IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 24 March 2025, accepted 8 April 2025, date of publication 15 April 2025, date of current version 2 May 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3561105

SURVEY

A Survey of Data Stream-Based Intrusion Detection Systems

RODRIGO SANCHES MIANI[1], GUSTAVO DI GIOVANNI BERNARDO[1], GUILHERME WEIGERT CASSALES[2], HERMES SENGER[3], AND ELAINE RIBEIRO DE FARIA[1]

[1]Faculty of Computing, Federal University of Uberlândia, Uberlândia 38408-100, Brazil
[2]AI Institute, University of Waikato, Hamilton 3240, New Zealand
[3]Department of Computer Science, Federal University of São Carlos, São Carlos 13565-905, Brazil

Corresponding author: Rodrigo Sanches Miani (miani@ufu.br)

This work was supported in part by the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES), Brazilian National Council for Scientific and Technological Development (CNPq) under Grant 302296/2023-9, in part by the São Paulo Research Foundation (FAPESP) under Grant 2019/26702-8 and Grant 2023/00566-6, and in part by the State of Minas Gerais Research Support Foundation (FAPEMIG) under Grant APQ-02196-18.

ABSTRACT Detecting malicious activities in network environments poses a challenge that attracts significant attention due to its complexity and importance. Advances in the field have led to the development of several algorithms that approach the problem under the view of a data stream machine learning task. This task involves a set of steps: data collection or choice of public datasets, data pre-processing, data reduction, development or application of data mining techniques, and evaluation methodology. However, these steps must address the inherent issues of dynamic environments such as data streams and intrusion detection systems. These issues include, but are not limited to, the continuous influx of data, changes in both normal and attack class distributions, the emergence of new attack types, and the scarcity of labeled data examples to update the decision models. This survey provides an overview of intrusion detection systems (IDS) using data stream machine learning techniques, characterizing the literature approaches according to the classic steps of the data mining task. In addition, we discuss recommendations for practical IDS development and highlight datasets and tools that can aid in detecting malicious behavior. Finally, we outline potential avenues for future research and open questions in the field.

INDEX TERMS Computer networks, cybersecurity, data-stream, intrusion detection systems, machine-learning.

I. INTRODUCTION

The cyber-threat landscape is expanding in both scale and complexity. Recent reports [1], [2], [3] highlight ransomware as the dominant threat for the last years, driven by the financial motivations of cybercriminals. Additionally, Distributed Denial of Service (DDoS) attacks are evolving to become more targeted, persistent, and multivector, often exploiting Internet of Things (IoT) devices to target critical sectors like healthcare and transportation. Mobile malware is also a rising concern, with Kaspersky data [4] showing an increase in attacks on mobile devices in 2023. These threats range from adware to sophisticated trojan bankers, which steal banking credentials. Some malware variants employ obfuscated memory techniques [5] to hide their presence and activities, often using phishing emails or mimicking e-commerce websites to deceive victims. To address these challenges, companies and academic institutions are turning to machine learning-based cybersecurity solutions, with intrusion detection systems (IDS) being a notable example.

According to [6], IDS is a mechanism whose main task is to detect malicious activities in host and/or network environments. Using the data provided by the IDS (usually alerts triggered by the system), cybersecurity professionals could analyze the activity to discover, identify, and mitigate ongoing attacks. Given IDS's role in the current cybersecurity infrastructure, the research and industrial community have developed several types of IDS utilizing various methods. Among these methods, data mining techniques, particularly ML, have become a primary approach to enhance IDS capabilities [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Junho Hong.

These systems, often called ML-based IDS, aim to distinguish normal from malicious behavior. There is extensive literature on developing intrusion detection models using data mining algorithms for network-related data. However, as pointed out by [8], ML-based IDS research typically focuses solely on applying data mining algorithms to network datasets. There is little discussion regarding the particularities of employing ML techniques in the intrusion detection problem, such as the type of data, the performance in terms of response time, or whether the detection/learning model can be easily updated to detect new threats.

For example, the learning process of most ML-based IDS found in the literature is typically done using batch learning mode. This means that the ML algorithm is applied once to a static training dataset, and after that, the built model is used to make predictions for incoming data. As discussed in [8], 54 out of 70 surveyed works use this approach to create the intrusion detection model.

Despite the increasing tendency of ML-based IDS to adopt batch learning models (*batch-based IDS*), these algorithms do not address the challenges posed by real-world intrusion detection environments. First, the continuous generation of network traffic at high rates poses a challenge, making it impractical to scan the data as most batch-learning algorithms propose repeatedly. In this case, only part of the data can be stored for a given period, or statistical data summaries are maintained. Second, the decision model is static and induced only once in the batch-learning scenario. However, intrusion detection is characterized by a non-static scenario in which concepts can evolve, making it critical to detect changes and update the decision model accordingly. Intrusion detection systems that address the continuous data generation and update of the model are known in the literature as *data stream-based IDS*. In the context of this paper, we use ML-based IDS to denote the systems that use any ML techniques (batch or not) and data stream-based IDS for the systems that use data stream techniques for intrusion detection.

Recently, some studies have addressed the data stream-based intrusion detection task using incremental or adaptive learning algorithms [9], [10], [11], [12], [13], [14], [15] allowing the decision model to be continuously updated to reflect changes in the data distribution of the network traffic. Some of these works have proposed mechanisms to forget outdated data, which no longer contributes to classifying new data. Updating the decision model is a challenging task that has been approached using unsupervised, supervised, or semi-supervised techniques.

Several studies have also proposed a taxonomy to organize the literature on ML-based IDS. Most of these surveys are focused on the batch-learning scenarios [16], [17], [18], [19]. The studies presented in [7] and [8] discussed challenges more relevant to real-world intrusion detection environments. However, they did not conduct an in-depth investigation on data stream-based IDS. This information is corroborated in [20]. The authors conducted a study to understand which data stream methods for IDS, such as active learning and concept drift detection, were discussed in 21 review or survey papers. Their results indicate that no high-quality reviews or surveys focus on data stream methods for IDS. Our work aims to fill these gaps by conducting a literature review of intrusion detection systems built upon data stream methods. As discussed in Section II, this is the first survey to approach this issue.

This article proposes a survey of the main data stream-based IDSs, which consider network traffic as a continuous stream of data instances and use adaptive or incremental learning algorithms. Our contributions can be summarized as follows: i) a problem formulation that describes the intrusion detection task as a data-stream classification task; ii) a taxonomy to organize the data stream intrusion detection works under the perspective of knowledge discovery from data (KDD); iii) an identification of the main gaps, open questions, and future directions for data stream-based IDS; iv) a discussion of the main challenges of ML-based IDS, such as non-uniformity of data distribution over time, cost of labeling, and update delay; v) recommendations for developing data stream-based IDS; vi) a critical analysis of public intrusion detection datasets and computer tools useful for data stream-based IDS research.

The rest of this paper is organized as follows. Section II discusses recent relevant reviews on ML-based IDS and examines the differences between the proposed study and previous surveys. Section III provides a formalization for the intrusion detection problem in data stream-based scenarios. Section IV presents the research methodology and the proposed taxonomy. Based on the analysis of the surveyed studies, Section V highlights several challenges for building and deploying data stream-based IDS. Section VI introduces guidelines for assisting the IDS research community focused on helpful computer tools and public intrusion detection datasets. Finally, in Section VIII, we present some concluding remarks and future research possibilities.

II. RELEVANT REVIEWS

Over the recent years, several review articles have been published on ML-based IDS [7], [8], [16], [17], [18], [19], [21], [22], [23], [24]. Next, we will discuss and compare such articles to our proposal.

The authors in [16] provide a classification and summary of batch-based IDS proposed until 2019. They aim to outline the main ideas of applying machine learning to security problems and analyze the current challenges and future advances. They argue that the previous surveys [25], [26], [27] primarily focus on how different machine learning algorithms are applied to IDSs. While this may be helpful for machine learning researchers, it does not address how to solve IDS domain issues using machine learning. To address this gap, they define a taxonomy based on detection methods (anomaly detection and misuse detection) and data sources (host-based IDS and network-based IDS). For network-based IDS, they

identify three data sources: packets, flows, and sessions. They also categorize three types of anomaly detection models: statistical, machine learning, and time series. They focus on studying machine learning models, especially the algorithms used to build ML-based IDS. One important conclusion in the paper relates to the lack of practical ML-based IDS. The authors argue that practical IDS needs to have high detection accuracy, run-time efficiency, and interpretability. We believe that data stream models can be better suited to develop ML-based IDS.

In their paper, [17] explore the application of deep learning (DL) methods for building ML-based IDS. The authors present a taxonomy of recent research papers on deep learning methods for intrusion detection and conduct an empirical evaluation of deep learning models using newer datasets such as CIC-IDS2017, CIC-IDS2018, and legacy datasets (KDD 99, NSL-KDD). Regarding the proposed taxonomy, the authors considered the machine learning practitioner's point of view (supervised instance learning, supervised sequence learning, semi-supervised instance learning, and other learning paradigms) rather than just providing a conceptual classification of machine learning models. Focusing on this point of view, especially on the learning component, they want to help researchers in practical tasks such as conducting data processing and building the model learning pipeline. The authors suggest two directions for future research when exploring novel machine learning paradigms and methods in the intrusion detection domains: transfer learning methods and interpretable machine learning techniques. A general issue of DL models is their high computational cost. Given the general idea of using several epochs to reach a global minimum, it becomes challenging to quickly update DL models with new data and provide good predictive performance. In this context, data stream methods that update their models according to the latest data are expected to provide better predictive performance because of the dynamic environment of intrusion detection.

The work in [8] points out that most surveys partially cover the steps necessary to perform intrusion detection from network traffic analysis. They argue that while data mining is the most addressed phase, steps such as data collection, processing and transformation, and detector updates are frequently overlooked in the literature. To address this gap, they propose a review of ML-based IDS proposals using the Knowledge Discovery in Databases (KDD) process, focusing on policies for learning models and keeping them updated. They investigated two approaches: batch learning and incremental learning, with the latter being less common but more suitable for continuous learning with new data. The authors observe that most incremental ML-based IDS are based on unsupervised approaches, but only a few proposals are based on supervised methods. However, they conclude that incremental approaches still need more maturity in the early stages of learning. We aim to investigate these incremental approaches in-depth and understand how this paradigm can enhance the development of data stream-based IDS.

In [18], the authors conducted a literature survey on using neural network algorithms in batch-based IDS. Unlike [17], they did not empirically evaluate neural network algorithms. Instead, they organized the reviewed papers according to the number of citations, dataset, neural network method, and IDS focus. However, they did not provide in-depth discussions or comparisons of the selected papers, and the research directions are generic. One interesting finding is that neural network algorithms have been used during the pipeline of several ML-based IDS.

Reference [19] surveys recent intrusion detection approaches focusing on the evaluation datasets and machine learning techniques employed. The authors note that most investigated studies use supervised or unsupervised learning, with only a small number of studies (5 out of 26) utilizing incremental learning techniques. There is no in-depth discussion regarding the impact of incremental learning techniques or directions for deploying ML-based IDS in real-world scenarios. The authors argue that the unavailability of efficient intrusion datasets is a significant issue for the field. However, there is no indication of future research directions concerning this issue.

Finally, [7] presents a survey focused on nine main challenges related to ML-based IDS or, as they denominate, data-driven network intrusion detection systems. They divide the challenges into the following groups: Lack of Real-World Network Data, Noisy Data, Redundant Data, Weakly Correlated Data, Too Few Labeled Data, Imbalanced Data, Dynamic Data, Big Data, and Small Data. For each challenge, they present a list of recent methods that might be used to resolve them. The solution for two of the challenges related to the instance-based challenges - ''Dynamic Data'' and ''Big Data'' - involves using data stream models and incremental learning. The main challenge here is to figure out how to use adaptive models to handle the changing landscape of cybersecurity. The authors briefly discuss how some ML-based IDS proposals use data stream models and incremental learning to tackle this issue. Despite acknowledging the importance of such a topic, there is no in-depth discussion or taxonomy about the proposals that adopt adaptive models or recommendations for developing practical ML-based IDS.

Some recent surveys [21], [22], [23], [24] have studied ML-based IDS from various perspectives. However, none have examined the surveyed works from the data stream perspective. Reference [21] focused on the method used by the IDS to identify an attack (signature, anomaly, or hybrid). For anomaly-based IDS, they discussed characteristics such as datasets, classifiers, evaluation metrics, and some pros and cons. Reference [22], on the other hand, only addressed ML-based IDS and proposed a taxonomy based on the learning approach: supervised, unsupervised, and ensemble. Similarly, [23] studied ML-based IDS schemes and examined the algorithms, evaluation metrics, datasets, attacks, and

performance accuracy. Reference [24] follows a similar approach to the previous reviews but also discusses some interesting issues, such as feature detection, sources of features, and certain shortcomings. In discussing future research perspectives, they pointed out concept drift as a promising direction.

Regarding data stream-based IDS, the only study that somewhat discusses certain concepts presented here is [28]. The authors use Online machine learning (OL) instead of data stream to refer to an incremental learning-based algorithm that receives data sequentially, builds a model, and updates it frequently with new data. They investigated the adoption of OL techniques applied to computer networks, especially in the IoT domain. According to the authors, concept drift and the lack of labeled data are significant challenges when using batch/offline learning algorithms in computer networks. The authors described some state-of-the-art solutions for tackling these issues, such as adaptive learning methods (e.g., ensemble learning). Finally, the authors empirically evaluated several online ensembles and tree-based algorithms for network traffic classification. To this end, they used three public datasets for intrusion detection: UNSW-NB15, NSL-KDD, and UNSW 2018 IoT Botnet. They concluded that tree-based algorithms achieved better performance in terms of accuracy when compared to ensemble algorithms.

Table 1 summarizes the past reviews. The idea of data stream-based IDS is either not mentioned or only superficially addressed in these studies. For example, [19] found that a few studies use incremental learning techniques. [8], on the other hand, states that batch learning IDS are not explicitly designed for streaming scenarios as they do not evolve with time. Based on this idea, the authors provide an initial discussion about data stream-based IDS (as they mention as ''Incremental learning NIDS'') focused only on the learning side: supervised or unsupervised. On the other hand, [28] highlighted the importance of using data stream techniques in the network domain and even conducted some experiments using several IDS datasets. However, they did not investigate the relationship between data stream methods and the intrusion detection problem. Here, we aim to fill this literature gap by investigating the current state of the art for stream-based IDS and providing an in-depth study about some specific characteristics of this paradigm, such as the model update and evaluation. We also provide recommendations and practical guidelines for developing data stream-based IDS. Finally, we argue that the data stream-based IDS concept would help solve several challenges of ML-based IDS [9]: non-uniformity of data distribution over time, cost of labeling, and update delay. Therefore, to our knowledge, our work proposes the first survey using data stream methods to develop intrusion detection systems.

III. INTRUSION DETECTION: PROBLEM FORMALIZATION

Recent changes in network data collection due to the popularization of IoT sensors and wearable devices and the growth of network traffic have generated massive amounts of data at high velocity, which requires real-time analysis to identify malicious activity. Under the machine learning perspective, identifying malicious behavior and its distinction from normal access can be formalized as a classification task that aims to categorize network traffic as a normal pattern or an attack.

