attacker's uncertainty on a given server.

Traditionally, MTD strategies can be interpreted as a three-step approach [12]. First, it constantly *Choose an Adaptation* for a given computational environment settings. For instance, the adaptation can be achieved by changing the servers' service ports and IP addresses. Second, it *Implement Adaptation* by applying the selected server settings in a computational environment. Third, it waits for a predefined *Delay* before choosing a new setting, hence, executing the first step again continuously and periodically. The MTD rationale is that a successful attack demands several steps to be executed, e.g., mapping ports, identifying services, identifying vulnerable components, and exploiting exploitation. Hence, if such settings are periodically changed, the attacker will not be able to perform all needed attack steps before a reconfiguration occurs, making the attacker execute all required actions from scratch [13].

III. Related Works

Intrusion detection tasks through ML techniques have been extensively explored in the literature [1]. In general, proposed techniques aim at higher classification accuracies on a specific dataset, with either real [2] or synthetic [14] network traffic. Despite relying on a realistic dataset for the training task, ML techniques must be evaluated taking into account the characteristics of networked production environments. Li Yang *et al.* [5] proposes a tree-based structured IDS to identify attacks. Their proposed approach was able to improve detection accuracies on CICIDS2017 [15] dataset. They used a traditional ML evaluation to identify new attacks or services that were not addressed.

The network traffic behavior changes were noted by Jielun Zhang *et al.* [16], which used a transfer learning approach on a neural network to update the detection scheme when a new service is provided. However, identifying new services is not addressed, and the impact on accuracy is not evaluated accordingly. Network traffic behavior changes are also addressed by E. Viegas *et al.* [2], where a co-learning approach is used for the model update task. The proposed scheme was able to update the underlying model reliably. However, the impact of the network traffic changes can only be addressed after the model update, leaving the IDS scheme unreliable. Therefore, although the model update task has been addressed in the literature [2], [16], the confidence impact due to new behaviors occurred while an outdated model is used is not addressed [3], [6].

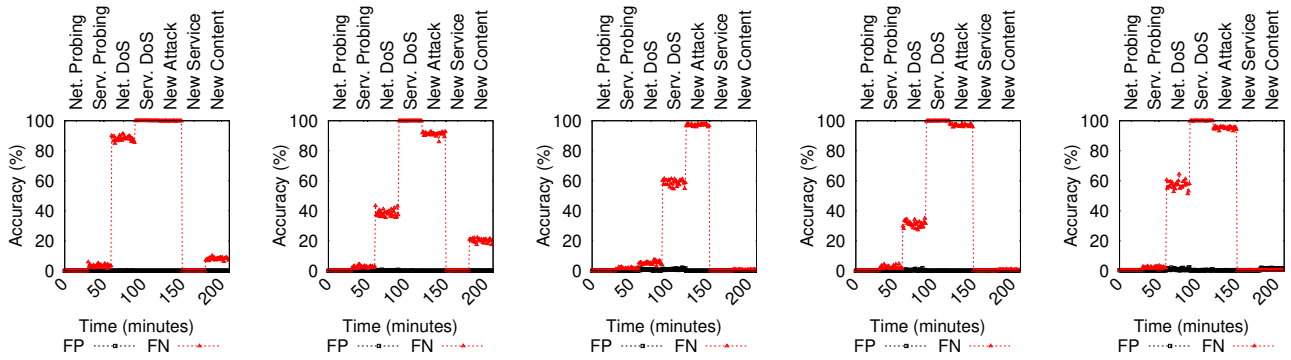
MTD strategies are often applied using an IDS as an adaptation triggering technique. R. Yang *et al.* [5] relies on a network-based MTD approach to decrease the time to failure of service components, assuming that attacks can be detected to trigger MTD procedures. D. Sharma *et al.* [13] relies on a software-defined network to dynamically change network settings according to the current attacker path, also assuming that attacks can be identified as needed. Therefore, the application of the MTD strategy on the IDS scheme remains unknown in the literature.

