Toward a Reliable Evaluation of Machine Learning Schemes for Network-Based Intrusion Detection

Eduardo K. Viegas, Altair O. Santin, and Pietro Tedeschi

Abstract

Over the last years, several works introduced network-based intrusion detection schemes based on machine learning techniques for securing IoT devices. Despite the promising results, proposed approaches are rarely adopted in production environments. Networked environments exhibit highly unpredictable behavior, unlike other areas where machine learning has been effectively adopted. Unfortunately, the changing behavior during the time may lead to higher classification errors than those measured in the test phase. In this study, we demonstrate that the existing machine learning techniques applied for network traffic classification fail when facing the characteristics of real-world environments. The experiments analyzed more than 30 TB of data spanning 10 years of real network traffic and 9 intrusion detection datasets. Besides the analysis, we define a set of guidelines to build reliable application of machine learning for network traffic classification, which may guide future research and ensure the reliability of machine learning model deployment in production environments.

Introduction

A Network-Based Intrusion Detection System (NIDS) is a security mechanism that monitors and identifies security attacks and violations occurring in Internet of Things (IoT) devices' network [1, 2]. In general, the detection scheme of proposed NIDS for IoT is implemented through either *misuse-based* or *behavior-based* techniques [3]. On the one hand, *misuse-based* techniques perform the detection task by exploring the current understanding of how the attack behaves, i.e., by leveraging a database of previously known attack signatures for network-traffic pattern matching. On the other hand, *behavior-based* techniques aim at building a model of expected system behavior to alert suspicious activities [4]. As a consequence, in contrast to misuse-based approaches, behavior-based techniques are expected to detect new patterns of attacks [5].

Over the last decades, many scientific contributions have proposed highly accurate behavior-based schemes for intrusion detection, typically implemented through Machine Learning (ML) techniques as a pattern recognition scheme [4]. To this aim, it is essential to collect huge amounts of normal and anomalous network patterns for the model-building task. The built ML model can then be used to identify intrusions according to their similarity to the previously evaluated normal or anomalous behavior [3]. Indeed, intrusions can only be detected if their behavior is similar to a previous attack learned by the system. Despite the promising research results reported at the test phase, ML-based detection techniques for NIDSs are rarely used in production environments [4]. This is due to the fact that the accuracy rates achieved during the test phase are rarely maintained in production settings. In practice, for reliable ML-based NIDSs deployment, the test phase conditions must present realistic production environment properties, enabling the evaluation of more realistic accuracy rates [6].

Eduardo K. Viegas is with Pontifícia Universidade Católica do Paraná (PUCPR), Brazil and Technology Innovation Institute, Secure Systems Researcher Center, Abu Dhabi, United Arab Emirates.
Altair O. Santin is with Pontifícia Universidade Católica do Paraná (PUCPR), Brazil.
Pietro Tedeschi is with Technology Innovation Institute, Autonomous Robotics Researcher Center, Abu Dhabi, United Arab Emirates.

Digital Object Identifier: 10.1109/IOTM.001.2300106

The traditional intrusion detection model was initially designed for host-based intrusion detection, not considering a networked deployment setting. However, the literature mistakenly borrows the assumptions of host-based intrusion detection to the network context, leading to inaccurate results [4]. Similarity-based detection, as performed by the majority of current ML-based techniques, introduces several challenges for their deployment in production environments. Due to the discovery of new attacks, the provision of new services, or even changes in the network traffic link that can modify the common network patterns, network traffic behavior is highly variable and changes over time. As a result, the characteristics of real network environments must be considered during the test phase. However, the literature often neglects them, and traditional ML evaluation takes place [6].

Contributions. In this article, we introduce and experimentally evaluate the aspects that make NIDS environments challenging for ML-based techniques. Our findings show that the accuracy results reported by the existing techniques from the literature are unreliable when facing production environment properties. To address this problem, we present a set of guidelines for a reliable evaluation of ML-based NIDSs.

Preliminaries

This section describes the typical ML workflow for NIDSs, and clarifies how networked environments pose a challenge for their deployment.