Each network traffic object O_i represents raw data extracted from each packet or an aggregated summary from packets, named flow data. A flow is defined as a set of IP packets passing an observation point in the network during a specific time interval. All packets belonging to a particular flow have common properties [29]. The object O_i comprises d attributes, which store numeric or nominal values. Some examples of packet attributes include source and destination port, header length, time to live (TTL), and TCP ack number. Some examples of flow attributes include total packets, the total size of packets, and the mean time between two packets sent in the flow.

When considering intrusion detection as a binary classification task, each data object O_i of network traffic is classified in one of two classes in the set Y, where $Y = \{normal, attack\}$. This task aims to learn a target function f that maps each object O_i to one of the predefined classes Y, $f(O_i) \rightarrow Y$.

Most of the intrusion detection works use supervised approaches that learn the target function (or classification model) from a set of labeled objects T, named Training Set, where $T = \{(O_1, y_1), (O_2, y_2), (O_3, y_3), \ldots, (O_l, y_l)\}$. The classification model f induced from T is then used to classify new examples of the Test Set. Semi-supervised approaches can also be used when few labeled examples are available.

The intrusion detection can also be faced as a multiclass classification task, where the set of class labels Y is extended to Y', where $Y' = \{normal, attack_1, attack_2, \ldots, attack_n\}$, where $attack_i$ represents different types of malicious traffic and n is the number of possible attacks. In this scenario, it is important to identify the specific attack type and distinguish between normal and attack traffic.

According to [8], IDS that implement these strategies are also known as Misuse-based systems, in which the classification of a new instance corresponds to a potential class learned during the training phase.

When considering intrusion detection as a one-class classification task, only data objects of the normal traffic are available to induce the classification model. Thus, the model can explain new data objects and then classify them as normal traffic; otherwise, they are classified as an attack (an abnormal behavior). This strategy is commonly used in the literature and referenced as anomaly detection or anomaly-based detectors [8]. In this case, unsupervised learning techniques are commonly used.

Considering data stream-based IDS deployed in real-world scenarios where new data objects are continuously captured from network devices, new types of attack can appear without

TABLE 1. Summary of past surveys and its respective topics and data-stream mentions.

Reference	Topics	Data-stream
[16]	Taxonomy of ML-based IDS from the perspective of data source and detection methods	×
[17]	Taxonomy of deep learning models in IDS	×
[8]	Categorization of IDS from the perspective of KDD	Preliminary discussion focused only on the learning side
[18]	Review about the use of neural networks in IDS	×
[19]	Review of ML-based IDS from the perspective of datasets and ML algorithms	Mention that only a few studies use incremental learning techniques
[7]	Taxonomy with eight challenges for ML-based IDS (they call as data-driven IDS)	Mention that one specific challenge (Dynamic Data and Big Data) involves using data stream models
[28]	Analysis of the adoption of Online machine learning (OL) applied to computer networks, especially for IoT domain	Enumerate some generic challenges (concept drift and imbalanced classes), and empirically evaluated some stream algorithms in three datasets
[21]	In-depth synthesis and analysis of the various IDS solutions focusing on detection approaches	×
[22]	Taxonomy of ML-based IDS from the perspective of learning models (supervised, unsupervised and ensemble)	×
[23]	Review of ML-based IDS from the perspective of datasets and attack types	×
[24]	Analysis of challenges of using ML techniques for building efficient IDS	×

prior knowledge of the number of unique types n. In such cases, the classification model must be updated when a new attack is identified. In addition, the normal and the known attack behavior can change over time, a phenomenon known in the literature as concept drift, indicating that the data distribution has changed. Therefore, more than merely learning a function f to achieve a good performance in such a dynamic environment, the classification model must be regularly updated to reflect changes in the data distribution and the emergence of new types of attacks. However, it is infeasible to train a new model each time the data distribution changes, or a new attack type is identified, making incremental learning algorithms a common strategy to address these challenges. Nevertheless, these techniques must do more than increment the classification model according to new classes and data distribution changes. They must also forget old concepts that no longer represent recent data objects of the network.

In the next section, we investigate the stream-based IDS literature regarding the characteristics discussed in this section: data source, classification task (binary/multiclass), learning strategies, and model update.

IV. RESEARCH METHODOLOGY AND TAXONOMY

We proposed a method composed of five steps for conducting the literature review of IDS studies using data stream mining techniques:

1) Creation of a boolean string search;
2) Submission of the string to different digital libraries and export the results to the Parsifal, a tool to perform systematic literature reviews[1];
3) Removal of papers from lower impact conferences, i.e., conferences that do not appear in the following rankings: CORE[2] and QUALIS[3];
4) Manual insertion of papers;
5) Reading each paper's title and abstract.

First, we crafted the following string as input for our search strategy.

("intrusion detection" OR "intrusion detection system" OR "network intrusion detection") AND ("data mining" OR "machine learning" OR "novelty detection" OR "anomaly detection") AND ("online learning" OR "data streams" OR "data stream" OR "stream-based" OR "incremental learning" OR "streaming learning" OR "stream learning" OR "adaptive learning") AND ("computer networks" OR "computer network")

We attempted to find data stream-based IDS studies by using the following terms: "online learning" OR "data streams" OR "data stream" OR "stream-based" OR "incremental learning" OR "streaming learning" OR "stream learning" OR "adaptive learning". In the second step, we found 670 papers across the following datasets: ACM Digital Library, IEEE Digital Library, ScienceDirect, Scopus, and Springer Link. After performing the first three steps, we obtained 54 papers related to specific studies about data stream-based IDS. For example, we removed papers that apply data stream-based mining techniques to generic anomaly detection solutions.

We organized every work regarding the following KDD steps: data source, data preprocessing, data reduction, data mining, and interpretation and evaluation. Those first-level attributes were partially based on the taxonomy proposed by [8]. Fig. 1 illustrates the proposed taxonomy, and Tables 2, 3, and 4 summarize the investigated efforts to design data stream-based IDS. Data type attribute refers to packet, flow, or event data. Data preprocessing refers to normalization, conversion, or anonymization. Data reduction refers to feature or sample reduction. Data mining refers to i) supervised or unsupervised models, ii) binary or multiclass detection models, iii) the mechanism that triggers the update

[1]https://parsif.al
[2]https://www.core.edu.au/
[3]https://ppgcc.github.io/discentesPPGCC/pt-BR/qualis/

TABLE 2. Summary of data stream-based IDS studies ordered by publication year.

Ref.	Year	Data type	Data preprocessing	Data reduction	Data mining	Interpretation and evaluation
[30]	2005	packet data	normalization	sample	unsupervised binary blind no feedback	batch recall, FAR
[31]	2005	flow data	-	feature sample	single, unsupervised binary blind feedback (partial)	holdout recall, FAR
[32]	2005	packet data	-	sample	unsupervised multiclass blind no feedback	- FAR
[33]	2008	flow data	-	sample	unsupervised multiclass blind no feedback	holdout False abnormal, false normal, purity, processing time
[34]	2008	flow data	-	sample	ensemble binary blind feedback (total)	holdout recall, FAR
[35]	2009	packet data	-	-	unsupervised binary informed (change detector) no feedback	- ROC curve
[36]	2009	flow data	normalization conversion	sample	single, unsupervised binary blind no feedback	holdout accuracy, running time
[37]	2009	flow data	-	sample	single, semi-supervised multiclass blind feedback (total and partial)	holdout precision, recall, F1
[38]	2010	packet data	-	sample	single binary informed (change detector) feedback (total)	streaming Forecast error False positive
[39]	2012	packet data	-	sample	ensemble binary blind feedback (total)	holdout accuracy
[40]	2012	packet data	-	sample	unsupervised binary informed (change detector) no feedback	streaming accuracy, precision, recall, FAR, AUC, ROC curve
[41]	2013	flow data	-	feature sample	ensemble binary informed (change detector) feedback (total)	streaming AUC
[42]	2013	flow data	-	sample	unsupervised binary informed (change detector) no feedback	batch FAR
[43]	2013	flow data	conversion	sample	semi-supervised binary informed (change detector) feedback (partial)	streaming accuracy
[44]	2013	packet data	-	sample	single multiclass blind	batch recall, FAR
[45]	2014	flow data	normalization	feature sample	unsupervised binary informed (change detector) no feedback	streaming recall, FAR, ROC curve, cluster distance
[46]	2015	flow data	conversion	feature sample	single binary, multiclass blind feedback (total)	batch accuracy, FPR, F1, precision, processing time
[47]	2015	flow data	-	sample	unsupervised binary blind no feedback	streaming accuracy, FAR
[48]	2015	flow data	-	sample	unsupervised multiclass informed (change detector) no feedback	batch, holdout recall, FAR, AUC, ROC curve, running time
[49]	2015	flow data	-	sample	single, ensemble binary blind, informed (change detector) feedback (total and partial)	streaming accuracy, FAR, Kappa, running time, memory consumption

(implict or informed by a change detection method), and iv) the availability of labeled instances (total feedback, partial feedback or no feedback). Interpretation and evaluation refer to the evaluation methodology (holdout, online or offline)

TABLE 3. Summary of data stream-based IDS studies ordered by publication year (cont).

Ref.	Year	Data type	Data preprocessing	Data reduction	Data mining	Interpretation and evaluation
[50]	2016	flow data	-	sample	ensemble binary informed (change detector) feedback (total)	- recall, precision, AUC
[51]	2016	flow data	-	sample	single binary blind	batch, holdout, streaming
[52]	2017	flow data	-	feature sample	ensemble multiclass blind feedback (partial)	- accuracy, kappa
[53]	2017	flow data	-	sample	single, unsupervised multiclass blind feedback (total and partial)	holdout accuracy
[54]	2017	flow data	-	sample	single, unsupervised binary informed (change detector) feedback (total)	streaming precision, recall, F1, Kappa
[55]	2017	flow data	-	sample	single, ensemble binary blind feedback (total)	holdout accuracy
[56]	2017	packet data	-	-	ensemble binary blind no feedback	streaming accuracy
[57]	2018	packet data	normalization	-	unsupervised binary blind no feedback	batch accuracy, recall, FAR
[58]	2018	flow data	-	sample	unsupervised binary blind no feedback	streaming
[59]	2018	flow data	normalization conversion	sample	single, ensemble binary, multiclass informed (change detector) feedback (total)	streaming accuracy, precision, recall
[11]	2018	flow data	-	feature sample	single binary blind feedback (total)	streaming precision, recall, FAR
[60]	2019	flow data	anonymization	sample	ensemble binary blind feedback (partial)	batch, holdout accuracy, recall, FAR, AUC, ROC curve, F1
[61]	2019	flow data	-	feature sample	single multiclass blind feedback (total)	holdout precision, recall, F1
[62]	2019	flow data	-	sample	unsupervised binary, multiclass informed (change detector) no feedback	holdout accuracy
[63]	2019	event log	-	-	single binary informed (change detector) feedback (total)	streaming recall
[64]	2019	-	-	-	unsupervised binary blind no feedback	-
[14]	2019	flow data	normalization conversion	sample	ensemble, unsupervised binary blind feedback (total)	holdout FNEW, MNEW
[12]	2019	flow data	-	sample	single, unsupervised multiclass blind feedback (total)	-
[65]	2019	flow data	-	sample	single, ensemble binary blind feedback (total)	holdout accuracy, FAR
[66]	2019	flow data	-	sample	unsupervised multiclass blind no feedback	batch recall, FAR

and evaluation measures such as Accuracy, Recall, Precision, or other metrics. Cells with "-" represent an unspecified attribute. Next, we discuss the surveyed works according to the proposed taxonomy.

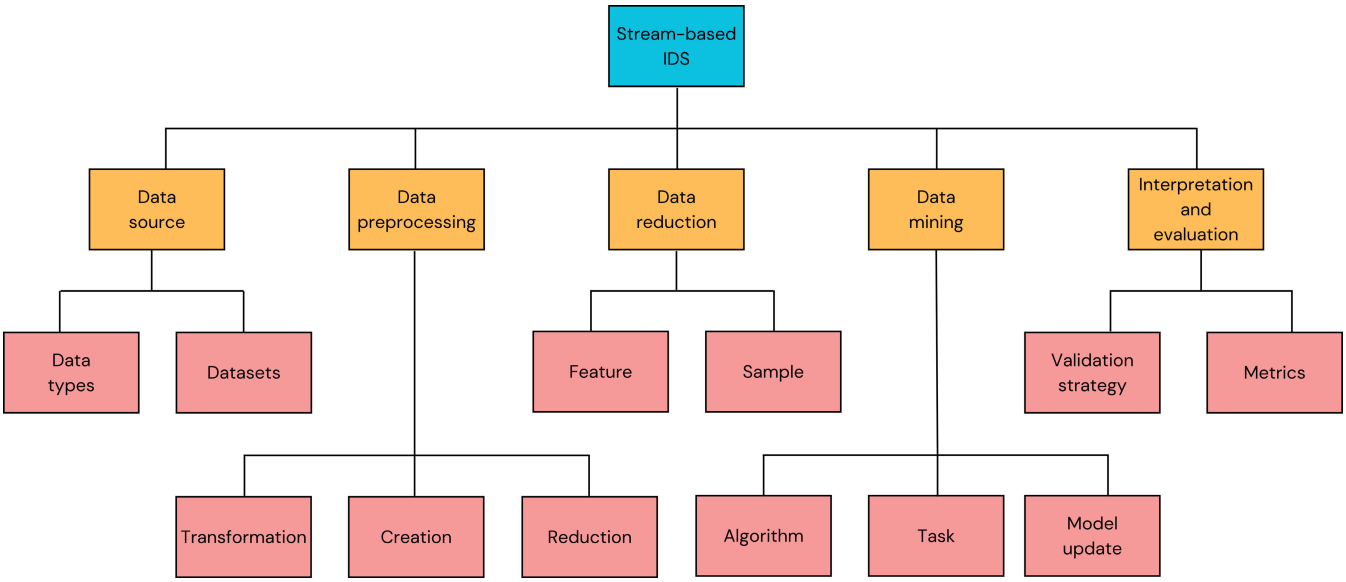

FIGURE 1. Proposed taxonomy for data stream-based IDS.

A. DATA SOURCE

Network traffic data is essential for data stream-based IDS because the learning process is based on data patterns. Therefore, it is important to understand the main data types and how researchers obtained and used them. For instance, datasets containing normal and anomalous networking patterns, labeled (or not) by human specialists or other IDS, can be used to build, update, or evaluate intrusion detection models. Although some old intrusion datasets have been used for over one decade (or more) for benchmarking algorithms and techniques, they have become outdated.

Given the constantly evolving nature of network traffic, it is imperative to keep intrusion detection datasets up-to-date with newly emerging attack patterns and recent normal traffic patterns. Doing so is crucial to building accurate intrusion detection models for real-world security. More than simply detecting novel intrusion patterns is required for robust IDSs. Instead, they must quickly detect intrusion patterns to allow for effective countermeasures. Therefore, datasets that include transient behavior observed in contemporary traffic within datasets could be highly beneficial.

