Machine Learning in Intrusion Detection: An Operational Perspective

Martin Husák*, Darshan Manoj†, Priyanka Kumar‡
*Institute of Computer Science, Masaryk University, Czech Republic
†Thomas Lord Department of Computer Science, University of Southern California, CA, USA
‡Department of Computer Science, The University of Texas Permian Basin, TX, USA
husakm@ics.muni.cz, dmanoj@usc.edu, kumar_p@utpb.edu

***Abstract*—Machine learning has become a prevalent approach in research on intrusion detection with enormous number of research publications on the topic, but its adoption by cybersecurity practitioners is falling behind. Recently, researchers conducted critical and pragmatic assessment of the capabilities of machine learning in this task and identified fundamental issues preventing wider application and easy use in practice. In this paper, we approach the topic from the perspective of network security management, focusing on the issues of compatibility with existing monitoring and security infrastructures, computational complexity, ease of use, and required skills of the operators. The research in machine learning-based intrusion detection strongly favors machine learning metrics (e.g., precision and accuracy) over any other outcome, including performance and usability, for which we have no actual results due to very low number of prototypes, implementations, and field studies. Moreover, there are very limited options of recognizing which type of attack was detected, which remains a strong advantage of traditional signature-based intrusion detection systems.**

***Index Terms*—Intrusion detection, machine learning, network security management, security operations**

I. Introduction

Machine learning (ML) has been applied in many fields with often a great success. It is no surprise that cybersecurity researchers and practitioners would like to leverage it in their field, and that the number of research publications on interesting applications of machine learning in cybersecurity were written, published, and discussed [1]. One of the prominent cybersecurity applications of machine learning is the intrusion detection, i.e., a task of detecting malicious behavior in network traffic or system logs. The research efforts are running for two decades already, and hundreds of published research papers claim high accuracy supported by replicable laboratory experiments with publicly available datasets [2], [3]. However, such methods do not seem to be applied and widely used by practitioners. We may be justly asking: *why is machine learning not widely used in operational intrusion detection?*

We are currently seeing efforts to re-think this research direction, clearly identify its issues [4], [5], and set recommendations and requirements [6], [7], so that the research can become applicable. The issues of low quality of the datasets (e.g., limited numbers of attack samples, rapid obsolescence, artificiality, imbalance, or large volume) are actively discussed and approached by researchers [8]. The same applied to best practices in training the ML models. So far, we seem to be on a good track to delivering high quality datasets and training accurate models [9]. However, there are far more issues potentially preventing the application of ML in intrusion detection. If a novel, ML-based, intrusion detection system (IDS) is to be adopted, it should surpass the existing tools in key performance indicators or provide novel functionality, while not losing any existing capabilities or at least provide sensible trade-offs.

In this paper, we formulated a number of questions on operational aspects ML-based intrusion detection systems (further referred to as ML-IDS) and discuss the implications for future research and development. The research was motivated by consulting experts in cybersecurity and ML, both from industry and academia, and finding common themes in the operational issues or critique of ML-IDS among the respondents. We focus on the most critical issues found in the literature and do not claim to provide satisfying answers to all the questions or covering the full scope of the topic. The aim of this paper is to raise awareness of the issues and prepare for an exhaustive survey of literature and implementations or formalized interviews with experts.

The remainder of this paper is structured as follows. In Section II, we summarize the background and related work on the topic, focusing on papers providing critical assessment of ML in intrusion detection. In Section III, we formulate number of questions that should be answered before ML-IDS is to be deployed or even implemented. We follow this question-answer scheme in Section IV, where we discuss the applicability of ML-IDS in its current state and possible directions for future development. Section V concludes the paper.

II. Background and Related Work

The use of machine learning for intrusion detection was critically assessed several times. The first such assessment dates back to 2010, when Sommer and Paxson [10] noticed that, despite extensive research efforts, ML-based intrusion detection is rarely employed in operational settings. They were the first to assess the differences between intrusion detection and other fields, where machine learning proved helpful. Intrusion detection turned out to be a significantly harder problem and, thus, the authors provided a set of guidelines

978-3-903176-66-9 ©2024 IFIP

for future research. However, the situation seems to remain unchanged despite the high citation of the paper.