Machine Learning for Network-Based Intrusion Detection

The intrusion detection task of behavior-based NIDSs typically relies on ML techniques implemented as a pattern recognition scheme [7]. To achieve this goal, we adopt a three-phase process, namely *training*, *validation*, and *testing*. In the *training* phase, a ML model is built through a resource-intensive model training process. The model-building task evaluates the behavior of a training dataset composed of huge amounts of normal and abnormal network traffic, enabling the extraction of a reliable behavioral ML model. Next, the quality of the model is estimated through a *validation* phase, such as performing feature selection and model hyper-parameter fine-tuning. Finally, the accuracy of the model is assessed during the *testing* phase, which is then assumed to be preserved during production usage. In practice, each phase requires a unique dataset, which can be used either

2576-3180/23/$25.00 © 2023 IEEE

during the model *training*, *validation*, or *testing* phases. During the last phase, the designed ML-based NIDS technique is evaluated by measuring its accuracy rates with several metrics, such as False-Positive (FP), False-Negative (FN), True-Positive (TP), and True-Negative (TN):

- **FP Rate.** Ratio of normal network events misclassified as anomalous.
- **FN Rate.** Ratio of anomalous network events misclassified as normal.
- **TP Rate.** Ratio of anomalous events correctly classified.
- **TN Rate.** Ratio of normal events correctly classified.

Therefore, when FP and FN rates are low, and TP and TN rates are high, the model can be reliably deployed in production.

Challenges of ML-Based Detection Approaches for NIDS

The behavior of a network traffic changes quite often, owing to either new services or even the discovery of new security threats [8]. Further, the non-stationary and highly variable behavior of network traffic poses a plethora of challenges that are not considered in the ML evaluation process. Indeed, the design of a reliable and responsive ML solution able to withstand changes in the deployed environment is considered a challenging task that can often only be addressed by conducting frequent and difficult ML model updates.

It is worth noting that the reliability of the ML models over time in intrusion detection is not considered often. In contrast, model lifespan is not taken into account, and it is usually considered an orthogonal problem in the scientific literature [9]. In practice, the time taken between for detection system's design and training until its deployment may render the system outdated before the deployment in production. Unfortunately, periodic model updates are not always feasible, as they require the constant monitoring of the environment, as well as the collection, and labeling of the network events, typically achieved with human intervention. Thus, ensuring the ML model reliability for longer periods of time is a crucial feature, as model updates are neither easy nor cheap to be conducted [10].

Moreover, the quality of ML-based techniques are strongly correlated with having a properly built training dataset that reflects the network behavior of the production environment. Indeed, providing a real-time responsive solution to the highly variable network behaviors is still a challenging task [6]. For example, the provision of new services or the request for new content and the discovery of new security threats are typical characteristics that alter the behavior of a network [11]. In such a situation, building a training dataset with all possible network behavior variations of the production environment is quite difficult for the current proposed ML schemes. As a result, besides having a properly built training dataset, the ML model must also be able to generalize the behavior from it by detecting new services, services' content, and attacks regardless of their presence in the training data [6].

Even if the ML model can generalize the behavior from the sample input packets in the training dataset and it is periodically updated, the network settings are also variable. Typical examples are the network throughput variation, the network topology, or the presence of new hosts in the network. Thus, the ML model must also be able to account for its generalization capacity from the network settings perspective, demanding it to operate regardless of the environment in which it was built [10].

In contrast, current datasets commonly used to evaluate the network properties are decades old and, hence, intrinsically outdated [7]. Owing to privacy concerns, training datasets are typically built over synthetic data obtained from traffic generation tools. However, even if the training dataset is properly built, researchers often fail by considering the network data static [5]. Notwithstanding, detection mechanisms must generalize the behavior from the training dataset instead of simply showing reasonable accuracies during the test phase [4]. Otherwise, the ML-based technique will be unreliable when deployed in the production environment [6].

The available datasets in the literature for NIDSs do not provide network datasets that span long time periods. Generally, a network dataset is split into training, validation, and testing sets without considering the time of occurrence.