IV. Problem Statement

The identification of new attacks and services is a widely used assumption on the *behavior-based* IDS literature [1], [6]. This section further investigates how ML techniques perform when facing new network traffic behavior. More specifically, we first introduce our used dataset, with several attacks and normal network traffic categories, and evaluate several ML classifiers' performance on it.

TABLE I: Testbed behavior variations, over time, according to each considered scenario. Both normal and attacker behaviors vary as time passes.

Scenario	Time Window (minutes)	Attacker Behavior	Normal Behavior
Net. Probing	$zero$ to 30	$udpscan$, $synscan$, $nullscan$, $finscan$, $xmasscan$, and $ackscan$	The 100 benign clients performs periodic queries on HTTP, and SNMP services
Serv. Probing	30 to 60	$osfingerprint$ and $servicefingerprint$	
Net. DoS	60 to 90	$synflood$, $udpflood$, $icmpflood$, and $slowloris$	
Serv. DoS	90 to 120	$smtpflood$, or $httpflood$	
New Attack	120 to 150	$vulnerabilityscan$	
New Content	150 to 180	$synflood$, $udpflood$, $icmpflood$, and $slowloris$ (Same behavior as *Net. DoS* scenario)	Requested service content differs from previous scenario
New Serv.	180 to 210		The 100 benign clients performs periodic queries on SMTP, NTP and SSH services



(a) Random Forest (b) Gradient Boosting (c) Adaboosting (d) Bagging (e) Multilayer Perceptron

Fig. 1: The accuracy behavior, over time, on the evaluated testbed of widely used machine learning classifiers. Classifiers are trained using only *Net. Probing* data occurred at a time window ranging from $zero$ to 30 minutes.

A. A Realistic IDS Dataset

The building and evaluation of intrusion detection schemes demand a realistic intrusion dataset to be used [1]. In the literature, authors resort to outdated datasets, with several known flaws [17]. However, even if a realistic dataset contains real and valid network traffic collected from production environments, it will fail to consider the network as static. Therefore, to properly evaluate ML-based intrusion detection schemes, we present *Fine-grained Intrusion Dataset* (FGD), built through the same methodology as [9]. The dataset comprises several network behaviors that appear in the real world that ML models must process in production environments.

The dataset was built in a controlled environment composed of 100 client machines that generate benign network traffic of various application protocols and several attacker machines that generate malicious traffic through well-known tools [9]. The machines performed the requests and attacks at a honeypot server responsible for the proper response and the collection of the network traffic. Table 1 shows a description of the testbed behavior variation over time regarding both normal and attacker entities. The detailed entity's behavior can be found in [9]. The testbed was executed for 4 hours while varying the generated network traffic behavior every 30 minutes. In total, the testbed execution resulted in 97.6 GB of network traffic, composed of 7 possible behavior variations, in which 5 are related to attack, and 2 concerning normal behavior variations.

B. A Realistic IDS Evaluation

To further investigate the impact of network traffic behavior changes on ML-based IDS, we consider a training environment composed of network-related probing attacks, such as port scans (Table I, *Net. Probing*). The evaluated ML algorithms are trained using as input the data from $zero$ to 30 minutes time window from our testbed (Table I). The behavior of network data, used as input by ML algorithms, is represented by a feature vector composed of 50 features [9], which summarizes the exchanged data between each testbed host and their related services in a 2-second time window.

To evaluate widely used ML-based approaches, we select five commonly used classifiers for intrusion detection, namely Random Forest (RF), Gradient Boosting (GBT), Adaboosting (Ada), Bagging (Bag), and Multilayer Perceptron (MLP). The ensemble classifiers (RF, Ada, Bag, and GBT) were implemented with 100 decision trees as their base-learners, where each one of them also uses $gini$ as the node split quality metric. The GBT classifier relies upon a 0.1 learning rate value, with $deviance$ as the loss function. The Ada classifier uses the $SAMME$ as boosting algorithm and 1.0 as the learning rate. The MLP classifier relies on 100 neurons on its hidden layer and 200 training epochs with a 0.001 learning rate. A random undersampling without replacement is used in the training procedure to balance the occurrence between the classes. The classifiers were implemented through $scikit-learn$ API v0.24. The classifiers are evaluated con-

cerning their false-positive (FP) and false-negative (FN) rates, where the FP denotes the ratio of normal events misclassified as an attack. In contrast, the FN represents the ratio of attack events misclassified as normal ones.