A decade later, more researchers attempted to assess the benefits and pitfalls of machine learning in cybersecurity, including intrusion detection. Pragmatic assessment and recommendations for using machine learning in cyber security in general were proposed by Apruzzese et al. [11], [12], Arp et al. [7], [4], and Ceshin et al. [5]. Apruzzese et al. also wrote an assessment on machine learning for intrusion detection in particular [6]. Corsini and Yang [13] explored whether the out-of-distribution techniques are suitable for intrusion detection. Recently, Munner et al. [14] proposed a critical review of AI-based intrusion detection, but covered only the AI aspects.

The actual research works on machine learning in intrusion detection are numerous to the point it is becoming unfeasible to review them all. The number of research papers on this topic exceeds hundreds and even thousands, depending on search words, according to searches in popular research libraries (e.g., Google Scholar, Scopus, IEEExplore, ACM Digital Library). Searching for surveys and literature reviews does not help, either; there are over 30 such papers published in 2023 only. We recommend the survey by Buczak and Guven [1], which despite being older, serves as an excellent introduction into the field. Recent examples include the works of Kilincer et al. [2], Chou et al. [3]; recent surveys also focused on particular application domain, such as IoT [15] or smart grid [16].

While there are thousands of research papers on the topic of ML-IDS, we were also interested in the number of existing implementations or prototypes. We searched GitHub for projects that apply machine learning for intrusion detection and categorized them by the programming language used. The results can be seen in Fig. 1. We can see a dominance of Python, which is widely used in the domain of machine learning, and while it can be used for implementing prototypes, most of the repositories contain only scripts for training and evaluating models on datasets. Projects using Jupyter Notebooks or MATLAB also purely laboratory experiments. We also encountered numerous projects using only TeX or HTML, containing only research papers or documentation of ML-IDS. Not even the repositories using Java or C++ contained technologically mature projects that could be deployed in operational environment. Vast majority of the projects only allowed for processing of datasets, not having any capabilities to process live network traffic or raw traffic samples.

One of the outstanding open-source projects is Slips (Stratosphere Linux Intrusion Prevention System[1]), an open-source intrusion prevention system crafted in Python at the Czech Technical University in Prague. We consider Slips as an example of a good practice in research on ML-IDS. It is a versatile prototype that allows for processing various inputs from raw network traffic in real time to PCAP files and network flows. It uses a combination of multiple ML models trained to detect various malicious activities, threat intelligence feeds, heuristics, and extensively trained thresholds that raise

[1]https://github.com/stratosphereips/StratosphereLinuxIPS

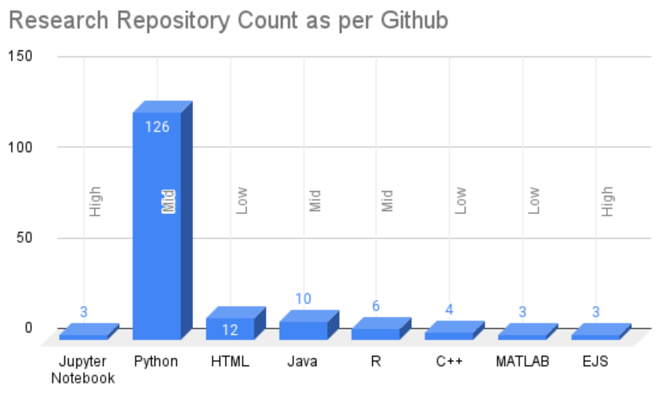

Fig. 1. Number of ML-IDS projects on GitHub per programming language.

alerts when enough evidence is accumulated. Even though it is still a prototype and a research project, we consider Slips as a leading open-source implementation of ML-IDS, since we could not find any other project of similar technological maturity.

Numerous commercial products promise the use of AI/ML for intrusion detection, but in most cases, they employ AI/ML for anomaly detection (see Section III-A) or as an enhanced XDR/NDR solution, not for specifically detecting the intrusions as a replacement of IDS. Moreover, it is often hard to obtain information about the employed ML methods or their application. The examples include Vectra[2], Darktrace[3], or ExtraHop[4].

III. Issues of ML-based Intrusion Detection

In this section, we formulate several questions on ML-IDS and its application in cybersecurity operations. The questions are followed by a brief discussion of the stated issue. We do not aim at providing a comprehensive analysis of applicability of ML-IDS, but rather pinpoint the most critical points from the operational perspective.