Modeling a Moving Target

Current ML-based techniques for NIDSs assume: (i) a constant-updated deployed system over time or (ii) the application of a systematic model update procedure. Since the evaluation of the model lifespan is usually not taken into account by proposed schemes, the impact of changes in the network traffic behavior over time on the classifier's performance remains unknown.

In practice, there is still a lack of knowledge on how the network traffic behavior changes as time passes, and also how it can affect the accuracy of the model. Despite the lack of the model's lifespan consideration, the requirements for the systematic model update procedures are a crucial gap in the scientific literature. In light of this, this section highlights how the evolving behavior of network traffic impacts the performance of a ML model for NIDSs.

Decade-Long Network Traffic

The available datasets in the literature for NIDSs do not provide network datasets that span long time periods. Generally, a network dataset is split into training, validation, and testing sets without considering the time of occurrence. Therefore, evaluating the system's reliability and accounting for the network traffic behavior changes over time by leveraging these datasets is not feasible. Indeed, to achieve good performance, the network dataset for this purpose must be obtained from a real deployed network environment. Unfortunately, the labeling process of real-world network data (i.e., tagging network traffic as either anomalous or normal) required by pattern recognition schemes is an unfeasible task that often requires expert assistance.

To address such a shortcoming, we adopt an extended version of the MAWIFlow dataset [9] to evaluate the accuracy of ML-based NIDSs over time. The new dataset comprises 10 years of data collected from publicly available real-world network traffic. In detail, it is based on the network flows extracted from the MAWI network packets traces [12], collected daily in a 15-minute interval from a communication link between Japan and the USA. The labeling of records is performed through MAWILab [13], which associates a label to each anomalous daily event (network flows) from MAWI through a combination of several state-of-the-art unsupervised anomaly detectors. In this work, we consider the network traffic captured in a 10-years range, spanning from 2007 to 2016. Furthermore, the MAWIFlow dataset is built using the BigFlow [9] feature extraction tool, which extracted 62 network flow features and comprised over 30 TB of data and over 28 billion of network flows.

Accuracy Over Time

The first MAWIFlow experiment evaluates the classification accuracy of traditional ML-based intrusion detection techniques over time. Current approaches in the literature rely on deep learning or shallow classifiers. On the one hand, deep learning techniques typically yield high accuracies, while usually demanding more computational resources and providing lower generalization. On the other hand, shallow classifiers are usually better suited for resource-constrained devices and are typically used for IoT applications. As a result, for this experiment, we use the Random Forest (RF) classifier, a widely used ML clas-

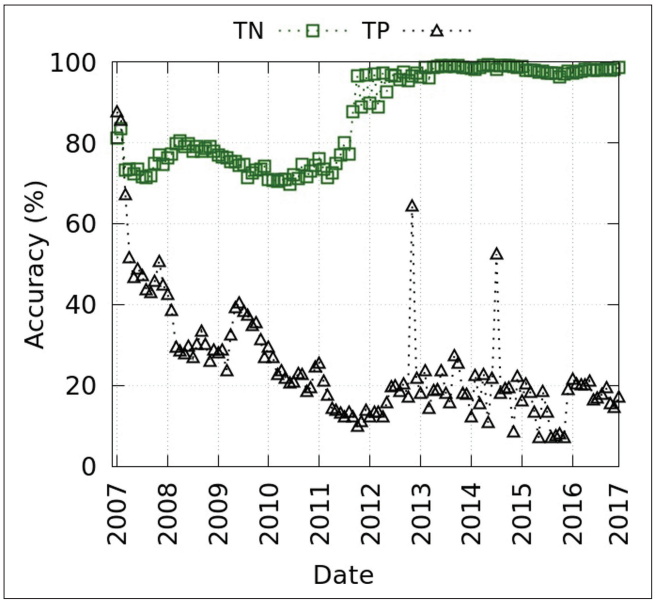

FIGURE 1. Monthly accuracy over 10 years of real network data (MAWIFlow dataset) using a RF classifier. The accuracy significantly decreases within a few months after the training period.