Figure 1 shows the accuracy variation over time of the evaluated classifiers. The evaluated approaches presented a significantly high accuracy rate while facing already known network behavior ($zero$ to 30minutes, *Net. Probing*). Evaluated techniques presented an average of only 0.2% and 0.1% of FP and FN rates, respectively. However, as time passes and the behavior of network traffic changes, evaluated approaches cannot reach the same level of reliability in classification. In such a case, evaluated techniques were able to present high classification accuracies (false rates $< 5\%$) when facing only 2 kinds of new network behavior (*Serv. Probing* and *New Content*). Therefore, if only a 5% increase in FP and FN rates is tolerated, 4 classifiers will be deemed unconfident for 4 out of 6 new behavior variations, which includes *Net. DoS*, *Serv. DoS*, *New Attack*, and *New Serv*, while only 1 classifier (Ada) will reliably for 3 our of 6 scenarios.

As a result, evaluated approaches cannot cope with the natural variability of network traffic over time. As new behavior occurs, either from attackers or benign clients, the false alerts will be significantly increased.

V. A Moving Target Defense Strategy to Improve Confidence of Intrusion Detection

To address the challenge mentioned above that considers an evolving network traffic behavior, we present a novel intrusion detection model that is confident and based on a moving target defense strategy. The proposal overview is shown in Figure 2, and aims to maintain or even improve the detection confidence facing the new network traffic behavior through two modules, the *Choose Adaptation* and *Implement Adaptation*. The insight of our proposal is that the classification confidence values can be used as a measure to select, from a pool of classifiers, those that are candidates for making the best classification task. Hence, adopting the classification settings without performing model updates, aiming to maintain the system reliability in a production environment.

The proposal considers an ML-based intrusion detection mechanism composed of a pool of classifiers that are interchangeably applied to the classification task. The classification procedure starts with a networking event collected for the classification purpose. The behavior of the collected network event is extracted by a feature extraction module, compounding a feature vector (set). The computed feature set is forwarded to a pool of classifiers (Figure 2, *Classifier N*), in which each classifier outputs related classification confidence. The confidence values are applied as a measure for evaluating the classifier correctness during the network traffic content classification (further described in Section V-A). Thus, the verifier module selects only those classifiers that meet a predefined correctness threshold. The IDS mechanism can select the most suited classifier to assign a class (normal or attack) to the event and associating the confidence that adapts

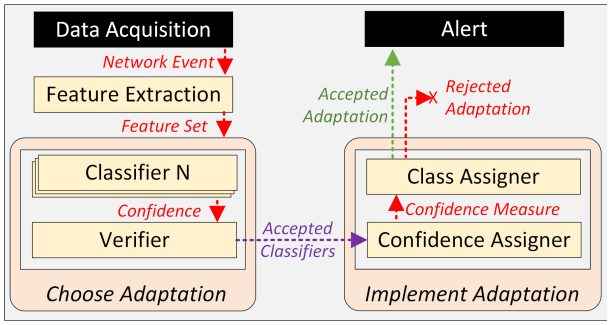


Fig. 2: Proposed moving target defense strategy to improve confidence of intrusion detection.

itself to the current environment behavior. Finally, the selected classifiers are used to compound an event confidence measure used as a rejection adaptation criteria of our proposal. The rationale of the confidence measure is to establish if our system can classify a given network event with confidence. This is because, in some cases, the current environment behavior may not be reliably classified by any of the selected classifiers.

The following subsections further describe the rationale of our proposal, including the modules that implement it.