A. What is the use case for ML-based IDS?

Naturally, the use case for IDS is to detect attacks in network traffic. However, attack detection covers various approaches and goals. Two major classifications are used. First, the IDS can be host-based or network-based. Host-based IDS (HIDS) resides on a single host and uses system logs, system calls, or analysis of the network traffic of the single host to detect attacks. Network-based IDS (NIDS) uses network traffic data from one or more observation points (e.g., router or a dedicated probe on a link) to detect anomalies in the network traffic of multiple hosts simultaneously. Even though both can process network traffic, the scope and capabilities of each type differs

[2]https://www.vectra.ai/
[3]https://darktrace.com/
[4]https://www.extrahop.com/

vastly, so it important to specify the use case upfront and train the ML-IDS accordingly.

Second, we distinguish intrusion detection systems (IDS) and anomaly detection systems (ADS). A traditional IDS is using signatures of cyber attacks and detects the signatures in the source data, thus recognizing known attacks. Signature-based IDS cannot detect unknown attacks (attacks, for which there is no known signature), but if it detects a signature, it is clear what kind of attack it is and how to defend against it. In ML-IDS, the signature database is replaced by a model trained on (artificial or real-world) attack samples. In contrary, the ADS detects anomalies in network traffic, i.e., deviations from common traffic patterns. Even the unknown attacks can be detected as anomalies, but an anomaly also requires further analysis (typically by a human operator), which can be time-consuming and complicates automated response. ML methods are widely used in ADS, but the model is trained on benign network traffic, not on the attack samples. It is important to specify whether we talk about ML-IDS or ML-ADS. The situation is complicated by the fact that the terms are often used interchangeably and that the ML-IDS are often more similar to ADS in terms of binary classification (benign x anomaly, benign x attack) and lack of explainability.

B. What data are on the input?

Surprisingly, many papers on ML-IDS do not specify what data are they working with. The authors simply reference a dataset that was used for evaluation, often without discussing what type of data would be used in operations or in a given use case. In network security, we may consider multiple types of data, all with their benefits and drawbacks. In a simple scenario, the IDS may process network packets as they arrive to the vantage point. Packet features can be extracted and forwarded as an input to the classifier. This setup may be enough for low-throughput networks, IoT networks, and so on. However, in cloud environments or when processing network traffic at a high-speed link, processing each packet individually might be heavily resource-intensive. A typical solution is using aggregation into network flows, such as NetFlow/IPFIX. The existing NetFlow/IPFIX technologies are *de facto* standard for high-speed network measurements. Some ML-IDS proposals consider network flows as an input, which facilitates their potential application, which is highly appreciated – the IDS can then directly access the already collected and pre-processed data. In other cases, there might be a need to implement a custom packet parser, flow aggregator, or feature extractor (see the following questions), which may take months or development and optimization to achieve similar performance and precision as existing approaches.

C. Are they data processed in batch or in a stream?

Another important aspect to consider is whether the data are processed in batches or as a stream. Either choice may have no effect on ML performance (e.g., accuracy, precision), unless the classification leverages any contextual information derived from history of network traffic. Nevertheless, even contextual information can be processed internally, which is the case in LSTM-based models. Thus, we do not consider this to a key issue for advancing ML-IDS. However, the design choices may impact the computational performance and delays between the malicious event and its detection, especially in IoT networks with centralized intrusion detection.

In a related work, Rahman et al. [17] compare the performance of centralized and distributed ML-IDS, thus highlighted another network management aspect of ML-IDS. The data can be collected at multiple vantage points and forwarded to the central IDS or processed by a distributed IDS (such as one employing federated learning) to detect attacks on each vantage point for reduced delays, bandwidth usage, and distribution of the workload among multiple nodes.

D. What features to use?

An absolutely vital question with regards to ML-IDS is which features to use. Feature engineering is typically one of the most difficult parts of proposing a ML-based solution to any problem. Numerous paper discuss this issue and elaborate on feature engineering. However, there is a potentially problematic repeated pattern among research paper on ML-IDS and that is blindly using all features of publicly available datasets. We argue that the datasets are not realistic in terms of feature selection and, thus, models trained on full scale of available features may become very difficult to implement. At the same time, defining solely on NetFlow/IPFIX or other standardized features might be limiting the ML performance. Still, the use of any uncommon feature (i.e., feature that cannot be extracted from a single packet of flow record) should be justified and parsers capable of extracting such features should be implemented or referenced.