Our primary focus here is to discuss the main datasets used for building and evaluating data stream-based IDS and the different data types.

1) DATA TYPES

Intrusion detection datasets may contain different types of traffic data. The most fine-grain data type, *packet data*, can be captured from network devices or hosts through popular programs such as tcpdump, Wireshark, Nmap, and others. Such programs use packet capture libraries (PCAP) implemented by several operating systems to collect a copy of transmitted packets. Packet data allows the discovery of traffic patterns by inspecting header fields of any protocol (e.g., IPv4, IPv6, TCP, UDP, ICMP, and others) carried out by packets. Among the reviewed papers, packet headers are the data source in [10], [30], [32], [38], [39], [40], [44], [56], [57], [69], [72], [74], and [77].

Flow level data contain a set of features (e.g., counters, statistics) extracted from network communication between hosts or services. The NetFlow protocol was initially introduced as a feature implemented by Cisco routers and switches to collect management and statistical flow information. Later, similar technologies were implemented by other manufacturers of networking devices. A flow is an abstraction for a unidirectional sequence of packets transmitted between two end-points (i.e., that share the same source and destination addresses, source and destination ports, and the same type of service). Flow level data is the most common data type in the reviewed studies, being used in [9], [11], [12], [14], [15], [31], [33], [34], [36], [37], [41], [42], [43], [45], [46], [47], [48], [50], [51], [52], [53], [54], [55], [58], [59], [60], [61], [62], [66], [67], [68], [70], [71], [73], [75], and [76].

Other data types may be used to compose datasets, including activity logs such as sessions, service requests, etc. For instance, [63] combines measurements of voltage, current, and the status of system devices of an industrial system with data from the control panel and Snort alerts.

Figure 5 summarizes the studies by data type. It is possible to see that most data stream-based IDS references are associated with flow-level data. This behavior seems to be an increasing trend in the last years due to several factors [78], [79]: i) inspecting the complete payload is computationally costly, ii) deep packet inspection is challenging to implement when network traffic is encrypted, iii) flow data can be collected from multiple locations across the network without

TABLE 4. Summary of data stream-based IDS studies ordered by publication year (cont).

Ref.	Year	Data type	Data preprocessing	Data reduction	Data mining	Interpretation and evaluation
[67]	2020	flow data	normalization conversion	sample	unsupervised binary informed (change detector no feedback	streaming accuracy, F1, running time
[68]	2020	flow data	-	sample	unsupervised binary blind no feedback	streaming accuracy, recall, FAR, AUC, ROC curve
[69]	2020	packet data	-	-	single, unsupervised binary informed (change detector) feedback (total)	holdout accuracy, recall, running time
[15]	2020	flow data	-	feature sample	single, ensemble binary informed (change detector) feedback (total)	streaming precision, recall
[70]	2020	flow data	-	sample	single, ensemble binary informed (change detector) feedback (total)	streaming accuracy, running time
[71]	2021	flow data	-	sample	single, ensemble binary informed (change detector) feedback (total)	streaming accuracy, Kappa
[72]	2021	packet data	-	-	single binary blind feedback (total)	streaming FAR, ROC curve
[73]	2021	flow data	-	feature sample	unsupervised binary blind no feedback	holdout accuracy, precision, recall, FAR
[9]	2021	flow data	-	sample	single, semi-supervised multiclass informed (change detector) feedback (partial)	streaming F1
[10]	2022	packet data	-	sample	single, ensemble binary blind feedback (total)	holdout F1
[74]	2022	packet and flow data	-	feature sample	single multiclass blind no feedback	recall, FAR accuracy, precision, F1
[75]	2023	flow data	-	feature sample	single, ensemble multiclass blind feedback (total and partial)	streaming accuracy, precision, specificity, recall, F1
[76]	2023	flow data	-	-	single, ensemble multiclass informed (change detector) feedback total	streaming accuracy, F1, AUC
[77]	2023	packet and flow data	-	feature	ensemble binary blind, informed (change detector) feedback (total and partial)	streaming recall precision, FAR

TABLE 5. Data stream-based IDS studies organized by data type.

Data type	Example of features	References
Packet level	Destination port, header length	[10], [31], [33], [39]–[41], [45], [57], [58], [70], [73], [75], [78]
Flow level	Flow duration, total number of packets	[32], [34], [42], [46], [47], [51], [53], [59], [61], [68], [69], [72] [12], [14], [15], [49], [56], [60], [62], [63], [67], [71], [74] [9], [11], [35], [37], [38], [43], [44], [48], [50], [52], [55], [66], [74] [54], [75]–[78]
Event log	Sensor measurements, failed attempt of logins	[64]

any additional cost, and iv) several recent datasets provide flow data adjusted to ML algorithms. However, it is important to note that some studies [13], [77] have shown that it is possible to develop a data stream-based IDS using

packet-level data (packet header) and achieve an equivalent detection performance to a flow-level IDS. This result can be used to explore other implementations of data stream-based IDS without converting packets to flows.

2) DATASETS

Typically, IDS evaluation is conducted using test datasets. These datasets, whether standard benchmarks or custom-made, are constructed using the collection techniques and data types mentioned earlier. They play a crucial role in the field of cybersecurity, providing a foundation for the development and testing of intrusion detection systems. The configurations of the classes in the dataset, as mentioned in III, are diverse. If the dataset includes both normal and attack instances, it usually maintains the actual ratio of normal usage and attack attempts. The features present in the dataset can vary depending on the type of attack one wants to defend against. For instance, a Web Application IDS will require additional data in the form of the requests made by the user to classify the behavior correctly. In most of these datasets, the majority of samples are background traffic, with attacks injected according to the specific purpose (e.g., DDoS attacks will have many malicious samples in sequence). Next, we present the datasets used among the 49 related works, providing information about availability, attack types, and data sources.

The most widely used is the KDD99, a dataset created for a knowledge discovery competition where the participants had to build a model to detect network intrusions. The models should distinguish between malicious and normal network traffic. This dataset contains attacks simulated in a military network environment, which can be grouped into four big groups: Denial of Service (DoS), user to root (u2r), remote to local (r2l), and probing (probe). KDD99 was built from network flows. Therefore, it contains identification features from the basic TCP connections and a set of generated or extracted features that uses a sliding window to calculate. Despite its age, it is still being used today [30], [31], [32], [33], [34], [36], [37], [41], [42], [43], [45], [48], [49], [51], [54], [59], [61], [62], [66], [67], [70], [74], [76]. Given its widespread usage, the KDD99 dataset has been the subject of extensive investigation and improvement. Researchers have found ways to enhance the dataset, such as reducing the sample size (i.e., exclusion of redundant samples). The NSL-KDD dataset, which has overcome this issue, has been adopted in numerous studies.

The CTU-13 is a collection of public cybersecurity datasets that are not only widely used for benchmarking but also hold practical relevance. This dataset represents several scenarios of Botnet attacks, with the following botnets being used: Neris, Rbot, Virut, Menti, Sogou, Murlo, and NSIS.ay. The datasets are available in the packet captures (PCAPs) format, allowing the researchers to choose their preferred data type between PCAPs and flow data. The CTU-13 datasets contain four labels: background, botnet, C&C channels, and normal.

MAWIFlow is a dataset that contains real and labeled network traffic records with 158 features each. It was extracted from 15-minute-long daily traces spread over a year of real network traffic. MAWIFlow comprises over 6 billion network flows with almost 8 TB of data. This dataset presents network anomalies like portscan, network scan, DoS, and DDoS. The studies in the literature that used this dataset mention its extensive size and span-period as strong points [10], [65], [72].

Researchers from Kyoto University created the Kyoto 2006+ dataset. The authors used honeypots composed of many devices like servers, printers, and IP cameras, which capture malicious data using three software, making the dataset multiclass and multilabel. This dataset was used in [14] and [46].

Many applications require specialized datasets in the field of cybersecurity. For example, Web Applications need the request header and body. Two datasets contain this kind of data: ECML-PKDD 2007 HTTP and CSIC HTTP 2010. Both are public and multiclass and were utilized in [39]. The first (ECML-PKDD) was created for a challenge to build a model for multiclass contextual classification in seven different attack patterns. The latter (CSIC) was created based on real requests to an e-commerce application with synthetic anomalous traffic.

ISCX-IDS, CIC-IDS2017, and CSE-CIC-IDS-2018 were built by the Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick. The ISCX-IDS intrusion detection evaluation dataset consists of packet captures that present normal activity and four different types of attacks: Network infiltration, HTTP DoS, DDoS with Botnets, and Brute Force SSH. This dataset was used in [50] and [76]. The CIC-IDS2017 dataset contains benign and the most up-to-date common attacks, resembling true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter with labeled flows based on the time stamp, source, destination IPs, source and destination ports, protocols, and attack (CSV files). The implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. Authors in [9], [68], [74], [75], [76], and [77] used this dataset. The CSE-CIC-IDS-2018 includes seven different attack scenarios: Heartbleed, Brute-force, Web attacks, DoS, DDoS, Botnet, and network infiltration from inside. One notable difference from the CIC-IDS2017 dataset is the network topology, with the attacking infrastructure consisting of 50 machines, while the victim organization includes 420 machines and 30 servers. This dataset was used in [76] and [77]. Besides these public datasets, the rest of the datasets used to evaluate the built models in related works were either created by the authors or used by only one work [35], [38], [40], [44], [45], [50], [55], [56], [57], [58], [63], [64], [71],

TABLE 6. Data stream-based IDS studies organized by datasets.

Dataset	Year	Origin	Labels	References
KDD99	1999	University of California	Multiclass	[31], [32], [34], [42], [46], [68] [49], [62], [63], [71] [43], [44], [52], [55], [60], [67] [33], [35], [37], [38], [75], [77]
NSL-KDD	2009	Tavallaee	Multiclass	[12], [47], [48], [53], [54], [70]
CTU-13	2011	CTU University	Multiclass	[11], [15], [61]
MAWIFlow	2019	Fukuda Lab Pontifical Catholic – University of Parana	Binary	[10], [56], [73]
Kyoto2006+	2009	Kyoto University	Binary	[14], [47]
ECML-PKDD	2007	Data mining competition	Multiclass	[40]
CSIC	2010	Spanish Research National Council	Binary	[40]
ISCX-IDS	2012	Universify of New Brunswick	Multiclass	[51], [77]
CIC-IDS2017	2017	Universify of New Brunswick	Multiclass	[9], [69], [75]–[78]
CSE-CIC-IDS-2018	2018	Universify of New Brunswick	Multiclass	[77], [78]
MISC	–	–	–	[36], [46], [51], [58] [39], [59], [72], [74] [45], [64], [65] [41], [56], [57]

[73]. These works can be found under the MISC category in Table 6.

Table 6 summarizes the datasets used by data stream-based IDS studies. Most studies still use KDD99 or NSL-KDD to evaluate intrusion detection systems. Our results are consistent with other surveys investigating the distribution of ML-based IDS datasets. Reference [80] shows that almost 70% of studies (85 IDS articles) use either KDD-99 or NSL-KDD. A similar ratio (82%) was found in [81]. This result indicates that the IDS research community is still unaware of the problems associated with these datasets. As discussed in [82], DARPA-derived data, used in both KDD99 and NSL-KDD, is nearly 25 years old and seems unlikely to contain event distributions (malicious or benign) that are representative of today's deployment contexts. Therefore, both datasets should be considered obsolete for current IDS research purposes. It would be important to see data stream-based IDS studies evaluated on more sophisticated and well-maintained datasets such as UNB CIC IDS,[4] UNSW-15,[5] and Stratosphere Laboratory Datasets.[6]

B. DATA PREPROCESSING

Data preprocessing is a preparation procedure applied to data before using it in the data mining step. This step aims to increase the data quality and ensure it is representative and reliable regarding the task one wants to perform. Despite the importance of this step for data mining algorithms, several studies have either not applied preprocessing techniques (using the data as provided by the datasets) [9], [10], [11], [12], [30], [33], [37], [41], [42], [47], [48], [51], [53], [54], [62], [64], [65], [66], [69], [74], [75], [76], [77] or not mentioned it [35], [38], [55], [56], [58], [71].

TABLE 7. Data stream-based IDS studies organized by data preprocessing techniques.

Technique	References
Normalization (scaling)	[14], [31], [37], [46], [58], [60], [68]
Conversion (one-hot encoding)	[14], [37], [44], [47], [60], [68]
Anonymization	[61]
Not applied	[9]–[12], [31], [34], [42], [70] [43], [48], [52], [54], [55], [66], [67] [38], [49], [63], [65], [75]–[78]
Not mentioned	[36], [39], [56], [57], [59], [72]

Many factors can influence this behavior. For example, achieving a good result without performing any preprocessing technique might be enough for authors to skip this step altogether. Another reason is the intersection between two extensive areas: network security and ML. While network security researchers have domain expertise, they may lack an understanding of adequately applying preprocessing techniques, leading to using the "raw" dataset. On the other hand, ML researchers may not understand the data in the datasets, which hampers the effectiveness of applying preprocessing techniques as the process turns into a implict trial and error.

[4] https://www.unb.ca/cic/datasets/index.html
[5] https://research.unsw.edu.au/projects/unsw-nb15-dataset
[6] https://www.stratosphereips.org/datasets-overview

FIGURE 2. Taxonomy for the preprocessing step of the data stream-based IDS studies.

Three of the important preprocessing tasks found in the surveyed works are: i) standardize the data to avoid unwanted bias, ii) convert nominal data into binary data, and iii) data anonymization (IP address). Table 7 shows the summary of studies that employed such techniques.

While the importance of preprocessing techniques for enhancing classification performance is widely acknowledged, it is surprising to note that a significant number of the surveyed studies either did not employ such techniques or failed to disclose the ones they used. This lack of attention to the preprocessing step is a notable gap in the current research landscape. Among the studies that utilized preprocessing techniques, the focus was primarily on standardizing the data and converting attributes. Notably, to the best of our knowledge, none of the studies explored the potential benefits of incremental preprocessing techniques.

Scaling is one of the most used normalization techniques that transform numeric features to eliminate big discrepancies in their values. For example, when calculating Euclidean distances, if one feature has a 100x bigger range than the rest, it will have a bigger weight in the distance calculation. Scaling techniques were used in [14], [30], [36], [45], [57], [59], and [67].

One-hot encoding is one of the most used techniques to transform nominal data into binary fields, where each possible nominal value of a feature will become a binary feature. Authors in [14], [36], [43], [46], [59], and [67] use this technique.

Other studies might have used normalization or conversion methods. However, we have not found an explicit mention of them. We believe that this is a bad practice when developing ML-based IDS since it harms reproducibility.

Among the surveyed papers, only [60] explicitly lists data anonymization as a process performed in the data. In this case, the authors developed a mechanism based on flow aggregation to anonymize specific IP addresses. Although most studies used the samples provided by the dataset, making this preprocessing technique implicit, some studies use proprietary datasets and do not mention anonymization techniques.

Notably, the surveyed studies did not address two fundamental preprocessing techniques in machine learning: noisy data and missing values. There could be many reasons for this omission, such as the possibility that the data used for the Intrusion Detection problem is naturally free of noise or missing values. Additionally, it is plausible that the authors resolved these issues with more sophisticated techniques, such as feature reduction, aggregation, or extraction.