A. Choose Adaptation

The behavior of networked environments follows a non-static nature. Consequently, a confident IDS mechanism must be able to adapt its classification settings accordingly. However, the model update task is not easily feasible in production environments and may take several weeks or even months to be performed. Thus, the system must be able to cope with new network behavior, even whether outdated.

Our proposal relies on a moving target defense strategy during event classification. Each collected event is processed by a pool of classifiers, while the system adapts itself to the current environment behavior by choosing the most suited one for the classification task. Although the models are not updated, the system can adapt itself by changing the set of classifiers taking into account the classification confidence of each model. The classification confidence values are used as a measure of model correctness on the event class (normal or attack) assignment task. The classification confidence is agnostic, e.g., the RF classifier outputs the classification confidences values according to the ratio of its decision trees that labeled the classified event. The assumption is that the classification confidence can select the most suited classifiers according to the current event behavior.

The *Choose Adaptation* module receives as input a event feature set for classification, which is then evaluated by a set of classifiers (Figure 2, *Classifier N*). Each classifier outputs a classification confidence value used by the verifier module to select the most suited classifier. In this case, a rejection threshold is used to attest to the classification confidence of each underlying model. Therefore, if a classifier does not meet a classification threshold, the event is rejected. According to the administrator's needs, the threshold values, e.g., a

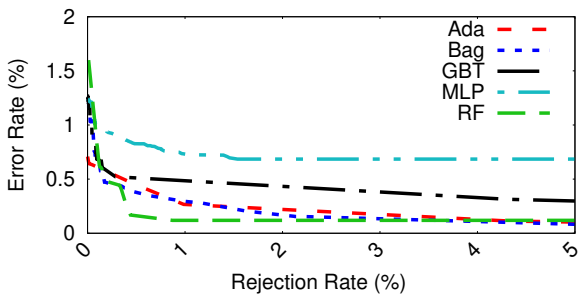


Fig. 3: The rejection rate of each classifier in the proposed scheme for the classification pool.

higher value, provide higher classification confidence. Finally, the chosen classifiers are considered to the next module, responsible for ensuring the classification's validity.

B. Implement Adaptation

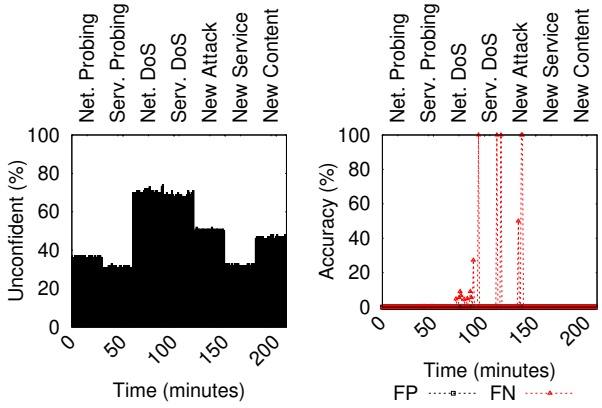
The *Implement Adaptation* module receives as input the selected set of classifiers from the verifier module. Its goal is to output a classification outcome for the collected event. The final event class is assigned using a majority voting procedure from the accepted classifiers. However, as several or no classifiers can be deemed confident for an event classification task, the module must first compute the confidence measure as a final acceptance criterion. Thus, even if several classifiers are considered confident for classifying a given event, the alert is only triggered considering the established confidence measure. The computation of the confidence measure is shown in Eq. 1, where N denotes the number of classifiers deemed as reliable, and $conf.$ the classification confidence values to $attack$ or $normal$ classes.

$$confidence = max(\prod_{i=1}^{N} conf._{attack}^{i}, \prod_{i=1}^{N} conf._{normal}^{i}) \quad (1)$$

The confidence measure denotes the maximum value from the product operator of either $attack$ and $normal$ confidence values from all accepted classifiers. Thus, even if several classifiers accept an event, the confidence measure will only be high if a consensus between accepted classifiers is reached, i.e., if all accepted classifiers labeled the evaluated event to the same class. Finally, the confidence measure is used to accept or not the system adaptation (Figure 2, *Accepted Adaptation*, or *Rejected Adaptation*). Rejected adaptations have their alerts suppressed, as the event classification reliability is low, even when using only the most suited classifiers as obtained from the *Choose Adaptation* module. As a result, our proposal can ensure confidence in event classification, even if none of the available classifiers can be used to classify the input event.