There are several examples of potentially problematic features that we can classify in two groups. In the first group, there are features that cannot be obtained from the network traffic and would require additional tools to provide them. For example, in KDD '99 dataset, there are features indicating the number of login attempts, indicators of successful login or login as root. Since most of the network traffic is now encrypted, making the deep packet inspection impossible, such features can now be obtained only from system logs on the target machine, which requires access to the target's logs and a difficult correlation with network flows. Further, IDS leveraging this feature is no longer an NIDS, but a combined NIDS/HIDS, potentially invalidating the proposed use case or scenario.

The second group of problematic features are uncommon network-based features, such as the number of connections having the same source or target or number of flows between the same hosts but on different ports. While obtaining such features might be an algorithmically trivial tasks, the parser of such features might be difficult to implement to allow for obtaining such features in high-speed networks. NetFlow/IPFIX probes have become highly optimized and efficient over decades of intensive development, and it may take months or years to reach similar computational performance with probes collecting alternative features. Therefore, the use of such features should be justified and the data collection process should be elaborated on, especially if the scenario assumes high-speed network traffic or low computational power.

E. Which model (and optimizations) to choose?

Enormous number of various models and their optimizations were used for intrusion detection. Popular recent approaches based on deep learning (e.g., Deep Neural Networks (DNN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM)) seem to outperform traditional ones (e.g., decision trees, random forests, support vector machines). However, there is no definite answer which one to choose for the best results, since new ones proposing even better performance are proposed often and their performance (accuracy, precision, recall, etc.) in most cases significantly exceeds 90%. The differences between two approaches can be as low as 0.1%. However, vast majority of works only compares the performance of their ML models to another works using ML models on the same datasets and experiments using multiple datasets or live networks are scarce. Common assumption is that the model trained on one dataset will most likely fail or underachieve on another dataset or in live network traffic, but we do not have a proof of that – perhaps because of avoidance of publishing negative results. Until we objectively measure the performance of ML-IDS in live networks or at least over multiple datasets and consider other issues (computational performance, frequency and difficulty of retraining, etc.), we may assume most of the ML models roughly equivalent. Nevertheless, we may temporarily disregard all the works dealing with various optimization, since such optimizations are premature and their time is yet to come.

F. How to train the model and how transferable it is?

Training the model on live data is only possible for anomaly detection since the lack of ground truth, and, thus, the datasets are used. The usual procedure in the literature is to use one or more existing datasets, and for each dataset use part of it for training, and the remainder for testing. While this is methodologically correct, using only one or a few datasets will not produce a model that is usable in production simply because of a limited number of attacks and their variations in the datasets and the lack of background traffic. Moreover, the quality and availability of dataset is a widely recognized problem in cybersecurity, and it is not suprising to see research works using outdated and flawed datasets like KDD'99 even in recent years. It is also surprising how many research works focus on specific environments, such as IoT or cloud, but use generic IT datasets for evaluation, completely missing the specifics of the environment.

The state of the art approach is simultaneous learning on two sources – the model is trained on the dataset, so it correctly identify the attacks, and (simultaneously or subsequently) trained on sample of data from the production environment to recognize the benign background traffic and common traffic patterns in the specific settings. One such approach is referred to as siamese neural networks [9]. While this may sound sufficient, it brings additional research challenges and poses additional overhead to the operators in terms of time and resources and also expertise required.

In conclusion, there is a way, although difficult, to train the ML-IDS to detect attacks well while acknowledging specifics of local environment. However, we are still relying on datasets and may face problems in situations, where no samples of background traffic are available (e.g., because the infrastructure to protect is not yet running). There is a need to produce and update the datasets with the traces of novel attacks.

G. How often to retrain the model and who should do it?

Another issue arises due to two important factors in ML and cybersecurity, the ML drift, which describes the situation, in which the ML model loses its accuracy over time, and everchanging cybersecurity threat landscape, in which the novel attacks or their variants appear on a daily basis [18]. The major benefit of ML-IDS is that it is capable of detection the attacks and its variants, contrary to traditional signature-based IDS, where there would be a need for a signature for each attack variant. However, even this benefit may be lost over time – the variants can move too far from the original attack or a novel attack may appear that does resemble any of the previous ones. Similarly, the benign background traffic is also changing its patters, some of which may resemble adversarial samples in the training set. Thus, there is definitely a need to retrain the models if they are to be used in practice.