It's worth noting that intrusion detection is highly unbalanced, with the attack class significantly underrepresented. While oversampling and undersampling techniques have been employed in batch-based IDS to address class imbalance, their application in data stream scenarios is a pressing and crucial question that has yet to be thoroughly explored for IDS. This underscores the importance of future work in this area.

C. DATA REDUCTION

Given the enormous amount of data generated by computer networks, data reduction is an important step in the KDD process that aims to remove irrelevant and correlated information and reduce the training time of the models.

In essence, data reduction means removing something based on criteria. We consider that the filter can be performed in either "axis" of the data, which means one can filter the samples [14], [70], or the features [14]. For example, authors may want to focus on only one attack type from a dataset that contains multiple attack classes. In this case, they must filter the other attack types from the dataset. On the other hand, if the authors notice that some features are deteriorating the classification performance or increasing the processing requirements without providing predictive performance benefits, they may want to remove those features from the data.

We found the following data reduction approaches in the surveyed works: Feature Reduction and Sample reduction. The feature reduction consists of selecting a set of features and eliminating mainly irrelevant or correlated data, thus improving the models' performance. The works used the following types: manual, filter, and wrapper. On the other hand, sample reduction consists of selecting a representative sample of the population (here, named sampling) or simply aggregating the data. Figure 3 represents the taxonomy of data reduction, and Table 8 organizes each article in its corresponding data reduction category.

1) FEATURE REDUCTION

There are several types of feature reduction techniques that are widely used in machine learning models [83]. However,

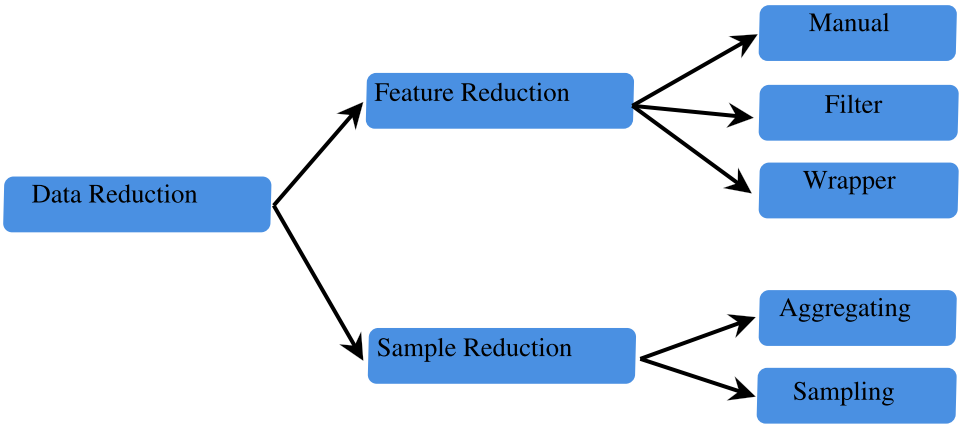

FIGURE 3. Taxonomy for the data reduction step of the data stream-based IDS studies.

TABLE 8. Data stream-based IDS studies organized by data reduction approaches.

Data Reduction	Type	References
Feature Reduction	Manual	[11], [15], [32], [42], [74]–[76], [78]
	Filter	[46], [47], [62], [78]
	Wrapper	[53]
Sample Reduction	Aggregating	[31], [32], [34], [46], [47], [51], [53], [68], [72], [15], [42], [49], [59], [61], [62], [69], [71], [74], [12], [14], [54], [55], [60], [63], [66], [67], [33], [35], [37], [38], [43]–[45], [48], [52], [9]–[11], [39], [50], [56]
	Sampling	[15], [31], [34], [39], [46], [47], [49], [62], [68], [14], [54], [55], [60], [63], [66], [67], [71], [33], [35], [37], [38], [40], [41], [43], [45], [52], [11], [56], [75], [76]

only a subset of these techniques is used in the ML-based IDS literature.

As shown in Table 8, manual feature reduction is the most common method, involving the manual removal of features. Some features are removed due to the variable type; for example, in [41], categorical variables were removed because the algorithm only works with numerical data. In other cases, a specialist manually selects the most relevant information and removes the irrelevant features, as seen in [15], [31], [73], and [77].

Another common method of selecting features is the application of filters, which occurs before the data mining step. Usually, the features are ranked based on statistical metrics or correlation with the target feature. References [45], [46], and [61] use this method. References [45] and [61] use the principal component analysis (PCA) technique, and [46], [77] uses correlation and consistency-based feature selection techniques.

The study [52] uses the wrapper method, which consists of evaluating a subset of features by training a machine learning model and evaluating its results. Ultimately, the set of features with the best performance is chosen.

There are other methods for feature reduction, such as the embedded type. This technique selects the optimal set of features while the model is being trained. Classification algorithms like decision trees are embedded approaches because they choose the most relevant features during tree building. Although several IDS proposals use tree algorithms (such as RandomForest, Hoeffding Tree, and ensemble methods) [10], [11], [14], [15], [49], [55], [65], they do not discuss which features are selected during model building.

2) SAMPLE REDUCTION

Sample reduction is a step that aims to remove irrelevant samples to speed up the training process and reduce computational costs without degrading classification performance. It is an important process for building ML-based IDS due to the high volume and velocity of network packets. The surveyed studies use two types of sample reduction: sampling and aggregating.

Aggregating, in the scope of network traffic management, is the process of summarizing packets in an information flow. In this case, we no longer use raw packet information but statistics of a set of packets. When using flow data, one

has to summarize all packets exchanged in a connection (i.e., measure statistics like bytes sent and bytes received and derive features from them). Flows can be much smaller than raw packets, as they do not need to store the payload information, making the data more concise and training much faster than the original information. A disadvantage would be the loss of some information due to aggregation. However, it is widely used, as seen in Table 8. Sketch, another way to summarize data, is a family of probabilistic data structures employing hashing technology for summarizing traffic data used in [38] to provide the network summary.

Another type of sample reduction is sampling, which involves extracting samples for model training. Table 8 shows that many authors use this strategy, varying between percentage sampling and fixed amounts. The choice between these techniques is typically based on the analyst's experience rather than the technique itself. Another approach combines both strategies: first, aggregating the data samples and then selecting a subset of them. Several studies [11], [45], [48] employ this method.

D. DATA MINING

Data mining is the main step in the ML-based IDS development process. Here, preprocessed packet or flow-level data (for NIDS) is fed into algorithms to detect malicious events. Our proposed taxonomy differs from previous surveys. First, we discuss the nature of the algorithm in terms of the learning aspect (supervised and unsupervised) and the number of algorithms (single classifier and ensemble). Next, we study the classification task, in other words, whether the detection model provides only binary classification or multiclass. Finally, we discuss an aspect specific to stream-based IDS: the model update. For model update, we consider different aspects: i) Availability of labeled instances; ii) Delay in obtaining labeled instances to update the model; iii) Type of update, i.e., if the strategy to update the model occurs online or offline; iv) The mechanism triggers the update, i.e., informed by a change detection method or blind (implicit), at regular time intervals.

1) ALGORITHM

Different strategies have been used to handle the intrusion detection task considering the algorithm aspect. First, we have the learning aspect: supervised or unsupervised. Next, we have the number of classifiers: single or ensemble. Figure 4 shows the proposed taxonomy to organize the algorithms used in the data mining step, and Table 9 summarizes the studies according to the characteristics of the data mining algorithms.

Supervised algorithms aim to train a decision model using a set of labeled instances. In this case, the dataset can contain examples of normal and attack classes, where different attack types can be described, and each one is considered a different class. Most data stream-based IDS studies use this strategy or a hybrid one (semi-supervised), as shown in Table 9.

On the other hand, few data stream-based IDS studies adopt unsupervised learning solutions. In general, unsupervised models treat the intrusion detection problem as an anomaly detection task [32], [40], [42], [48], [56], [73]. In this case, a set of non-malicious examples (normal traffic) is used to induce a decision model. The examples not explained by this model are classified as an anomaly (or attack). Some work has recently used a different approach for unsupervised learning in data-stream IDS. They use unsupervised models, like clusters, but on a labeled dataset. In this case, each class of the problem is represented by a set of clusters, and each cluster is associated with a class. During the test phase, new clusters (different attack types, for example) can emerge, or clusters can evolve without using labeled instances, thus facilitating the model update.

Another important factor in data stream classification algorithms is the number of classifiers used. There are two basic approaches: using a single classifier or an ensemble of classifiers. Single-classifier methods involve training only one decision model, which must be updated as new examples arrive [10], [11], [12], [15], [31], [36], [37], [38], [44], [46], [49], [51], [53], [54], [55], [56], [59], [61], [63], [65], [69], [70], [71], [72], [74], [75], [76]. On the other hand, ensemble classifiers involve building a set of classifiers used to classify a new instance [10], [14], [15], [34], [39], [41], [49], [50], [52], [55], [59], [60], [65], [70], [71], [75], [77]. Ensemble methods can be divided into two types: homogeneous and heterogeneous. The homogeneous ensemble is composed of members that share a single base learning algorithm. In contrast, the heterogeneous ensemble consists of members with diverse base learning algorithms, including K-NN, SVM, and tree-based algorithms. It is interesting to see that most proposals focus on homogeneous ensemble. Due to the nature of network traffic data, using heterogeneous ensemble methods might be an interesting approach for data stream-based IDS.

While single-classifier approaches facilitate model updates and allow online updating, ensemble approaches must train a new model, which must replace one of the ensembles (e.g., the oldest or the one with the worst performance). It is important to note that although ensembles have achieved good performance in the IDS scenario, training a new model is a time-consuming task that must be done in batch mode.

When considering the algorithms used, one can note a variety of proposals. Recently, several literature studies have used decision trees, especially Hoeffding trees and their variations (including ensembles), since they are incremental algorithms [10], [11], [15], [49], [55], [61], [65], [70], [71], [77]. Other examples of incremental algorithms include online sequential extreme learning machine (OSELM), feature adaptive OSELM (FA-OSELM), and knowledge preservation OSELM (KP-OSELM) [76]. It is important to see that the number of data stream-based IDS studies that use incremental algorithms is still low compared to the other approaches. Finally, other algorithms include neural

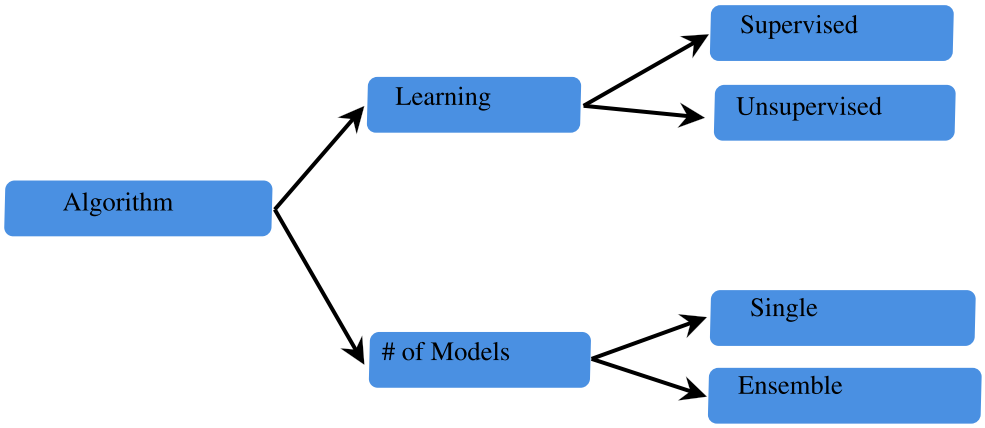

FIGURE 4. Taxonomy for the algorithms (data mining step) of the data stream-based IDS studies.

TABLE 9. Data stream-based IDS studies organized by the characteristics of data mining algorithms.

# of models and learning task	Algorithm	References
single supervised	tree based models	[10], [11], [15], [50], [56], [62], [66], [71], [72]
	neural network models	[47], [60], [70], [76]
	SVM	[12], [32], [37], [38], [54], [62], [71], [75]
	time series	[39], [64], [73]
	logistic regression	[62]
	K-NN	[55], [57], [71], [76]
	others	[45], [52], [77]
ensemble supervised	heterogeneous	[10], [60], [66], [71]
	homogeneous	[14], [15], [42], [51], [53], [61], [66], [72]
semi-supervised	single	[10], [35], [40], [50], [56], [76], [78]
		[9], [38], [44], [54]
unsupervised	clustering based	[31], [34], [36], [46], [49], [58], [59], [69]
		[14], [33], [37], [54], [63], [67], [70], [74]
	neural network models	[12], [48], [55], [65], [68]
	anomaly detection	[33], [41], [43], [49], [57], [74]

TABLE 10. Data stream-based IDS studies organized by the number of classes of the data mining step.

# of classes	References
Binary	[10], [11], [32], [39], [42], [47], [51], [58], [59], [61], [68], [69], [72], [73] [14], [37], [40], [41], [43], [44], [48], [52], [55], [60], [64]–[66], [71] [15], [35], [50], [56], [57], [70], [74], [78]
Multiclass	[9], [12], [33], [34], [38], [45], [47], [49], [53], [54], [60], [62], [63], [67] [75]–[77]

networks [46], [59], [69], SVM [12], [31], [36], [37], [53], [61], [70], logistic regression [61], and KNN [54], [70].

2) NUMBER OF CLASSES (BINARY X MULTICLASS)

Most data-stream IDS studies treat the intrusion detection task as a binary classification task, in which new instances are classified as normal or attack [10], [11], [14], [15], [31], [34], [36], [38], [39], [40], [41], [42], [43], [46], [47], [49], [50], [51], [54], [55], [56], [57], [58], [59], [60], [63], [64], [65], [67], [68], [69], [70], [71], [72], [73], [77]. While this is an effective strategy, it can also be valuable to identify the type of attack in various contexts. Therefore, some approaches treat the intrusion detection task as a multiclass classification problem, including a normal class and separate classes for each type of attack [9], [12], [32], [33], [37], [44], [46], [48], [52], [53], [59], [61], [62], [66], [74], [76]. One of

the reasons for using multiclass approaches is that different types of attacks may require specific actions. Additionally, new attacks can differ from the already known ones (such as different DoS types), and appropriate strategies must be created to mitigate them. Table 10 shows the studies according to the number of classes used by the data mining algorithms.

Another important point to be addressed by both binary and multiclass classification approaches concerns the evaluation of models. In the case of multiclass models, different error types need to be thoroughly investigated. Some types of errors may be more critical than others. For instance, classifying an attack instance as an incorrect type of attack has less impact than classifying this instance as normal (non-malicious). Therefore, it is crucial to consider appropriate evaluation measures for the problem.

Finally, domain experts can identify which attack classes the model performed better when considering the multiclass classification task. They can also determine which types of attacks the model is not prepared to deal with and how different decision models can handle different types of attacks.