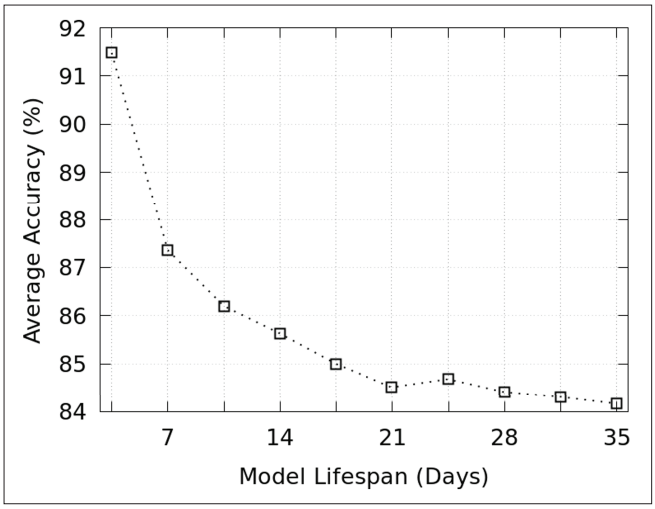

FIGURE 2. Relation between the frequency of model updates and average accuracy throughout 2016 (MAWIFlow dataset) using a random forest classifier. Average accuracy increases if the model's lifespan (interval between model updates) is less than 21 days.

sifier on the IoT context due to its promising reported results, lightweight inference implementation, and the accuracy evaluation [9, 14]. The RF is implemented with 100 decision trees as a base learner, trained using the first month of MAWIFlow (January 2007) data, and evaluated throughout the remaining 10 years. It is important to note that several other well-known classifiers were also evaluated and achieved similar results.

Figure 1 shows the obtained monthly TP and TN accuracy rates. It is worth noticing that there is a significant accuracy decrease in both TN and TP rates in the months following the training period. For instance, five months later, the TP rate has already decreased by half (compared with TP during the model building). Notwithstanding, only one month after the training, the TP rate has already decreased by 20 percent. Over time, the TP rates significantly decrease, reaching only 7 percent in May 2015, an 80 percent decrease from the training period. In contrast, TN rates slightly decrease until the end of 2011 and increase after this period. Such a behavior can be explained by a probing performed in the entire Internet Protocol version 4 (IPv4) space at the MAWI transmission link, which began in September 2011. As a result, the evaluated network traffic behavior significantly changed after that period, which consequently impacted the ML model accuracy.

Model's Lifespan

Behavior changes in the network traffic over time impact the accuracy of the deployed ML model by affecting its lifespan (i.e., the period when the ML model is reliable) as time passes. In practice, the accuracy obtained at the test phase is not preserved in production over time, demanding model updates to be conducted. Therefore, the second experiment aims to evaluate the trade-off between the model's lifespan and its accuracy. The goal is to measure how the periodicity of model updates impacts the model's accuracy over time.

Figure 2 shows the average yearly accuracy and the model's lifespan trade-off. We measure the average accuracy as the average of both TP and TN values. It is possible to note a direct relationship between the model's lifespan and average accuracy. For instance, when weekly model updates are performed, the average accuracy reaches 87 percent in contrast to 91 percent obtained with the two weekly updated counterparts. In addition, the model's update frequency does not significantly impact the model's accuracy after three weeks of the model's lifespan, showing similar accuracy rates after this period.

Discussion

The ML model update task for NIDSs is not often considered in related works. In general, the literature assumes that periodic model updates will be conducted, without considering the methodology to perform this task. In such a case, due to the evolving behavior of networked environments, the ML model can become outdated only a few days after the training period.

In this study, we have experimentally evaluated the impact of changes in network traffic over time on the ML model performance. Leveraging 10 years of real-network data, we have shown that ML model accuracy rates significantly decrease weeks after the training period. As a result, to deploy ML-based intrusion detection schemes, the model's lifespan must be taken into consideration. We have evaluated the impact of the model's lifespan on its accuracy, showing a direct relationship between shorter model lifespans and higher system accuracies. However, according to the experiments (Fig. 2), the most important impact occurs before three weeks of the model's lifespan. After this period, the accuracy rates are not significantly affected.