The problem is studied by the research community [19], but how often and who should be doing it is an unresolved question in the literature. We may only guess whether to perform this once a month or once a year, since no long-term measurements are not known to us at the moment. Retraining the models may also pose a major overhead for the operators, which also poses a risk of the users not retraining their models often or at all due to the lack of time, thus losing the benefits of ML-IDS and vanishing the efforts spent on deploying it.

H. What is the computational performance of ML-IDS?

A critical issue that is avoided in the literature is the computational performance of ML-IDS, with a few exceptions [20]. If a computational performance evaluation is conducted, the authors typically show configuration of the computer on which the experiment was conducted and measure the time it took to 1) train the model, and 2) process the testing dataset in a batch. We find these metrics insufficient. Training time might be interesting in certain use cases, but if it finishes in reasonable time (e.g., minutes or few hours at most), it is not as important as the data preparation, which may take hours or days of manual labor of an expert. Processing the whole batch of testing data may give only a rough estimate of performance. However, one of the important metrics is the network throughput, i.e., how much network traffic can be successfully processed by the IDS in terms of packets, flows, or bytes per second using a commodity hardware. This cannot be simply derived from the batch processing time and size of dataset due to numerous implementation decision to be made. Nevertheless, authors should state what is the expected network throughput for the ML-IDS and ensure its implementation will be capable of operating at 1, 10, or even more Gbps. The situation may get complicated in IoT use cases, whether on device or centralized, where the ML-IDS should be capable of either run on low-power device or it should be ensured that the devices are capable of collecting and sending data to process to a central instance of IDS. Still, this questions remains mostly unaddressed in vast majority of works on ML-IDS, even if they have an existing implementation.

I. How are the alerts raised, how do they look like, and how many are there?

Since majority of ML-IDS proposed in the literature lack implementation or only implements the ML parts, it is not clear how the alerts are raised and would they look like. The features are extracted from the raw data and forwarded to the ML model as a vector, often undergoing normalization and other forms of manipulation. The ML model then decides whether the vector described an attack or a benign traffic. An alert is raised when an attack occurs, but what would it contain? A simple alert informing about an attack reduces the expresiveness of ML-IDS to the level of ADS, even if meta-data like involved IP addresses or ports are involved. Moreover, attack types differ in what is important to alert – we are interested in source IP address and destination port in horizontal scans, while the list of scanned IP addresses is not important and often too large. In contrary, in DDoS, we can only make use of the destination IP address and port, not the potentially vast number of sources. We argue that unless the ML-IDS can recognize between various types of attacks, it cannot provide the information of the same level of detail and relevance as traditional signature-based IDS.

Moreover, since the alert raising mechanisms are rarely implemented, we do not know how many alerts would an ML-IDS raise, regardless of using aggregation mechanism or not. Information overload and alert fatigue are major concerns in operational cybersecurity, so we should avoid using detection tools that produce large amounts of alerts with low information value [21], [22]. In case of ML-IDS, there is a need to employ thresholds, alert aggregation, and possibly even correlation even before the alert is send out by the IDS, since even a simple malicious action like a network scan would be spread among hundreds or thousands of network flows and, later, feature vectors. Timing of the detection in another unknown, since the alerts can be raised in real time, but at the risk of raising too many alerts. In conclusion, the evaluation of the ML-IDS based simply on correct labelling of feature vectors is formally right, but does not tell much about the actual capability of the tool to deliver timely and adequately-sized information about an attack.

J. What options does a user have to configure or modify the IDS?

ML-based systems are typically black boxes with close to no options for configuration. In this question, we are not interested in the configuration of inputs or outputs (e.g., where to read the input data, where to send the alerts), but in the configuration of the function of the ML component itself. The natural way of reconfiguring an ML-based system is to re-training the model. However, an ML-IDS may compose of a single model trained to detect various attacks simultaneously, while having no mechanism to differentiate between types of attacks, and reconfiguring the model may be a difficult task requiring a skilled personnel, as we discussed earlier.