3) MODEL UPDATE

Model updates are crucial in data stream-based IDS, as network behavior can change over time due to new attacks or shifts in non-malicious traffic. To maintain an accurate decision model, it is essential to use the most recent instances for updates. Both instance-incremental and batch-incremental learning approaches are used for this purpose. Instance-incremental methods, like incremental decision trees, update the model with a single instance at a time. Batch-incremental methods, such as ensembles of trees, update the model using batches of training instances. This flexibility allows us to choose the most suitable approach for the intrusion detection scenario, giving us control over the model update process.

Regarding the strategy for deciding when to update the decision model, one approach is to update it every time a new labeled data instance arrives. In contrast, other approaches incorporate a drift detector method and update the model upon detecting a drift. In both cases, labeled instances are a prerequisite for supervised model updates. However, a major challenge in data stream-based IDSs is managing real-world scenarios in which not all data instances will be labeled, and labels will often be obtained with a delay.

Figure 5 shows the main aspects to be considered in the model update. The update occurs with the arrival of new data instances after the decision model classifies them. Different types of feedback can be considered in the model update: total, partial, and no feedback. Table 11 summarizes the studies according to the model update feedback.

Most works assume that after their classification, the instances will have their label available [10], [11], [12], [14], [15], [34], [36], [37], [38], [39], [41], [46], [49], [50], [54], [55], [59], [61], [63], [65], [69], [70], [71], [72], [75], [76], [77] (total feedback). Although this was the approach used by the first data stream classification works, it becomes impractical for IDS scenarios since having a domain expert available to label such instances is impossible. On the other hand, some works update the decision model without using labeled instances (no feedback). Thus, the model is updated using only the data [12], [14], [30], [32], [33], [35], [36], [40], [42], [45], [47], [48], [53], [54], [56], [57], [58], [62], [64], [66], [67], [68], [69], [73], [74]. A common approach is to use cluster-based models that can update summary statistics such as mean and standard deviation without knowing the true label.

While unsupervised strategies can be interesting because they do not require labels, the predictive model's performance can generally suffer as many changes in the data flow happen, especially with the emergence of new types of attacks (zero-day attacks, for example). Recent approaches to address the problem of decision model updates assume that a subset of labeled instances may be available (partial feedback). Semi-supervised approaches can be used here to evolve the model using both labeled and unlabeled instances [9], [37], [43], [53]. Another category of strategies involves using active learning techniques [14], [31], [49], [52], [53], [60], [77], which aim to choose the best instances to be labeled by the expert for updating the model and improving its predictive performance. However, an important point yet to be explored is that the domain expert is not always an oracle. The instances they label may contain errors that directly impact the model.

Even when a domain specialist is available to label instances, an important aspect that needs to be explored is the delay in labeling instances. Domain experts are typically involved in various daily tasks, and labeling packets or network flows is challenging and takes time. However, most approaches assume that the label of instances will be immediately available [10], [11], [12], [14], [15], [15], [34], [36], [37], [38], [39], [41], [46], [49], [50], [54], [55], [59], [61], [63], [65], [69], [70], [71], [72], [75], [76]. Consequently, few studies [14], [77] have considered that instances used to update the model may arrive with a delay, which can directly impact the identification of similar instances. More details on evaluating data streams with delayed labels can be found in [84].

When developing IDS for real-world scenarios, addressing the complexity and time required for updating the model is crucial. Many data stream algorithms have both an online and offline phase. The offline phase is a batch task requiring more computational. The model update occurs during the online phase when the network traffic is being classified. Updating a model can be a low-complexity task, such as creating a new branch in a tree using the Hoeffding Tree algorithm [11], [15], [65], [71]. However, it becomes more complex when training a new model for an ensemble of classifiers [10], [59], [65], [70]. The literature still needs to explore the complexity of updating decision models in data stream-based IDS.

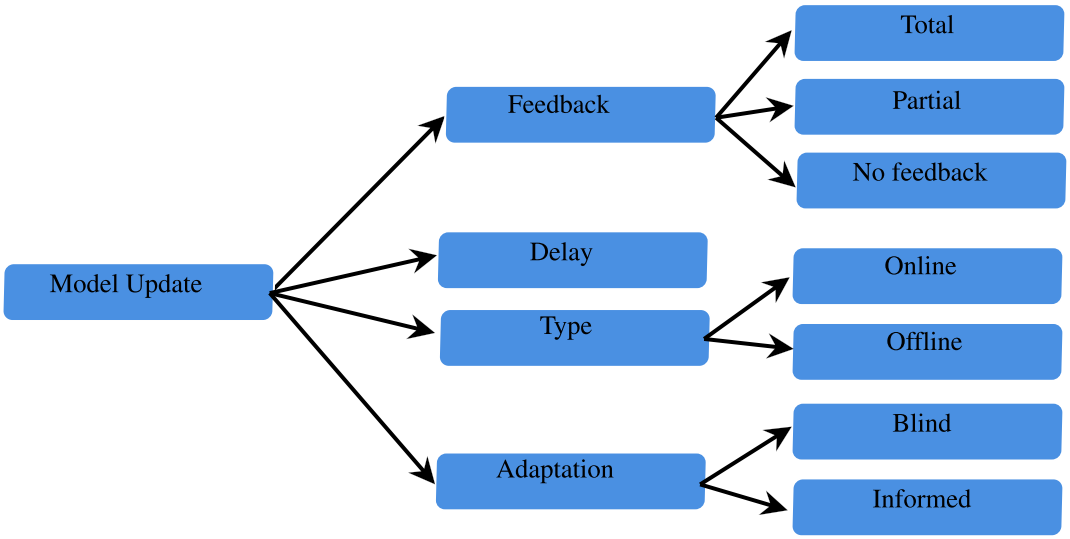

FIGURE 5. Taxonomy for the model update (data mining step) of the data stream-based IDS studies.

TABLE 11. Data stream-based IDS studies organized by the model update feedback.

Feedback	Idea	References
Total	Labels readily available for model update	[10], [11], [15], [47], [56], [62], [66], [70]–[72] [12], [37]–[39], [60], [73] [14], [15], [40], [42], [51], [55], [64] [35], [50], [76], [77]
Partial	Subset of labels may be available	[9], [14], [32], [38], [44], [50], [53], [54], [61], [76], [78] [31], [34], [36], [46], [49], [58], [59], [69], [74]
No feedback	No labels available for model update	[14], [37], [54], [63], [67], [70] [12], [33], [55], [65], [68] [41], [43], [48], [57], [75]

Another important aspect of updating the decision model concerns the timing of the update. In data stream classification, two approaches are used: blind (implicit) and informed. Table 12 summarizes the studies according to the type of update. Implict approaches assume that the decision model will be updated whenever new instances are available, regardless of any indication of a change in the probability distribution [10], [11], [12], [14], [30], [31], [32], [33], [34], [36], [37], [39], [44], [46], [47], [49], [51], [52], [53], [55], [56], [57], [58], [60], [61], [64], [65], [66], [68], [72], [73], [74], [75], [77]. Some works, like [65], set a periodicity for such updates. On the other hand, a group of works uses guided strategies (informed), which update the model only when there is evidence of a change in the data [9], [15], [35], [38], [40], [41], [42], [43], [45], [48], [49], [50], [54], [59], [62], [63], [67], [69], [70], [71], [76], [77]. In this case, the appearance of a new attack would be an example of a scenario in which the model should be updated. Algorithms in this category generally use a change detection strategy. Widely used approaches in the literature, such as OzaBagAdwin, use the Adwin method for change detection [10], [15], [65], [70], [71], [77].

E. INTERPRETATION AND EVALUATION

Regardless of the learning task used (e.g., supervised or unsupervised), evaluating how well the built model can generalize its representation is essential to classifying new data. In addition, evaluation stages aid in detecting problems in the model before deploying it in the production environment. Numerous strategies exist to perform this task, which can vary significantly in the evaluation method or measure used. Next, we present the techniques used in the 50 reviewed papers to validate their models.

1) EVALUATION METHODOLOGY

The classical method to evaluate a model in batch learning uses a K-fold Cross-Validation (CV) strategy that divides the dataset into k folds, creating k different configurations (training and test sets) to evaluate the model. This means that the algorithm uses $k - 1$ folds to train the model and test it using the one left out, repeating enough times to use every fold as a test set once. In this method, a significant difference between a specific fold and the global (average) accuracy indicates a model that overfits the training data.

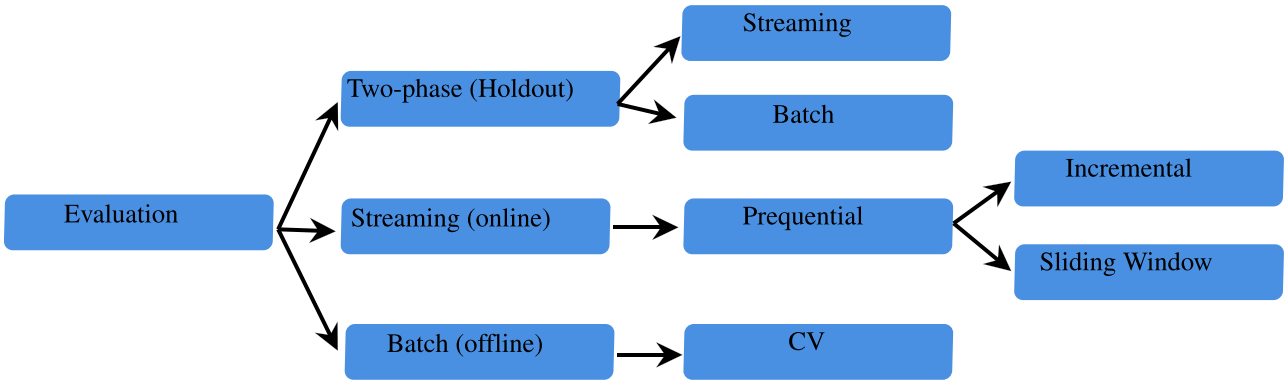

FIGURE 6. Taxonomy for the interpretation and evaluation step of the data stream-based IDS studies.

TABLE 12. Data stream-based IDS studies organized by the type of the update.

Model Update	Goal	References
Blind	Update whenever new instances are available	[31], [32], [34], [47], [53], [58], [73] [14], [59], [61], [62], [65], [69], [74] [12], [48], [52], [54], [66], [67] [11], [33], [35], [37], [38], [40], [45] [10], [50], [56], [57], [75], [76], [78] [9], [36], [46], [51], [68], [72]
Informed	Update when there is a change in the data	[15], [39], [42], [49], [70], [71] [43], [55], [60], [63], [64] [41], [44], [50], [77], [78]

The CV was the only technique used in some studies [30], [42], [44], [46], [66], [75]. Although using CV in data streaming scenarios is usually not a good idea because of the limited processing time and memory restrictions, several authors in the field have used ingenious ways to leverage the advantages of the CV while adhering to the data streaming scenario requisites. For example, the authors in [48], [60], and [51] used a two-phase approach to create an offline phase where CV can be applied before using the rest of the data in a streaming fashion (online phase).

Two entirely separate dataset files are expected in batch scenarios: the train set and the test set. While the two files contain the same features and data distribution, the idea is that data from the test set should not exist in the train set. This strategy is known as train/test in the batch community, while the online community usually calls it holdout, and it guarantees that the model uses unseen data for evaluation. Initially, the Holdout technique was intended as a method that continuously provides instances for training with eventual chunks of data being used for testing once in a while [85]. Nonetheless, some of the reviewed papers use holdout with a single division, transforming the method into a two-phase one. In such cases, the second phase is the test phase, which usually employs prequential evaluation.

Given its popularity, there is a wide variance in the proportion used in the splits, which we will discuss next. In [14] and [51], the authors use 10% of the dataset to train the model and the other 90% to test (i.e., a 10/90 train/test split). It is also possible to find more evenly distributed splits in the literature like an 80/20 split [76], 75/25 [53], 70/30 [60], 60/40 [34], [36] and even a 30/70 holdout [48]. In [74], a distinct form of holdout is employed. To assess the efficacy of an incremental transfer learning approach, the authors randomly chose percentile subsets of data for various test iterations. There are numerous reasons why each author uses a different split, such as data that is hard to model or algorithms with a high degree of complexity, which can increase the amount of training data needed, as it takes longer for the model to capture the behavior pattern correctly.

Regarding purely streaming evaluation methods, there is the Prequential [86], also known as Interleaved-Test-Then-Train. In this case, instances usually arrive individually, and then the model uses each instance for two processes. First, the model outputs a prediction, and second, the model updates itself using a single instance of the stream. If the algorithm is not purely incremental, a widely used strategy is to create windows where the "batch" processing can happen but has reduced latency; such strategy was named windowed prequential in [86]. This strategy allows replicating established offline methods to build and evaluate the models with wrappers, reducing development time. Another benefit of this approach is the capability to use algorithms more realistically

without dealing with incremental restrictions. Unfortunately, most studies that use the prequential evaluation method assume that all labels are instantly available, which is too optimistic considering real scenarios. For more realistic cases, some studies try to address the problem of delayed labeling and semi-supervised learning [9], [10].

The authors in [11], [15], [38], [40], [41], [49], [51], [58], [63], [68], [70], and [71] used prequential evaluation without the addition of windows or fading factors, while the windowed prequential evaluation method was used in [9], [43], [47], [56], [59], [67], [72], and [77]. Similar to holdout, different algorithms and datasets require different window sizes to achieve the best performance. The window varies between 100 and 5000 instances among the current selection of works.

Finally, some studies did not provide information regarding this aspect [12], [32], [35], [39], [50], [52], [64], [65] and therefore were not included in any of the previous categories.

2) EVALUATION MEASURES

Depending on the objective of the task, the bias of the problem, the type of model employed, and the characteristics of the data, specific measures will be better in assessing performance. For example, the majority classifier assigns all instances to the most common class. This can achieve high accuracy in datasets with significant class imbalances, as is often the case with intrusion detection datasets. Therefore, it would be better to use measures like Recall or Precision, which can evaluate the performance of the minority class.

One can divide the measures according to the extraction method and the measures themselves to better understand and organize them. The extraction methods are relative to the granularity used to calculate the measures, like incremental statistics or windows. On the other hand, the measures are usually mathematically tested and designed to measure a specific task.

Whenever an algorithm divides the stream into windows to build the model, it usually means that the evaluation measure will also assess the performance inside such a window. Among the many window configurations, the usage of sliding windows is widespread since it allows the evaluation of the latest chunk of data [32], [38], [40], [41], [47], [54], [56], [57], [62], [71], [72], [75]. It is assumed that all other studies use incremental statistics.

Table 13 presents the references for each used measure. Next, we define each measure and provide a little background for them. When considering a binary IDS problem with only two classes, there are four possible prediction outcomes: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In such cases, negatives are usually related to the 'normal class,' while positives are associated with the 'attack' class. On the other hand, true and false tell whether the prediction was correct or incorrect according to the actual label of each instance.

One of the most widely used measures to evaluate the correctness of the decision models is the accuracy measure, which states the number of correct classifications among all the predictions.