The evaluation results have shown that current techniques for ML-based NIDSs cannot cope with changes in network traffic behavior. However, the lower the periodicity of model updates, the more complex the detection system for a real-world environment. The ML model update task is a high-cost process that requires event storage, labeling, model retraining, and evaluation. Therefore, it becomes crucial that ML models designed for NIDSs have a longer lifespan. Otherwise, even if the ML model has high accuracy, the need for constant model updates may render the proposed approach unfeasible for reliable deployment.

Modeling a Highly Variable Target

The behavior of network traffic in IoT environments is highly dynamic, posing a significant challenge to design ML-based techniques. This section analyzes how the high variability of network traffic impacts the performance of ML-based NIDS.

Fine-Grained Intrusion Dataset

Over the last decades, several contributions have proposed new datasets to build and evaluate NIDSs [7]. In general, the proposed techniques follow a coarse-grained rationale that does not individually consider the detection rates of services,

services' content, and security attacks. In contrast, to enable their deployment in production environments, the ML model needs to detect network events according to each possible network behavior variation. For example, network operators must be able to define the accuracy impact of their detection system when a new service is deployed. Indeed, they can take countermeasures according to the expected accuracy impact of a new behavior on the ML model, like a model update, and in the meanwhile, alert the presence of newly deployed service. Unfortunately, proposed ML-based NIDSs approaches do not measure their accuracies in a fine-grained rationale.

On the contrary, most literature relies on outdated datasets with several known design flaws [4]. As a result, in recent years, significant research efforts have been conducted to improve the quality of the used intrusion datasets. For example, researchers may augment the training data, use multiple datasets, and even resort to simulation environments [15]. Unfortunately, providing a realistic intrusion dataset is still an open challenge. This shortcoming makes proposed techniques that perform well on a specific dataset, and they are inefficient when adopted in normal production environments due to (i) dataset design flaws or (ii) the missing evaluation with real up-to-date traffic.

Therefore, in contrast to pursuing the most accurate ML model for a specific dataset, it is essential to design a reliable model that performs when facing the changing network traffic in the industrial environment. In such a context, we use Fine-Grained Dataset (FGD), which includes a series of network behaviors (i.e., packets) that can occur over several possible network variations that ML models must process.

The established properties are defined based on the highly variable network traffic behavior. In practice, FGD comprises the network traffic variability that can occur because of the design limitations of the intrusion datasets, such as variations in the service behavior and even the occurrence of a new service or attack. The dataset enables the proper evaluation of ML models according to each possible network behavior variation, taking into account the service and its content as well as the attacker. For each dataset setting, we consider three situations, namely *known* (during the training), *similar* (behavior similar to training data), and *new* (not available during the training time). The similarity score is defined according to the network administrator's discretion.

To provide fine-grained data control, we used the proposed FGD in [6] in a controlled environment. FGD comprises more than 10 network-based attacks, ranging from probing to service vulnerability exploitation, and 6 service protocols, which were used to generate normal traffic, resulting in 8 datasets in total. The client's and attacker's behaviors significantly vary during the network environment monitoring period [6].

Fine-Grained Accuracy

We use FGD to evaluate how the ML model behaves under real-world network environment settings. To achieve this goal, we also make use of the same RF classifier configuration evaluated previously. In this case, the selected classifier is trained using a dataset containing the known events of FGD. Further, we evaluated the obtained model with the dataset containing similar and new related events. The final goal is to measure the model reliability under every possible scenario variation of the production environment.

Figure 3 shows the FP and FN rates for the RF classifier while being evaluated with several production environment situations. We note that the selected classifier can provide similar detection rates when varying the service content (Fig. 3a), showing that ML models are robust to similar and new services' content variation. In contrast, a significant error increase can be noted when new services or attacks are evaluated (Figs. 3b and 3c), increasing the FP rate to 13 percent and the FN rate to 68 percent respectively. As a result, the ML must be rebuilt to address the new types of services and new categories of attacks.