In cybersecurity, we often encounter situations when there is a need to adjust the detection tools, e.g., by turning off the underperforming or no longer needed detection methods or temporarily suspend the detection of certain attacks. Alternatively, in threshold-based detection, we can simply manipulate the thresholds, e.g., to reduce false positive rate and raise alert only in cases, where it is clear there is an attack. ML-IDS are expected to lack such simple features and will likely require a postprocessing of the alerts to be able to quickly adapt to changing environment.

The lack of configuration may also lead to lower reliability of ML-IDS and usability in automated incident response procedures. Practitioners prefer low false positive rates over other parameters since there will always be some attacks that avoid detection, but incorrect incident response (e.g., mitigation action triggered by false positive detection) may lead to harm to the infrastructure, restricting the users, and reputation loss of the security team. Threshold-based detection methods often use thresholds that minimize false positive rate to 0 %, regardless of true positive rate, just to allow prompt and automated incident response (e.g., blocking the attacker). ML-IDS are usually trained to maximize true positive rate, so unless they minimize false positive rate by careful training or other methods, they are risky to be used in the discussed situations.

IV. Discussion

Herein, we follow the question-answer format of the previous section, but instead of querying the applicability of ML-IDS, we formulate the question so that the answers suggest how to make use of ML-IDS and what should be done to make effective and efficient use of them.

A. Is it really better than traditional IDS?

The overview of operational issues in the previous section highlighted several existing of potential drawbacks of ML-IDS. There are numerous commercial or open-source traditional IDS that are well known by practitioners and efficient even in high-speed networks, and there are numerous signatures, patterns, or detection algorithms available. The signatures, patterns, and algorithms can be easily manipulated with and configured, often simply by moving a threshold. The user can freely choose what to detect and how and keep an eye on performance, the alerts are descriptive, the type of attack can be inferred from the used signature, and the true and false positive rates can be manipulated by configuration to achieve desired outcomes.

On the contrary, ML-IDS actually offer very few improvements. We may rightfully argue that the capability to detect as many attacks as possible is the most important feature of an IDS, but we should not underestimate the other features. ML-IDS seem to improve precision and accuracy of detection for the cost of disabling or complicating other features. The real question is: is the improvement in detection capability so significant that we can omit the other features? Alternatively, can we compromise on the features of an IDS?

In related work, there are a few benchmarks or comparisons, but often comparing traditional IDS between themselves [23] or ML-IDS between themselves [24], [25]. A qualitative and quantitative comparison, evaluation, and benchmarking between traditional and ML-based IDS would be a valuable addition to the field.

B. How would a good ML-IDS look like?

What is a good ML-IDS is up for discussion and there is no definite answer yet; we may only argue which features are more important and which less so. It also highly depending on the particular use case. Nevertheless, there are some aspects that should be considered when proposing an (ML-)IDS.

To start with, it is a good idea to use standardized inputs, such as network flows. Data collection can be offloaded to a dedicated measurement infrastructure, there is no need to develop and maintain custom packet parsers. Network flow-based input may not be as rich as the data from custom parsers, but it will be easier to set up or replace traditional IDS with ML-IDS. Scalability of flow-based approaches would be another benefit.

We should aim for an ML-IDS that distinguishes between different types of attacks. This is actually a challenging goal, but there are two approaches to take. First, the ML-IDS may use an ensemble of models, each trained to detect different type of attack. This is essentially very similar to traditional IDS, only with models instead of signatures. While it may seem more laborious to train each model separately, it may be simpler – each model would be trained on traces of a particular attack, the dataset of background traffic can be shared, and underperforming models can be easily turned off or retrained separately. However, computational performance may be hindered severely. Second, we may use multi-label classification – the model is trained on the dataset, in which each attack class is labelled separately, and the detected intrusion may be labelled with one or more such labels. Such an approach is more difficult even from the ML perspective and requires much more elaborated datasets, but may provide valuable insights into attacks and anomalies that would not fit into one category. AS a side note, we dot consider explainable AI (XAI) to be a reasonable solution to this problem, since it is time- and resource-consuming and only pinpoints the features that caused classifying the input vector as malicious – a security analysts should rather analyze the raw network traffic and correlate it with other sources (e.g., systems logs and threat intelligence), since the features can be too abstract.