The accuracy metric is defined by Eq. 1:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

Next, we define Recall, also known as sensitivity, hit rate, or True Positive Rate (TPR). Recall indicates the detection ratio among all positive instances. The Recall is defined according to Eq. 2:

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Precision (or Positive Predictive Value) is a measure that reflects the correctness of all the positive predictions and is defined by Eq. 3:

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

The False Alarm Rate (FAR) or False Positive Rate (FPR) calculates the model's rate for wrong positive predictions and is defined in Eq. 4:

$$FAR = \frac{FP}{FP + TN} \tag{4}$$

The ROC curve (Receiver Operating Characteristic curve) is a chart showing the performance of a classification model by plotting TPR vs. FPR. The chart shows the relationship between the two metrics, which are usually inversely proportional. In addition, the ROC curve can be evaluated in a two-dimensional space, also known as Area Under the ROC Curve (AUC). The AUC metric provides an aggregate performance measure across all possible steps of the ROC curve. AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0, while one whose predictions are 100% correct has an AUC of 1.0.

The Fmeasure, F-value, or F-score is a measure of accuracy given by the harmonic mean of precision and Recall. Fmeasure is defined in Eq. 5:

$$Fmeasure = \frac{TP}{TP + \frac{(FP+FN)}{2}} \tag{5}$$

in binary cases, but it is also employed in multiclass scenarios. In multiclass scenarios the $Fmeasure_{macro}$ is calculated according to Eq. 6:

$$Fmeasure_{macro} = \frac{Fmeasure_{class\_1} + \ldots + Fmeasure_{class\_n}}{n} \tag{6}$$

The Kappa (Cohen's Kappa) statistic measures the level of agreement for categorical items. This means that Kappa can handle multiclass and imbalanced class problems where other metrics like accuracy, precision, and Recall are insufficient. Kappa is defined by Eq. 7:

$$k = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}, \tag{7}$$

where p_0 is the observed agreement, and p_e is the expected agreement.

Some predictive performance measures were used by only a few works like the Energy Consumption [69], the Equalized Lost Accuracy (ELA) [60], False Negative Rate (FNR) [49], FNEW [14], and MNEW [14]. For the sake of completeness, we are adding their definition in the following equations..

$$ELA = \frac{A_{0\%} + A_{x\%}}{A_{0\%}} + \frac{100 + A_{0\%}}{A_{0\%}}, \tag{8}$$

where $A_{0\%}$ is the accuracy of the classifier with a noise level 0%, and $A_{x\%}$ is the accuracy of the classifier with a noise level x%.

$$FNR = \frac{FN}{P} = \frac{FN}{FN + TP} = 1 - TPR \tag{9}$$

$$FNEW = \frac{FP * 100}{N - Nc} \tag{10}$$

where N_c is the number of examples belonging to the novelty class in the stream.

$$MNEW = \frac{FN * 100}{Nc} \tag{11}$$

Some works approached the performance evaluation differently, considering memory consumption, energy consumption, running time, and throughput. These metrics are not related to the IDS's predictive performance for malicious attacks but may be essential to assess a system's computational performance. Since the IDS is a real-time system, such metrics can indicate the feasibility of employing the solution in specific environments. For example, a high-volume network might perform better using a high throughput but a slightly less accurate model, as the predictions must be fast.

V. CHALLENGES FOR BUILDING AND DEPLOYING DATA STREAM-BASED IDS

Based on the analysis of the surveyed studies, we identified several challenges for building and deploying data stream-based IDS. This section summarizes these challenges into the following groups: data source, model building and updating, decision model, scalability, and performance. Next, we discuss these challenges and provide insights on overcoming them.

A. DATA SOURCE

In the last decade, building ML-based IDS using network flow records has gained widespread popularity to the point where most literature on the subject does not even discuss the existence of other data sources. This trend can be attributed to several factors, including the speed of flow-based intrusion detection as compared to packet-based inspection, the ability to inspect encrypted network traffic using summarized information from packet headers, the prevalence of evaluation datasets that provide only network flow files, and the possibility of achieving near-real-time response times [78]. However, using network flow data, which provides only a summarized representation of a set of network packets, may pose challenges to detecting a wide range of attacks. For instance, flow-based IDSs cannot detect attacks injected into packet payloads, such as SQL injection and cross-site scripting. The flow export interval's value can also impact the system's performance. For example, minimal time intervals may overload the system, while large intervals may result in undetected short-timed attacks. Therefore, deploying flow-based IDSs without considering the specifics of the system can be risky, especially in real-world scenarios. To overcome these limitations, combining flow-based techniques with Deep Packet Inspection or using stream mining classification techniques in packet-level data (header and/or payload) are promising alternatives. As discussed in [13], such techniques offer similar performance to flow-based methods and operate directly on packet-level data, requiring no infrastructure for processing network flows.

Considering flow-based data sources, generating network flows from raw packets is common using tools like CICFlowMeter and NFStream. In general, such tools also create a list of statistical flow features, making feature extraction another crucial aspect of ML-based IDS. A common approach to ML-based IDS proposals, including incremental ones, uses a fixed set of features selected from the generated list. However, using a fixed set of features can lead to redundant and insignificant data that does not contribute to the model's predictive power. To tackle this issue, researchers have developed techniques to remove redundant features based on their importance to the model.

According to [7], more rigorous methodologies for feature selection and feature removal would improve the reproducibility of studies and aid in conducting more reliable intrusion detection evaluation. However, the dynamic nature of cyberattacks could impose limitations on robust feature engineering methods for IDS. For example, what happens when a new attack type arises and an attribute or a set of attributes important to detect it is removed in the feature selection process? Such a situation demands non-static solutions within the scope of data stream-based IDS. However, there is a lack of studies in this area. Some examples of adaptive preprocessing techniques proposed for IDS include the dynamic discretization feature method at training time [87] and online feature selection [88], [89].

B. MODEL BUILDING AND UPDATING

The choice between supervised and unsupervised algorithms for network traffic classification is a complex decision, particularly in the context of data stream-based IDS where labeled examples are not always readily available. When only examples of normal traffic are at hand, it is common to resort to unsupervised approaches, simplifying the problem to one-class classification. This allows new examples to be classified as normal and used to update the model in an unsupervised manner. Any examples that the model fails to classify are labeled as potential attacks.

TABLE 13. Data stream-based IDS studies organized by evaluation measures.

Metric	References
Accuracy	[40], [50], [58], [61], [68]–[72], [74] [37], [41], [44], [45], [48], [52], [55], [56], [60], [63], [66] [57], [75]–[77]
Precision	[11], [15], [38], [41], [47], [51], [55], [60], [62], [74]–[76], [78]
Recall/TPR	[15], [32], [46], [58], [60]–[62], [69], [70], [74] [11], [31], [35], [41], [45], [49], [51], [55], [67] [38], [64], [75], [76], [78]
FAR/FPR	[46], [49], [58], [61], [67], [69], [74] [31]–[33], [35], [41], [43], [45], [50] [11], [48], [66], [75], [78]
AUC	[9], [41], [42], [42], [51], [61], [71], [73], [77]
ROC	[36], [41], [46], [49], [61], [69], [73]
F-measure	[9], [10], [38], [47], [55], [61], [62], [68], [75]–[77]
Kappa	[50], [55], [72]
Running time	[37], [49], [50], [68], [70], [71]
Memory Consumption	[50]

When a set of normal and attack examples are available, an initial classification model can be induced using supervised or semi-supervised approaches. However, the decision model must be constantly updated to incorporate the data distribution changes due to the concept drift and concept evolution phenomena (described in Section III). In this case, supervised approaches require a domain specialist to label new examples for the model updating. Combining supervised and unsupervised learning techniques provides an interesting approach. The classification model uses unlabeled samples to update itself without external feedback, but it can also integrate new labeled examples when they become available. Alternatively, researchers have yet to extensively explore important topics like data summaries that can use labeled and unlabeled data to update their models.

Considering that supervised approaches can result in more accurate models and address the binary and multiclass classification of network traffic, its main disadvantage is that obtaining labeled examples is costly. The presence of a specialist to manually label all data stream examples is impracticable and inefficient, as studies show that a small set of carefully chosen examples is enough to improve the model's performance at a significantly lower cost. Although recent works show how the choice of a small representative set of examples to be labeled impacts the performance of the classification model, some issues still need to be solved. Randomly choosing examples to be labeled is a naive approach that tends to favor the majority class (normal traffic) instead of choosing decision boundary samples (which usually provide the most significant benefit). Here, the choice of samples based on the classification probability [90], distance to the already labeled samples, and detection of an element in density regions [91] are possible approaches.

Even if the number of samples to be labeled by a specialist is small, the process is still manual and laborious to scale up, as it is subject to the work hours of the specialist. Such constraints can increase the delay between the arrival of a sample and its respective label, impacting the model's performance. The faster the samples are labeled, the faster the model is updated and can react to changes. A fast update is crucial when attack behaviors are constantly changing or new attacks (such as zero-day attacks) emerge. For such scenarios, the novelty detection approaches can be used to identify changes in the data distribution or appearance of new concepts and, thus, trigger alarms for domain specialists.

Several studies have discussed incremental model updating for network traffic classification. On the other hand, few studies have addressed the issue of how to forget old concepts learned by these models. The trade-off between having an updated model with the most recent data and having a model with sufficient information about the past to avoid retraining is a crucial question to be addressed. In practice, decay factors are commonly used mechanisms for forgetting old concepts. Still, selecting the appropriate decay factor is directly related to the rate of changes in a specific network and cannot be easily identified in advance.

C. DECISION MODEL

In most IDS studies, binary classification algorithms have been used to distinguish between normal traffic and an attack. Alternatively, one-class classification strategies have been proposed in which only the normal concept is learned. However, neither of these approaches can identify the specific type of attack that has occurred.

Knowing the specific type of attack can aid domain specialists in taking action to contain the damage caused by the attack. As such, several studies have employed multiclass classification algorithms to classify the network traffic in the normal class or one of the known attack types. This method poses another challenge in the form of the emergence of new

attack types. In such cases, the model needs constant updates to keep the correct representation of the data.

Model interpretability is also crucial when developing data stream-based IDS. Interpretable models are valuable in scenarios where an IDS alert prompts a domain specialist to understand how the attack was detected and which key variables were involved before making a decision. In these scenarios, closed-box models are inadequate because they only identify an attack, but it is impossible to understand how the model can reach such a result. One of the most used interpretable models for intrusion detection is the decision trees and their variants (for example, random forest and Hoeffding Tree). However, as the intrusion detection datasets contain continuous attributes, choosing between binary and multi-way splits of these attributes is an issue few have explored in the literature. In addition, the interpretability of a tree is linked to its size. Deeper trees with more leaf nodes generate more complicated rules that are difficult to understand.

Recent works have used eXplainable AI (XAI) for ML-based IDS [92] to improve the control and trust in detecting new types of attacks. Due to the importance of XAI in the current scenario, this could be a challenging research area that needs further investigation.

D. SCALABILITY AND PERFORMANCE

According to [6], there are two types of metrics for IDS evaluation: security-related and performance-related metrics. The former focuses on quantifying the accuracy of attack detection, while the latter quantifies the non-functional properties of an IDS, such as capacity and resource consumption. Regarding ML-based IDS, it is common to use the confusion matrix as the source of security-related metrics, such as true-positive rate, false-positive rate, etc. The majority of ML-based IDS literature focuses on security-related evaluation. Aside from straightforward performance metrics collected in batch scenarios, such as model training/testing time and memory/CPU consumption during training/testing, this type of IDS evaluation is rarely discussed in papers. Several other questions should be addressed if we consider using data stream-based IDS.

When developing data stream-based IDS suitable for real-world environments, the time spent classifying a sample is crucial. An intrusion detection system that perfectly classifies all samples but takes unreasonable time will not be helpful for most organizations, as a single attack can be distributed across multiple network packets and flows. Therefore, the classification time of each sample must lie within acceptable constraints that allow for the swift deployment of security policies. In this sense, evaluating the relationship between the complexity of the classification algorithm and the time taken to classify a sample, especially in data stream-based IDS, is paramount for the system's performance.

Additionally, in data stream-based IDS, the time required to update the classification model is also a critical metric. Although incremental models offer a more comprehensive approach to building robust IDSs, it is vital to assess whether the time required to update the model in such cases is reasonable. For example, suppose that a new unlabeled sample arrives. After obtaining the label of such a sample, the IDS would need some time to update the decision model. How many samples of this new label were misclassified until the end of the update process? This information might give us an idea about the ''reaction time'' of IDS when dealing with unseen attacks.

Finally, despite the extensive study of security-related metrics conducted in [6] and the wide adoption of confusion matrix metrics, there is one topic that should be considered when data stream-based IDS are being used in real-world environments: what range of values are acceptable for IDS evaluation metrics (precision, recall, and false positive rates, for example)? The values found in IDS literature, where training and testing samples usually belong to the same dataset, are usually very high. However, it is unclear whether the same result holds when the intrusion detection model faces challenging scenarios, such as training an IDS using a public/labeled dataset and testing it using real-world network traffic samples.

VI. PRACTICAL GUIDELINES: COMPUTER TOOLS, PUBLIC INTRUSION DATASETS, AND RELEVANT DATA STREAM-BASED IDS

Section V summarized a series of challenges for building and deploying data stream-based IDS. In this section, we want to provide some practical guidelines to support the development of the upcoming generation of ML-based IDS. Due to the diversity of available solutions, we focus on two topics: computer tools and public intrusion datasets. First, we discuss the computer tools for building and evaluating stream models (Subsection VI-A). Next, we examine computer tools for handling network packets and flow (Subsection VI-B) and present some guidelines for selecting the most appropriate public intrusion dataset (Subsection VI-C). Finally, we discuss some relevant data stream-based IDS studies (Subsection VI-D).

A. TOOLS FOR BUILDING AND EVALUATING MODELS

In this survey, we presented several studies that proposed novel algorithms to address data stream-based IDS. Most surveyed works employed publicly available algorithms in data stream mining tools. Next, we present some of these tools.

- MOA [85]: is an open-source Java framework for data stream mining, which includes a set of algorithms for different machine learning tasks such as classification, clustering, regression, and outlier detection. Many of the tree-based classification algorithms used in data stream-based IDS are available at MOA, especially different strategies of the ensemble of classifiers. This framework also includes a set of evaluation methods for

data stream classification as interleaved-test-than-train, windowed and prequential, and different evaluation measures. Another important functionality available in MOA is the possibility of performing experiments using active learning strategies, which allows for evaluating the intrusion detection task from a more realistic point of view.

- SAMOA [93]: the intrusion detection task faces important challenges, such as the high volume and velocity of the data. To overcome these challenges, experiments must be performed using distributed streaming algorithms. SAMOA provides a collection of distributed streaming algorithms for different machine learning tasks. In addition, it allows the development of stream mining algorithms and their execution in multiple Streaming Processing Engines (SPE) such as Apache Storm, Apache S4, and Apache Samza.
- River [94]: is a Python library to build online machine learning algorithms, which are essential to data streams. Here, several algorithms, evaluation methods, and evaluation measures are available. An important feature of this tool is its ability to deal with feature evolution, which means considering that features of the dataset can appear or disappear over time. Although the feature evaluation strategies are challenging, they have yet to be explored in IDS works.