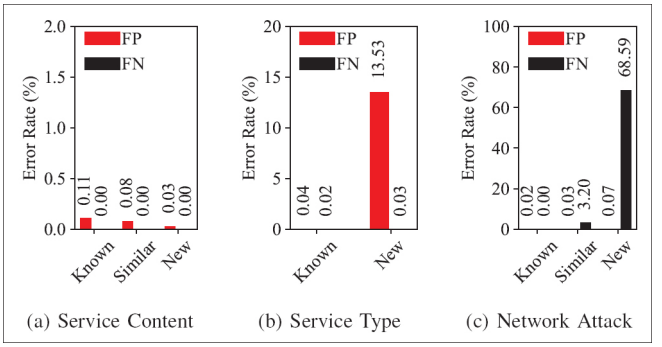

FIGURE 3. Fine-grained evaluation of a random forest classifier under several production environment characteristics. Accuracy rates significantly decrease according to the evaluation criteria, the classifier is trained on the *Known* environment.

Discussion

Current evaluation techniques for ML-based to NIDSs do not provide robust detection accuracies concerning the performance for detecting network traffic behavior variations. Although the literature assumes that ML-based techniques can detect new kinds of network behaviors, our experiments show that proposed schemes cannot cope with the detection of new services nor with new categories of attacks (Figs. 3b and 3c). In contrast to the NIDS literature assumption, ML-based schemes perform the detection task following a similarity-matching rationale, thus, failing to detect new types of behaviors.

Consequently, when ML-based NIDSs are deployed in production environments, they cannot reach the same level of accuracies measured during the test phase, which often only includes a restrained set of network behavior variations. It results that the accuracy impact is also affected by an inadequate detection scheme evaluation. Introducing the highly variable nature of network traffic behavior during the test phase becomes necessary to handle such a challenge. Indeed, addressing this challenge will allow the operator to adopt the proper countermeasure, such as disabling false service alarms or even performing the model update to address unknown behaviors.

Modeling a General Environment

The classification performance of ML-based NIDSs is affected from the network environment in which it is evaluated. Most of proposed techniques can provide high accuracies in almost any given dataset (Fig. 3, *Known*). However, such schemes often consider a single and static set of environment settings, such as the number of hosts, network topology, and network link bandwidth. In practice, designed ML-based schemes are often only evaluated with respect to how the model performs in a specific environment, neglecting how it would operate if the environment characteristic changes. This section evaluates the generalization capacity of ML-based NIDSs with respect to their performance regardless of the environment in which it was designed.

NIDS Generalization Capabilities

The generalization capacity of ML-based NIDSs is evaluated through the same RF classifier used previously according to its performance in two separate datasets. The selected RF classifier is trained considering the FGD *Known Service Content* dataset (Fig. 3a). At the same time, its accuracy is also evaluated in the well-known DARPA1998 dataset, following a cross-validation rationale. To ensure that we adequately evaluate the model generalization capabilities, we select only the attacks from DARPA1998 that mutually occur on both datasets.

Table 1 shows the accuracy rates attained at each dataset. It is worth noticing that there is a significant increase in the error rate when the model is evaluated in a different environment. Both the

Environment	Accuracy	False-Positive	False-Negative
FGD (Known)	99.99	0.11	0.00
DARPA1998	87.27	14.10	9.93

TABLE 1. Random forest generalization performance. The classifier is trained with the FGD Known Service Content dataset.

FP and FN rates are increased when compared with those obtained for the FGD dataset, i.e., the environment used for the model training. As a result, the obtained model cannot adequately generalize the characteristics learned from the training dataset, demanding a model update for each environment where it will operate.

Discussion

To ensure the reliable deployment of ML-based NIDS, the chosen techniques must pave the way for ML model portability. Unfortunately, current literature approaches do not consider the model's generalization capacity, as authors often assume that their ML model will reach similar accuracy rates when deployed in a different environment. As shown, pattern recognition schemes require that the training environment is similar to the environment where the final model deployment will occur. Indeed, if the ML model is adopted in a different environment, the accuracy rate significantly decreases unless model updates are conducted, a task that demands additional time and expert assistance. To effectively enable their use in production, it becomes crucial that the designed ML-based NIDSs can generalize the model by resorting to a suitable training dataset.