Finally, we should invest into usability of ML-IDS in the training phase and possible re-training. If the ML-IDS are to be used widely, we should not expect the users to gather knowledge of ML to train their own models. Instead, we should facilitate or outsource the training – pre-trained models could be shared publicly or be provided by commercial enterprises, especially for ensemble-based systems. End used would then use the pre-trained models directly or re-train them with the sample of their own background traffic, thus avoiding dealing with the datasets and labeling malicious patterns.

C. Is it really worth it?

There is a popular concept in the world of DevOps, the *pet and cattle* analogy [26]. In essence, pets are devices and services to which we pay special attention; they are given unique names and are often configured and maintain manually, paying attention to many details, and spending much time on them. The cattle refers to devices or services that are deployed often in large numbers and configured and maintained mostly automatically, and no special attention is given the individual assets when an issue arises since they can often be easily replaced or redeployed. While we are not in the DevOps field, we can still get inspired by this concept – so far, we have only seen ML-based IDS acting as a pet. It is hard to imagine the ML-IDS to act as cattle as that would require significant development in the areas outlined in Section III.

We would like to encourage anyone working with ML-IDS to "groom their pets," since this is the way to resolve the issues around ML-IDS. However, we argue that the way forward is turning the pets into cattle, i.e., preparing an ML-IDS that is easy to deploy, configure, and use. We understand that a well groomed pet may present a great contribution to the user, but at this moment, it typically means that a dedicated person is sacrificing time and effort to only one tool in the cybersecurity toolset. Given the worldwide shortage of cybersecurity professionals and the increasing workload of often understaffed teams, we may ask whether it is really worth it to spend resources on ML-IDS.

D. Are there any alternative approaches?

Even if it seem, by the sheer number of publications, that training the ML models to perform intrusion detection is the dominant approach, it is not a single one. A viable alternative of leveraging AI and ML in intrusion detection is letting the AI generate signatures for traditional IDS [27]. Such an approach allows for leveraging AI without the need of touching the existing monitoring and detection infrastructure. Preliminary results are promising and may be deployed immediately, and if successful, may persuade users into adopting AI/ML-based solutions faster than ML-IDS.

V. Conclusion

The research in machine learning for intrusion detection has produced numerous publications and even became overly hyped topic. Despite the number of publications in over a decade, we still do not see that many implementations, commercial products, nor field trials. Researchers and practitioners has even started to question the role of ML in cybersecurity and its applicability. In this paper, we analyzed the ML-based intrusion detection from the perspective of network security management and operations and raised number of questions regarding the deployment and effective and efficient use of ML-IDS. Our major finding is that while ML-IDS exceeds traditional IDS in detection capabilities, it is for the cost of extensive expert labor, lack of configuration, and frequently also inability to understand what was detected. We argue that at the moment, ML-IDS present an interesting tool for cybersecurity experts with sufficient knowledge of ML and plenty of time and resources to deploy it successfully.

In our future work, we are going to further survey and analyze the domain of ML-IDS, look up prototypes and implementations, and scrutinize them under operationally-relevant conditions. We would also like to raise awareness of operational issues of cybersecurity and steer the research community towards the applicable results. We hope the effort spent on researching ML-IDS will not go in vain, but that the discussed issues of ML-IDS will be resolved and applied in practice.

References

[1] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[2] I. F. Kilincer, F. Ertam, and A. Sengur, "Machine learning methods for cyber security intrusion detection: Datasets and comparative study," *Computer Networks*, vol. 188, p. 107840, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621000141

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

[4] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Lessons learned on machine learning for computer security," *IEEE Security & Privacy*, vol. 21, no. 5, pp. 72–77, 2023.

[5] F. Ceschin, M. Botacin, A. Bifet, B. Pfahringer, L. S. Oliveira, H. M. Gomes, and A. Grégio, "Machine learning (in) security: A stream of problems," *Digital Threats: Research and Practice*, sep 2023, just Accepted. [Online]. Available: https://doi.org/10.1145/3617897

[6] G. Apruzzese, P. Laskov, and J. Schneider, "Sok: Pragmatic assessment of machine learning for network intrusion detection," 2023.