Another tool to perform experiments using fast streaming decision trees is STREAMDM C++ [95], a library in C++ containing tree-based algorithms and ensembles and evaluation methods for data streams. ADAMS [96] (Advanced Data Mining And Machine Learning System) is a flexible workflow engine to quickly build and maintain data-driven, reactive workflows, easily integrated into business processes. This library uses operators/actors in a tree-like structure to define the workflow. It can integrate several other software such as WEKA, MOA, MatLab, Excel, and SQL databases.

B. TOOLS FOR HANDLING NETWORK PACKETS AND FLOWS

As discussed in Section IV, the feature extraction for ML-based IDS could be done on two levels: network traffic data and event logs. In both cases, several computer tools can be used to facilitate the data-gathering process. Since the event logs are usually related to host-based IDS and depend on the target system, we will discuss some tools for handling network traffic data. First, we present tools focused on handling raw packet data and then show tools associated with flow-level data.

- Tcpdump [97]: is command-line packet analyzer. Van Jacobson originally wrote it as part of an ongoing research project to analyze and improve the performance of Internet protocols such as TCP. Tcpdump runs in a single computer and prints the contents of network packets. It can read packets from a network interface card or a previously created saved packet file. All public network intrusion datasets surveyed here use Tcpdump (or some variant, such as Wireshark) to capture network traffic data.
- Gulp [98]: is a remote packet capture tool based on Tcpdump. Corey Satten wrote the tool, focusing on dropping fewer packets than similar tools and having a higher capture rate. Gulp reads packets from the network card and flushes its buffer if nothing has been written in the last second. After receiving an interrupt, Gulp stops filling the buffer but writes the remaining buffered data into the disk.
- Zeek [99]: is a network traffic analyzer initially designed by Vern Paxson. The main goal of Zeek is to inspect network traffic and generate different types of logs associated with the detected activity. The tool consists of an event engine that is triggered by various events in the network traffic. One common functionality is converting raw network traffic into logs summarizing TCP and UDP data.

Some relevant tools to manage network flow level data include:

- CicFlowMeter [100]: is a network traffic flow generator and analyzer developed by the Canadian Institute for Cybersecurity at the University of New Brunswick. The tool can generate bidirectional flows by reading a packet capture file (PCAP) or sniffing a network interface in real-time. More than 80 statistical network traffic features, such as Duration, Number of packets, Number of bytes, Length of packets, etc., can be calculated separately in the forward and backward directions. Currently, two versions are available: Java[7] and Python.[8] Several public network intrusion datasets use CICFlowMeter. It is also common to see ML-based IDS studies that use CICFlowMeter to generate network flows from PCAPs.
- Argus [101]: is an open-source tool that processes packet data and generates summary network flow data. The Argus tool consists of an Argus server and Argus clients. While the Argus server writes/processes PCAP files of receiving packets, the Argus client extracts the features from the Argus files. The tool runs on Mac OS X, Linux, Unix, or Cygwin-enabled Windows systems. The UNSW-NB15 dataset, for example, used Argus to create its features.
- Nfstream [102]: is a recent open-source multi-platform Python framework providing flexible and expressive data structures designed to work with online (live-capture) or offline network traffic data (PCAP files). Flow features are derived from the IP, TCP, and UDP packet headers. It includes post-mortem statistical features (e.g., minimum, mean, standard deviation, and maximum packet sizes and inter-arrival time) and early flow features (e.g., sequence of first n packets

[7]https://github.com/ahlashkari/CICFlowMeter
[8]https://pypi.org/project/cicflowmeter/

sizes, inter-arrival times, and directions). Besides, new features can be written using NFPlugins. Due to the lack of standardization in previous related tools, their goal is to provide a unified network data analytics framework, increasing data reproducibility across experiments.

C. GUIDELINES FOR USING PUBLIC INTRUSION DETECTION DATASETS

Here, we provide some practical guidelines for using public intrusion detection datasets. We evaluated five attributes in each of the datasets described in Section IV-A2. It is important to see that Table 6 contains only network-based intrusion datasets.

1) Label - refers to the availability of labels for each instance in the dataset. We also identify whether the label is related to a binary problem (normal traffic/attack) or a multi-class problem (attack type 1, attack type 2, and so on);
2) Data source - there exists three types of data: packet level data (PCAP or other formats), flow level data (usually a CSV file), or other (a set of event logs, for example);
3) Environment - real, emulated, or synthetic [79]. Real stands for real network traffic captured within a real-world network environment. Emulated indicates that real network traffic was captured within a testbed or emulated network environment. Synthetic denotes that the network traffic was created synthetically and not captured by a real (or virtual) network device.
4) Network traffic type - usually, we have two traffic types: normal and malicious.
5) Variety of attack types - [82] studied the different attack types presented in public intrusion detection datasets using the following taxonomy: reconnaissance (REC), authentication (AUT), Denial of Service (DoS), spam (SPM), worms (WRM), cross-site scripting (XSS), SQL injection (SQL), backdoor (BCK), rootkits (RTK), shell (SHEL), secure shell (SSH), masquerade (MSQ), data manipulation (DAT), honeypots (HON), domain (DNS), buffer overflow (BUF) and other features (OTH). Our idea is to discuss the number of attack types in each dataset.

Table 14 summarizes the relationship between each dataset in our survey and the five attributes above. It is possible to see that if a researcher wants to work with flow-level data, CTU-13, MAWIFlow, ISCX-IDS, and CIC-IDS2017 are well-suited. One can also use network flow tools such as Nfstream to create network flows by using the PCAP files available in such datasets. We can also see that all investigated datasets, except Kyoto2006+ and MAWIFlow, can be used to evaluate intrusion detection models tailored for specific attacks since they provide labels for different attack types.

Mixing data from different datasets can be a very interesting strategy. For example, will a model trained with real and emulated normal network traffic perform better when tested in a real-world environment? The same logic applies to mixing different attack types. The studied datasets can be used for this purpose. Finally, despite the wide variety of attack types and widespread usage, KDD99 and NSL-KDD are outdated datasets and should be considered obsolete [103]. For example, the underlying network traffic for both sets dates back to 1998. Therefore, using modern intrusion detection datasets such as UNB CIC IDS,[9] UNSW-15,[10] and Stratosphere Laboratory Datasets[11] should be a pivotal principle to evaluate data stream-based IDS. References [79] and [82] provide further details about these and other public intrusion datasets.

D. RELEVANT DATA-STREAM BASED IDS

In this section, we discuss some examples of relevant data stream-based IDS proposals found during the survey. We selected six studies: [9], [15], [49], [65], [76] and [77]. The main goal here is to illustrate how these studies tackled the different issues associated with the design and implementation of data stream-based IDS.

In [9], authors propose INSOMNIA (Incremental training iNtrusion System Over tiMe-stamped Network traffIc dAta), a semi-supervised data stream-based IDS that combines incremental, active, and transfer learning. According to the authors, the proposal aims to overcome a set of open challenges that limit the practicality of current ML-based IDS: the non-uniform distribution of network traffic over time, the high cost of labeling, latency during model updates, and the need for explainability. The system initially learns an intrusion detection model from a limited amount of labeled instances. The new unlabeled instances are classified as benign or attack using the knowledge from the initial model and are aggregated into batches of a specific size. An oracle mechanism also labels the new instances using a different algorithm. Once a batch is complete, the model update process begins. The idea here is to update the initial model using an active learning strategy based on an uncertainty sampling query. In other words, the least certain predictions from the initial model are labeled using the oracle mechanism. Using this strategy, the intrusion detection model would be updated using the ''pseudo-label'' obtained from a set of new instances. The authors evaluated the method using a revised version of CIC-IDS2017 [104]. They trained the initial model with the first two days of the dataset. The performance, in terms of F1 score, was around 80% when using a 50% selection rate for uncertainty sampling. This means that only the top 50% most uncertain predictions were used to update the model.

INSOMNIA is an excellent example of data stream-based IDS. The authors developed an end-to-end system that handles the continuous data generation and model update

[9] https://www.unb.ca/cic/datasets/index.html
[10] https://research.unsw.edu.au/projects/unsw-nb15-dataset
[11] https://www.stratosphereips.org/datasets-overview

TABLE 14. Summary of public intrusion detection datasets found in the surveyed studies. *b*=binary; *m*=multiclass; *p*=packet; *f*=flow; *o*=other; *n*=normal; *a*=attack.

	Label	Data source	Environment	Network traffic	Attack types
KDD99	*b* and *m*	*o*	emulated	*n* and *a*	8
NSL-KDD	*b* and *m*	*o*	emulated	*n* and *a*	8
CTU-13	*b* and *m*	*p* and *f*	real	*n* and *a*	4
MAWIFlow	*b*	*p* and *f*	real	*n* and *a*	7
Kyoto2006+	*b*	*o*	real	*n* and *a*	1
ECML-PKDD	*b* and *m*	*o*	real	*n* and *a*	3
CSIC	*b* and *m*	*o*	emulated	*n* and *a*	4
ISCX-IDS	*b* and *m*	*p* and *f*	emulated	*n* and *a*	8
CIC-IDS2017	*b* and *m*	*p* and *f*	emulated	*n* and *a*	9

and simulates a more realistic scenario for evaluating ML-based IDS. Interestingly, the obtained performance differs from other batch-learning proposals that can reach F1 score values higher than 95% [105]. This shows the difficulty of deploying ML-based IDS when we do not assume that traffic data is stationary. Researchers can use INSOMNIA's ideas to develop more robust ML-based systems. Two topics can be further investigated here: i) improve the model update process to decrease the update latency time and ii) use cross-evaluation dataset strategies (train the initial model on dataset A and test the model on a different dataset B).

While in INSOMNIA [9], the authors tackle the drift during the model update process by using the least certain predictions and the idea of "pseudo-label" from an oracle, [76] proposes the use of specific drift detectors (Drift Detection Method, in their case) combined with online sequential extreme learning machine (OSELM) to build stream based-IDS. The Drift Detection Method (DDM) assumes the data comes from a stationary process and detects changes in data distribution by monitoring the error rate over time. It calculates the classifier's average error rate and variance, updating them as new data points arrive. When the error rate exceeds a threshold determined by the Hoeffding bound, the algorithm signals a drift [106].

The streaming data passes through a data balancing block, which updates non-balanced data with previously stored data from minority classes. The intrusion detection model is updated with each new data batch. Then, the DDM is responsible for deciding whether to update the model. The results are promising for various drift types and datasets (KDD Cup '99, CICIDS-2017, CSE-CIC-IDS-2018, and ISCX2012). For the CICIDS-2017, for example, the authors reach a F1 score of 89%. Despite using an interesting technique to deal with concept drift and, consequently, the unseen attacks, the proposed data stream-based IDS uses a total feedback approach for the model update, which means that the ground-truth labels are readily available during the incremental learning. It would be important to see other studies combining different concept drift technique variants with partial or no model feedback.

BigFlow [65] is another example of data-stream-based IDS proposed in the literature. It uses the concept of classification with rejection, in which the classifier can reject an instance based on the confidence of its classification. If the classifier has low confidence in classifying an instance, it rejects it, making the administrator aware that a possible change can occur. After a period, the reject instances are labeled by a specialist or a tool and used to update the model. If a reject instance is not labeled as an attack after a long period, it is deemed a normal event. Although classifying with the reject option is an interesting approach to dealing with concept drift, if the model is not updated for a long period, its confidence in the classification of new instances can be outdated and not reflect the current state of the stream. Thus, mechanisms of unsupervised continuous updates of the decision model are also important to intrusion detection.

Finally, the studies conducted in [15], [49], and [77] are particularly relevant as they compare various data-stream classification algorithms using public IDS datasets. Table 15 summarizes each study, detailing the computational tools, selected algorithms, datasets, and key performance results. It can be observed that all of them employ tree-based ensemble models and utilize the MOA framework. In terms of results, the initial benchmarking studies indicate strong performance (precision and recall), even when dealing with limited labeled instances and delayed data availability.

VII. DISCUSSION, LIMITATIONS, AND FUTURE RESEARCH DIRECTIONS

The findings of this survey indicate that research in ML-based IDS is shifting from merely applying batch-based models to public datasets toward more in-depth discussions on real-world implementation challenges. These include issues such as the lack of labeled data, model updating, concept drift, and data quality. This paradigm shift acknowledges that intrusion detection must account for the continuous nature of network traffic and the evolving behavior of attackers. Using data stream techniques in IDS implementation and evaluation is crucial, enabling models to be evaluated under conditions resembling real-world scenarios.

TABLE 15. Summary of some benchmarking studies found in the surveyed work.

Reference	Computer tools	Algorithms	Dataset	Performance
[50]	MOA	AccuracyUpdatedEnsemble, ActiveClassifier, LeveragingBag, LimAttClassifier, OzaBagAdwin, OzaBagASHT, and SingleClassifierDrift	KDD Cup 1999	Accuracy >= 90% after $1.5 * 10^5$ instances
[15]	MOA	VFDT, Hoeffding Adaptive Tree, Ozabag, Ozaboost, and OzaBagAdwin	CTU-13	Precision >= 90% in some scenarios using less than 5% of labeled instances
[78]	MOA	LeveragingBag, Limited Attribute Classifier, OzaBagAdwin, OzaBagASHT, and Adaptive Random Forest	CICIDS2017 and CICIDS2018	Precision and Recall around 80% when using less than 5% of labeled instances and a delay of 10,000 instances

Although many works focus on intrusion detection in computer networks using classical or data stream-based machine learning techniques, several issues remain unresolved. Key areas that need attention include the development of online pre-processing methods, the management of unbalanced data sets in real-time, and the ability to update models with only a few labeled examples. In particular, these studies must also consider the requirements for developing intrusion detection systems (IDS) that are suitable for real-world applications. This means that such systems cannot assume immediate labeling and must account for delays in the arrival of labels, often resorting to self-supervision techniques.

A. LIMITATIONS

Among the limitations of the proposed work, it is important to note that it focuses only on reviewing intrusion detection systems within the context of computer networks, as evidenced by the search string described in Section IV. However, while the principles of intrusion detection can be applied in other contexts, the requirements differ significantly. For instance, the task of securing vehicles against unauthorized access will have to cope with a stronger constraint on computational power and memory capacity. Examples of research in this area include [107] and [108]. Accordingly, a promising research direction is exploring how data-stream IDS are being adopted in specific application domains such as vehicle networks and controller area networks.

Another limitation of the proposed work is that important aspects of machine learning for data streams, including how to deal with online pre-processing and unbalanced classification, are only superficially discussed due to lack of representation in the reviewed studies. Although recent studies have addressed these issues in data stream scenarios, they are yet to be explored in the context of network IDS. Therefore, we highlight them as future directions for the field.

B. FUTURE RESEARCH DIRECTIONS

While several studies have been proposed to address data stream-based IDS, important challenges remain unaddressed. Here, we outline key challenges that future research should tackle.

1) PREPROCESSING

despite being a critical stage in ML-based IDS, the research community needs to pay more attention to it [7], and most studies need more details regarding their procedures. Besides hindering reproducibility, this behavior prevents understanding the proper impact of different preprocessing techniques on intrusion detection model performance. To address these issues, researchers should propose two types of experimental studies: i) identifying the relationship between different preprocessing techniques and the predictive power of the intrusion detection model and ii) evaluating the use of preprocessing techniques tailored for data-stream-based IDS solutions. The authors in [109] discussed this issue and conducted some preliminary tests in non-IDS-related scenarios. They discovered that when selecting preprocessing methods for data stream mining, one must consider the model performance and the computational costs associated with such methods.