VI. Evaluation

The evaluation aims at answering the following research questions: ($Q1$) *Is the proposed adaptation accessible to improve system confidence?* ($Q2$) *How does the proposed model behave in the face of a new traffic behavior?* ($Q3$) *Is the proposal able to improve detection confidence?*



(a) Rate of instances deemed as unconfident over time in the testbed

(b) Accuracy rate on confident instances concerning behavior variations

Fig. 4: Evaluation of the confident intrusion detection model. The proposed scheme is trained using only *Net. Probing* data, occurred at time window $zero$ to 30 minutes (Table I), and evaluated using a 50% rejection rate.

A. Reliability in Intrusion Detection

The proposed scheme was evaluated on the same testbed evaluated previously (see Section IV, Table I). The proposal classifier pool is composed of the same set of classifiers evaluated previously (Figure 1), namely RF, GBT Boosting, Ada, Bagging, and MLP. The classifiers are trained using the same set of parameters and data that occurred during the *Net. Probing* attacks (Table I, $zero$ to 30minutes). The first experiment aims at answering question $Q1$, and evaluates if classification confidence can be used to attest classification reliability (Figure 2, *Choose Adaptation*). For each classifier a rejection threshold is defined taking into account the classification confidence values from the *Serv. Probing* scenario (Table I, 30 to 60 minutes). The confidence values for each classifier are obtained through the $predictproba$ from $sklearn$ API. One can recall that the established thresholds are used for selecting the accepted classifiers (Figure 2, *Accepted Classifiers*). The selected set of thresholds remains unchanged throughout the testbed execution. For rejection threshold, is used the Class-Related-Threshold (CRT). Thus, each classifier has two rejection thresholds (one for *normal* and one for *attack*). Figure 3 shows the error $vs.$ rejection rates for each used classifier. A high rejection rate can improve the accuracy rates of the evaluated classifiers. Thus, it can be used to attest to the system's reliability as time passes.

The second experiment aims to answer question $Q2$ about classification performance. Each classifier's rejection thresholds are set at a 50% rejection rate and used throughout the testbed data. One can recall that both rejection thresholds should be defined according to the administrator's needs and set a high rejection rate on our evaluation to investigate the

TABLE II: Accuracy of evaluated techniques according to the testbed behavior, considering reject events. (Fig. 4a)

.

Testbed Behavior	Detection Measure	*Detection Approach Accuracy (%)*					
		Random Forest	Gradient Boosting	Adaboosting	Bagging	MLP	Proposed Approach
Net. Probing	FP	0	0.0	0.0	0.0	0.4	0.0
	FN	0.0	0.0	0.01	0.0	0.2	0.0
Serv. Probing	FP	0.0	0.0	0.0	0.0	0.3	0.0
	FN	3.3	2.5	1.4	2.2	2.1	0.05
Net. DoS	FP	0.0	0.1	0.7	0.4	1.1	0.0
	FN	88.1	38.3	5.2	31.1	57.2	2.58
Serv. DoS	FP	0.0	0.0	1.1	0.0	0.4	0.0
	FN	99.9	99.9	58.6	99.8	99.9	10.0
New Attack	FP	0.0	0.0	0.0	0.0	0.3	0.0
	FN	99.8	91.1	97.1	96.9	95.2	8.33
New Content	FP	0.0	0.0	0.0	0.0	0.3	0.0
	FN	0.0	0.0	0.0	0.6	0.0	0.0
New Serv.	FP	0.0	0.0	0.0	0.2	1.6	0.4
	FN	8.0	20.1	0.6	0.6	0.3	0.0
All Scenarios	FP	0.0	0.0	0.2	0.1	0.6	0.05
	FN	42.7	36.0	23.3	32.9	36.4	2.99

performance further. Figure 4a shows the rejection rate, and figure 4b the related accuracy rates throughout the testbed. The proposed approach significantly improved detection accuracy even when facing new network traffic behavior. For instance, if up to a 5% error rate increase is tolerated, it maintains the detection confidence for all evaluated scenarios. Noteworthy, the confidence measure was also able to detect unreliable classifications, as noted by a rejection rate increase on *New Attack* scenario.