Guidelines for ML-Based NIDS Development

Networked environments present many challenges to ML-based techniques when compared to other fields where it has been successfully applied. Frequent network traffic changes negatively impact current techniques' performance, a situation in which the adequate evaluation of ML-based NIDSs becomes a must. We present a set of evaluation guidelines to address this challenge to ensure reliable NIDSs.

- **Detection of known behaviors.** To deploy a ML-based NIDS operators must have prior knowledge of the accuracy rates under a set of known constraints. To this aim, it is important to estimate the detection rate of known behaviors, including normal and anomalous events. Over the last few years, this kind of evaluation has been a standard practice in the literature.
- **Detection of similar behaviors.** Building an intrusion dataset with all the behavior variations expected in production environments is challenging. Yet, network operators must be able to evaluate how their system will perform when facing an unexpected service or attack behavior variation. In practice, evaluating such a property is not easily feasible; for example, the operator may decrease the available behavior in the training dataset while using the extra data for evaluation purposes. As a result, the administrator can assess in advance with a reasonable level of confidence whether a highly variable service or attack behavior will imply in model updates.
- **Detection of new behaviors.** A common assumption in the literature is that their designed ML-based NIDS can detect new categories of attacks or services. However, as shown in this article, a new behavior can significantly decrease the accuracy rates of ML-based NIDSs, demanding that operators evaluate whether a new service or attack occurring in the passing network traffic requires a model update to be conducted. Indeed, to evaluate the detection rate of new security attacks and new services the administrator can build a dataset with new behaviors when compared to the training data. Thus, the network operator can assess it through the built dataset and assume that the measured rates will be evidenced in production settings.
- **Detection over time.** The behavior of networked environments changes over time, demanding that ML schemes are updated to account for those changes. To take corresponding actions, network administrators must have a clear understanding of how the network behavior changes impact the NIDS accuracy. For instance, administrators may collect environment data, label them, and periodically evaluate the performance of the ML model. As a result, the operator can ensure that the ML model has the expected level of accuracy, even after months since the training period.
- **Detection in new environments.** Existing ML-based NIDS assumes that the system will be deployed in the same training environment. In contrast, for production environments, the designed schemes must be deployed as a tool ready for deployment, regardless of their current operating environment. To this aim, network operators must be able to establish how the system will perform in their own environment. For example, ML-based NIDSs should ideally maintain their detection rates under different environments to increase reliability.

The required properties to achieve reliable intrusion detection have been aimed at by related works in recent years. The evaluation of the detection of similar and new behaviors has been explored by researchers that make use of a broader range of intrusion datasets, or even data augmentation techniques [15]. Similarly, the detection reliability as time passes has been explored by related works proposing reinforcement learning [9], or even active learning schemes. As a result, the challenges that are making the deployment of ML-based intrusion detection schemes on real-world environments are a target of research in the field. However, proposed schemes must take into consideration all of the aforementioned guidelines to enable reliable intrusion detection, which can pave the way toward their scheme deployment in production environments.

Conclusion

Over the last years, designed ML-based techniques for NIDSs have failed to provide the expected level of detection reliability for production deployment. This study has experimentally highlighted that each possible network behavior variation impacts the accuracy of the designed techniques. To address this issue, we presented a set of guidelines for a realistic evaluation of ML-based techniques for NIDSs, considering the challenges related to the network traffic in production environments. In conclusion, we highlight research challenges, directions, and countermeasures that Academia and Industry must address to progress in developing secure machine learning-based intrusion detection systems.

Acknowledgment

This work was partially sponsored by Brazilian National Council for Scientific and Technological Development (CNPq) grant n. 304990/2021-3, and by Technology Innovation Institute, Abu Dhabi, EAU. The findings reported herein are solely responsibility of the authors.

References

[1] P. Tedeschi, S. Bakiras, and R. Di Pietro, "IoTrace: A Flexible, Efficient, and Privacy-Preserving IoT-Enabled Architecture for Contact Tracing," *IEEE Commun. Mag.*, vol. 59, no. 6, pp. 82–88, 2021.