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

[8] M. Komisarek, M. Pawlicki, T. Simic, D. Kavcnik, R. Kozik, and M. Choraś, "Modern netflow network dataset with labeled attacks and detection methods," in *Proceedings of the 18th International Conference on Availability, Reliability and Security*, ser. ARES '23. New York, NY, USA: Association for Computing Machinery, 2023.

[9] M. Pawlicki, R. Kozik, and M. Choraś, "Improving siamese neural networks with border extraction sampling for the use in real-time network intrusion detection," in *2023 International Joint Conference on Neural Networks (IJCNN)*, 2023.

[10] 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*, 2010, pp. 305–316.

[11] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security," in *2018 10th International Conference on Cyber Conflict (CyCon)*, 2018, pp. 371–390.

[12] G. Apruzzese, P. Laskov, E. Montes de Oca, W. Mallouli, L. Brdalo Rapa, A. V. Grammatopoulos, and F. Di Franco, "The role of machine learning in cybersecurity," *Digital Threats: Research and Practice*, vol. 4, no. 1, mar 2023. [Online]. Available: https://doi.org/10.1145/3545574

[13] A. Corsini and S. J. Yang, "Are existing out-of-distribution techniques suitable for network intrusion detection?" 2023.

[14] S. Muneer, U. Farooq, A. Athar, M. Ahsan Raza, T. M. Ghazal, and S. Sakib, "A critical review of artificial intelligence based approaches in intrusion detection: A comprehensive analysis," *Journal of Engineering*, vol. 2024, no. 1, p. 3909173, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2024/3909173

[15] S. V. N. Santhosh Kumar, M. Selvi, and A. Kannan, "A comprehensive survey on machine learning-based intrusion detection systems for secure communication in internet of things," *Computational Intelligence and Neuroscience*, vol. 2023, no. 1, p. 8981988, 2023.

[16] N. Sahani, R. Zhu, J.-H. Cho, and C.-C. Liu, "Machine learning-based intrusion detection for smart grid computing: A survey," *ACM Trans. Cyber-Phys. Syst.*, vol. 7, no. 2, apr 2023. [Online]. Available: https://doi.org/10.1145/3578366

[17] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Network*, vol. 34, no. 6, pp. 310–317, 2020.

[18] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "Insomnia: Towards concept-drift robustness in network intrusion detection," in *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 111–122.

[19] F. Uccello, M. Pawlicki, S. D'Antonio, R. Kozik, and M. Choraś, "An innovative approach to real-time concept drift detection in network security," in *Advances in Internet, Data & Web Technologies*, L. Barolli, Ed. Cham: Springer Nature Switzerland, 2024, pp. 130–139.

[20] Y.-C. Lai, D. Sudyana, Y.-D. Lin, M. Verkerken, L. D'hooge, T. Wauters, B. Volckaert, and F. De Turck, "Machine learning based intrusion detection as a service: task assignment and capacity allocation in a multi-tier architecture," in *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, ser. UCC '21. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3492323.3495613

[21] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber, "Dealing with security alert flooding: Using machine learning for domain-independent alert aggregation," *ACM Trans. Priv. Secur.*, vol. 25, no. 3, apr 2022.

[22] M. Baruwal Chhetri, S. Tariq, R. Singh, F. Jalalvand, C. Paris, and S. Nepal, "Towards human-ai teaming to mitigate alert fatigue in security operations centres," *ACM Trans. Internet Technol.*, vol. 24, no. 3, jul 2024.

[23] Q. Hu, S.-Y. Yu, and M. R. Asghar, "Analysing performance issues of open-source intrusion detection systems in high-speed networks," *Journal of Information Security and Applications*, vol. 51, p. 102426, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212619306003

[24] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset," *IEEE Access*, vol. 9, pp. 22 351–22 370, 2021.

[25] S. Rizvi, M. Scanlon, J. McGibney, and J. Sheppard, "An evaluation of ai-based network intrusion detection in resource-constrained environments," in *2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2023, pp. 0275–0282.

[26] S. Armstrong, *DevOps for Networking*. Packt Publishing, 2016.

[27] M. Zipperle, Y. Zhang, E. Chang, and T. Dillon, "PARGMF: A provenance-enabled automated rule generation and matching framework with multi-level attack description model," *Journal of Information Security and Applications*, vol. 81, p. 103682, 2024.