Finally, for answering question $Q3$, we compare the average accuracy of our proposal $vs.$ the traditional approaches over each evaluated testbed (Figure 1). In general, the proposed scheme was able to improve detection accuracies when compared to traditional approaches significantly. For instance, considering the average accuracy (Table II, *All Scenarios*), our proposed model increased the FP by only 0.05% for both RF and GBT, while improved the FP rates by up to 0.15%, 0.05%, and 0.55 when compared to the Ada, Bagging and MLP classifiers respectively. However, the FN rates are significantly decreased by our proposal, as it can find the most suited set of classifiers for new attacks, hence improving the FN rates by up to 39.71%, 33.01%, 20.31%, 29.91%, and 33.41% when compared to the RF, GBT, Ada, Bagging and MLP respectively. Therefore, the proposed model based on a moving target defense provides confidence in intrusion detection, even when facing new network traffic behavior.

VII. Conclusion

This paper has shown that current ML-based schemes in the literature cannot cope with new network traffic behaviors, significantly degrading their accuracy in such situations. Besides, we have proposed a confident intrusion detection model based on a moving target defense. The proposed model provided significantly higher accuracy on intrusion detection even when facing new network traffic behavior. As future works, we plan on extending our proposed model to incorporate model updates during model adaptation.

Acknowledgment

This work was partially sponsored by Brazilian National Council for Scientific and Technological Development (CNPq), grant nº 315322/2018-7.

References

[1] 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.

[2] E. K. Viegas, A. O. Santin, V. V. Cogo, and V. Abreu, "A reliable semi-supervised intrusion detection model: One year of network traffic anomalies," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. IEEE, Jun. 2020.

[3] 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.

[4] 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 *ICC 2021 - IEEE International Conference on Communications*. IEEE, Jun. 2021.

[5] L. Yang, A. Moubayed, I. Hamieh, and A. Shami, "Tree-based intelligent intrusion detection system in internet of vehicles," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Dec. 2019.

[6] 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.

[7] 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.

[8] F. Pinagé, E. M. dos Santos, and J. Gama, "A drift detection method based on dynamic classifier selection," *Data Mining and Knowledge Discovery*, vol. 34, no. 1, pp. 50–74, Oct. 2019.

[9] 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.

[10] R. L. Tomio, E. K. Viegas, A. O. Santin, and R. R. dos Santos, "A multi-view intrusion detection model for reliable and autonomous model updates," in *ICC 2021 - IEEE International Conference on Communications*. IEEE, Jun. 2021.

[11] 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.

[12] R. Zhuang, S. A. DeLoach, and X. Ou, "Towards a theory of moving target defense," in *Proceedings of the First ACM Workshop on Moving Target Defense - MTD '14*. ACM Press, 2014.

[13] D. P. Sharma, S. Y. Enoch, J.-H. Cho, T. J. Moore, F. F. Nelson, H. Lim, and D. S. Kim, "Dynamic security metrics for software-defined network-based moving target defense," *Journal of Network and Computer Applications*, vol. 170, p. 102805, Nov. 2020.

[14] J. Liang and M. Ma, "Co-maintained database based on blockchain for idss: A lifetime learning framework," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2021.

[15] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. of the 4th Int. Conf. on Inf. Systems Sec. and Priv.* SCITEPRESS - Science and Technology Publications, 2018.

[16] J. Zhang, F. Li, H. Wu, and F. Ye, "Autonomous model update scheme for deep learning based network traffic classifiers," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Dec. 2019.

[17] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 40, no. 5, pp. 516–524, 2010.