[2] V. Ravi, R. Chaganti, and M. Alazab, "Deep Learning Feature Fusion Approach for an Intrusion Detection System in SDN-Based IoT Networks," *IEEE Internet of Things Mag.*, vol. 5, no. 2, pp. 24–29, Jun. 2022.

[3] D. Arp *et al.*, "Dos and Don'ts of Machine Learning in Computer Security," *USENIX Security Symp.*, 2022.

[4] E. M. Campos *et al.*, "Evaluating Federated Learning for Intrusion Detection in Internet of Things: Review and Challenges," *Computer Networks*, vol. 203, Feb. 2022, p. 108661.

[5] R. Zhao *et al.*, "Semisupervised Federated-Learning-Based Intrusion Detection Method for Internet of Things," *IEEE Internet of Things J.*, vol. 10, no. 10, 2023, pp. 8645–57.

[6] 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, 2017, pp. 200–16.

[7] B. Molina-Coronado *et al.*, "Survey of Network Intrusion Detection Methods From the Perspective of the Knowledge Discovery in Databases Process," *IEEE Trans. Network and Service Management*, vol. 17, no. 4, Dec. 2020, pp. 2451–79.

[8] Z. Yang *et al.*, "A Systematic Literature Review of Methods and Datasets for Anomaly-Based Network Intrusion Detection," *Computers & Security*, vol. 116, May 2022, p. 102675
[9] R. R. dos Santos *et al.*, "Reinforcement Learning for Intrusion Detection: More Model Longness and Fewer Updates," *IEEE Trans. Network and Service Management*, 2023, pp. 1–17.
[10] Y. Mirsky *et al.*, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," *Proc. 2018 Network and Distributed System Security Symp. Internet Society*, 2018.
[11] Y. Wu *et al.*, "Paradise: Real-Time, Generalized, and Distributed Provenance-Based Intrusion Detection," *IEEE Trans. Dependable and Secure Computing*, 2022, pp. 1–1.
[12] MAWI, "MAWI Working Group Traffic Archive — Samplepoint F," 2023; https://mawi.wide.ad.jp/mawi/
[13] R. Fontugne *et al.*, "MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking," *Proc. 6th Int'l. Conf. Emerging Networking EXperiments and Technologies (CoNEXT)*, 2010.
[14] H. Bangui and B. Buhnova, "Lightweight Intrusion Detection for Edge Computing Networks Using Deep Forest and Bio-Inspired Algorithms," *Computers and Electrical Engineering*, vol. 100, May 2022, p. 107901.
[15] V. Kumar and D. Sinha, "Synthetic Attack Data Generation Model Applying Generative Adversarial Network for Intrusion Detection," *Computers & Security*, vol. 125, 2023, p. 103054.

Biographies

Eduardo K. Viegas (eduardo.viegas@ppgia.pucpr.br) received the B.S. degree in computer science in 2013, the M.Sc. degree in computer science in 2016 from PUCPR, and the Ph.D. degree from PUCPR in 2018. His research interests include machine learning, network analytics and computer security.

Altair O. Santin [M] (santin@ppgia.pucpr.br) received the B.S. degree in Computer Engineering from the PUCPR in 1992, the MSc degree from UTFPR in 1996, and the Ph.D. degree from UFSC in 2004. He is a full professor of Graduate Program in Computer Science (PPGIa) and head of Security & Privacy Lab (SecPLab) at PUCPR. He is a member of the ACM and the Brazilian Computer Society.

Pietro Tedeschi [M] (pietro.tedeschi@tii.ae) is a Senior Security Researcher at Technology Innovation Institute, ARRC, Abu Dhabi, UAE, from January 2022. He obtained his Ph.D. in Computer Science and Engineering from Hamad Bin Khalifa University, Doha, Qatar, in December 2021, and the Master's degree with honors in Computer Engineering at Politecnico di Bari, Italy, in February 2017. From 2017 to 2018, he worked as Security Researcher at Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Italy, for the EU H2020 SymbIoTe project. He is serving in the TPC of several conferences. His major research interests include security and privacy in UAVs, IoT, Cyber-Physical-Systems, Applied Cryptography.