2) FEATURE EXTRACTION

network-based IDS can be built using packet header, payload, or network flow features. However, despite their smaller size than raw packets, the ML-based IDS research community often uses network flow features without proper motivation or justification. Thus, it is important to understand the advantages and drawbacks of each data source and its ability to detect different attack classes. Furthermore, commonly used methodologies that perform feature extraction from the network flow, feature selection, and model evaluation can be rendered ineffective since features can have varying importance in online scenarios where network traffic behavior can change over time. Therefore, researchers should consider using feature evolution methods [110] when designing ML-based IDS for real-world scenarios.

3) MODEL EVALUATION MEASURES

the classical strategy of evaluating a decision model by computing evaluation measures after processing all data instances must be adapted to IDS. Identifying the moment when new attacks appear and the time taken to adapt the decision model and incorporate these new concepts is crucial. In addition, in the presence of concept drift, where the known

concepts evolve, it is important to evaluate the adaptation of the decision model. Considering supervised models, some issues to be considered in evaluating data stream-based IDS include the number of labeled instances used to update the model and the delay in obtaining the true label of these instances.

4) CONCEPT DRIFT

the rationale behind using adaptive models that can learn from new data is based on the assumption that intrusion detection involves non-stationary data distribution due to inherent changes in network traffic over time. Such changes can be categorized as abrupt, gradual, incremental, or recurrent. While specific detectors may perform well in the face of sudden or gradual changes, it is essential to explore further the various types of changes that occur in intrusion detection and determine the appropriate methods to handle them.

5) ACTIVE LEARNING

active learning strategies have been employed in the machine learning field to reduce the number of labeled instances required to update the decision model. However, their application in IDS is yet to be extensively explored. Several issues remain open, including the delay associated with obtaining true labels, the presence of uncertainty in the labeling process, the specialist's availability to label new instances, the knowledge of the specialist, and the time taken to label different types of attacks. Although some recent works have attempted to address these questions, they generally assume that specialists are always available to label instances and can label all selected instances constantly. They also assume the delay in obtaining the true label of all instances is fixed. Additionally, these works consider that the predictions of the specialist are always correct. Most of these assumptions may not be realistic for real-world scenarios.

6) IDS DEPLOYMENT

the ML-based IDS literature is not focused on solving practical issues associated with the system operation. Some relevant questions include: i) in which hardware architecture should the system be installed? ii) will all the required data sources be available when deploying the IDS? For example, IDS based on network flows requires a flow processing tool for incoming/outgoing network packets. Which tool will be used for that? Can this tool be deployed along with the IDS? How will the incoming/outgoing packets be processed? iii) How will the IDS be managed? Is there a GUI for the final user? How will the alerts be shown to the final user? iv) How will the final user include labeled samples in the system in online scenarios?

7) EMPIRICAL EVALUATION

given the importance of empirical evaluations and comparative performance analyses in IDS research, a key direction for future work is to conduct empirical studies comparing the different data stream-based IDS reviewed in this work and assessing how effectively they address the main challenges of data stream scenarios. This was initially explored in studies such as [15], [49], and [77]. However, as shown in Table 15, these studies lack key elements required for a comprehensive benchmarking analysis, such as a broader selection of algorithms, datasets, and evaluation metrics. We encourage researchers to further explore data-stream algorithms for ML-based IDS, following the approach of benchmarking batch-based IDS using the CICIDS2017 dataset [105]. Furthermore, a detailed examination of implementation tools and their practical usage is a crucial task that must be undertaken to enhance the practicality and effectiveness of data-stream-based IDS.

8) ADDITIONAL ML TECHNIQUES

other ML techniques might be worth exploring within the IDS scenario. For example, Novelty Detection [111] tries to detect new emerging concepts from the stream as they are happening. Such techniques can be combined with Active Learning and specialist feedback to compose a robust IDS framework, as they can reduce the amount of data that requires specialist attention. Another example is the Continual Learning [112] technique, which tries to learn a model for many sequential tasks without forgetting the old concepts. Usually, only data from the new tasks are available when learning them, and the new tasks represent classes from a problem that shares the same feature space. These techniques have mechanisms resembling the apparition of new attacks in the IDS scenario and might provide suitable future work venues. It's worth noting that most of the surveyed studies have used tree-based algorithms or an ensemble of trees. VFDT (Very Fast Decision Tree) and its variants are the most commonly used algorithms. Although these have shown promising results for data-stream based IDS, it is crucial to thoroughly investigate other algorithms, such as SVM, KNN, and neural networks.

VIII. CONCLUSION

Over the past two decades, the intrusion detection systems (IDS) literature has advanced significantly with the integration of machine learning techniques. Nevertheless, many studies fail to address key characteristics of real-world environments, such as the continuous arrival of new data instances, the emergence of previously unseen attack types, and the scarcity of labeled samples. Data stream-based IDS approaches tackle these challenges through incremental learning, active learning, novelty detection, and concept drift detection. In this work, we present a comprehensive survey on data stream-based IDS. In addition to proposing a detailed taxonomy, we offer an updated formalization of the IDS problem, review relevant datasets and software tools, and provide practical recommendations and guidelines for developing data stream-based IDS solutions.

Our findings indicate that robust data stream-based IDS can effectively overcome several limitations of conventional

ML-based IDS, such as handling concept drift (allowing for the detection of zero-day attacks), reducing dependence on large amounts of labeled training data, and mitigating delays in obtaining ground-truth labels. Future research directions include conducting empirical evaluations or benchmarking data-stream algorithms in ML-based IDS, analyzing the impact of network traffic variations on their performance, exploring active learning strategies, and tackling practical challenges in deploying data stream-based IDS.

REFERENCES

[1] (Nov. 2022). *Threat Landscape*. Accessed: Apr. 10, 2023. [Online]. Available: https://www.enisa.europa.eu/topics/threat-risk-management/threats-and-trends

[2] D. Ehrlicher. (2021). *Council Post: The Evolution Of Cybersecurity in 2021—Forbes.com*. Accessed: Mar. 22, 2023. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2021/03/05/the-evolution-of-cybersecurity-in-2021/

[3] M. J. Firch. (2021). *10 Cyber Security Trends You Can't Ignore in 2021—PurpleSec—purplesec.us*. Accessed: Mar. 22, 2023. [Online]. Available: https://purplesec.us/cyber-security-trends-2021/

[4] A. Kivva. (2024). *The Mobile Malware Threat Landscape in 2023*. Accessed: Jun. 30, 2024. [Online]. Available: https://securelist.com/mobile-malware-report-2023/111964/

[5] S. S. Shafin, G. Karmakar, and I. Mareels, "Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications," *Sensors*, vol. 23, no. 11, p. 5348, Jun. 2023.

[6] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating computer intrusion detection systems: A survey of common practices," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 1–41, Sep. 2015.

[7] D. Chou and M. Jiang, "A survey on data-driven network intrusion detection," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–36, Dec. 2022.

[8] 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. Netw. Service Manage.*, vol. 17, no. 4, pp. 2451–2479, Dec. 2020, doi: 10.1109/TNSM.2020.3016246.

[9] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "INSOMNIA: Towards concept-drift robustness in network intrusion detection," in *Proc. 14th ACM Workshop Artif. Intell. Secur.*, Nov. 2021, pp. 111–122, doi: 10.1145/3474369.3486864.

[10] P. Horchulhack, E. K. Viegas, and A. O. Santin, "Toward feasible machine learning model updates in network-based intrusion detection," *Comput. Netw.*, vol. 202, Jan. 2022, Art. no. 108618.

[11] V. G. T. D. Costa, B. B. Zarpelão, R. S. Miani, and S. B. Junior, "Online detection of botnets on network flows using stream mining," in *Proc. Anais do 36th Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, May 2018, pp. 225–238. [Online]. Available: https://sol.sbc.org.br/index.php/sbrc/article/view/2418

[12] C. Constantinides, S. Shiaeles, B. Ghita, and N. Kolokotronis, "A novel online incremental learning intrusion prevention system," in *Proc. 10th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Jun. 2019, pp. 1–6.

[13] G. Olimpio, P. F. C. Silva, L. Camargos, R. S. Miani, and E. R. de Faria, "Intrusion detection over network packets using data stream classification algorithms," in *Proc. IEEE 33rd Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2021, pp. 985–990.

[14] G. W. Cassales, H. Senger, E. R. de Faria, and A. Bifet, "IDSA-IoT: An intrusion detection system architecture for IoT networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2019, pp. 1–7.

[15] G. H. Ribeiro, E. R. de Faria Paiva, and R. S. Miani, "A comparison of stream mining algorithms on botnet detection," in *Proc. 15th Int. Conf. Availability, Rel. Secur.*, Aug. 2020, pp. 1–10, doi: 10.1145/3407023.3407053.

[16] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Appl. Sci.*, vol. 9, no. 20, p. 4396, Oct. 2019.

[17] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102767.

[18] A. Drewek-Ossowicka, M. Pietrołaj, and J. Rumiński, "A survey of neural networks usage for intrusion detection systems," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 1, pp. 497–514, Jan. 2021.

[19] G. Singh and N. Khare, "A survey of intrusion detection from the perspective of intrusion datasets and machine learning techniques," *Int. J. Comput. Appl.*, vol. 44, no. 7, pp. 659–669, Jul. 2022.

[20] C. Nixon, M. Sedky, and M. Hassan, "Reviews in online data stream and active learning for cyber intrusion detection—A systematic literature review," in *Proc. 6th Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Dec. 2021, pp. 1–6.

[21] O. H. Abdulganiyu, T. Ait Tchakoucht, and Y. K. Saheed, "A systematic literature review for network intrusion detection system (IDS)," *Int. J. Inf. Secur.*, vol. 22, no. 5, pp. 1125–1162, Oct. 2023.

[22] T. Sowmya and E. A. Mary Anita, "A comprehensive review of AI based intrusion detection system," *Measurement, Sensors*, vol. 28, Aug. 2023, Art. no. 100827. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2665917423001630

[23] A. Momand, S. U. Jan, and N. Ramzan, "A systematic and comprehensive survey of recent advances in intrusion detection systems using machine learning: Deep learning, datasets, and attack taxonomy," *J. Sensors*, vol. 2023, no. 1, Jan. 2023, Art. no. 6048087.

[24] A. Thakkar and R. Lohiya, "A review on challenges and future research directions for machine learning-based intrusion detection system," *Arch. Comput. Methods Eng.*, vol. 30, no. 7, pp. 4245–4269, Sep. 2023.

[25] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018.

[26] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[27] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Proc. Comput. Sci.*, vol. 60, pp. 708–713, Jan. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915023479

[28] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "A comparative study on online machine learning techniques for network traffic streams analysis," *Comput. Netw.*, vol. 207, Apr. 2022, Art. no. 108836. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622000512

[29] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of IP flow-based intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 3, pp. 343–356, 3rd Quart., 2010.

[30] Y. J. Hai, Y. Wu, and G. Wang, "An improved unsupervised clustering-based intrusion detection method," *Proc. SPIE*, vol. 5812, pp. 52–60, Mar. 2005, doi: 10.1117/12.603086.

[31] Z. Zhang and H. Shen, "Application of online-training SVMs for real-time intrusion detection with different considerations," *Comput. Commun.*, vol. 28, no. 12, pp. 1428–1442, Jul. 2005, doi: 10.1016/j.comcom.2005.01.014.

[32] A. Bellaachia and R. Bhatt, "An enhanced stream mining approach for network anomaly detection," *Proc. SPIE*, vol. 5812, p. 342, Mar. 2005.

[33] E. J. Spinosa, A. P. de Leon F. de Carvalho, and J. Gama, "Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks," in *Proc. ACM Symp. Appl. Comput.*, Mar. 2008, pp. 976–980, doi: 10.1145/1363686.1363912.

[34] Y. Wang, W. Hu, and X. Zhang, "Online boosting based intrusion detection in changing environments," in *Proc. Int. Conf. Artif. Intell.*, Las Vegas, NV, USA, H. R. Arabnia and Y. Munth Ed., Jan. 2008, pp. 735–741.

[35] W. Wang, F. Masseglia, T. Guyet, R. Quiniou, and M.-O. Cordier, "A general framework for adaptive and online detection of web attacks," in *Proc. 18th Int. Conf. World Wide Web*, New York, NY, USA, Apr. 2009, pp. 1141–1142, doi: 10.1145/1526709.1526897.

[36] H. Du, S. Teng, M. Yang, and Q. Zhu, "Intrusion detection system based on improved SVM incremental learning," in *Proc. Int. Conf. Artif. Intell. Comput. Intell.*, vol. 1, Nov. 2009, pp. 23–28.

[37] Y. Yu, S. Guo, S. Lan, and T. Ban, "Anomaly intrusion detection for evolving data stream based on semi-supervised learning," in *Advances in Neural Information Processing*, M. Köppen, N. Kasabov, and G. Coghill, Eds., Berlin, Heidelberg: Springer, 2009, pp. 571–578.

[38] Z. Li, Y. Gao, and Y. Chen, "HiFIND: A high-speed flow-level intrusion detection approach with DoS resiliency," *Comput. Netw.*, vol. 54, no. 8, pp. 1282–1299, Jun. 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128609003375

[39] H. T. Nguyen and K. Franke, "Adaptive intrusion detection system via online machine learning," in *Proc. 12th Int. Conf. Hybrid Intell. Syst. (HIS)*, Dec. 2012, pp. 271–277.

[40] H. Lampesberger, P. Winter, M. Zeilinger, and E. Hermann, "An online learning statistical model to detect malicious web requests," in *Security and Privacy in Communication Networks*, M. Rajarajan, F. Piper, H. Wang, and G. Kesidis, Eds., Berlin, Heidelberg: Springer, 2012, pp. 19–38.

[41] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proc. Volumes*, vol. 46, no. 20, pp. 12–17, 2013, doi: 10.3182/20130902-3-cn-3020.00044.

[42] A. M. Said, D. D. Dominic, and I. Faye, "Real-time network anomaly detection architecture based on frequent pattern mining technique," in *Proc. Int. Conf. Res. Innov. Inf. Syst. (ICRIIS)*, Nov. 2013, pp. 392–397.

[43] F. Breve and L. Zhao, "Semi-supervised learning with concept drift using particle dynamics applied to network intrusion detection data," in *Proc. BRICS Congr. Comput. Intell. 11th Brazilian Congr. Comput. Intell.*, Sep. 2013, pp. 335–340.

[44] K. Shafi and H. A. Abbass, "Evaluation of an adaptive genetic-based signature extraction system for network intrusion detection," *Pattern Anal. Appl.*, vol. 16, no. 4, pp. 549–566, Nov. 2013.

[45] W. Wang, T. Guyet, R. Quiniou, M.-O. Cordier, F. Masseglia, and X. Zhang, "Autonomic intrusion detection: Adaptively detecting anomalies over unlabeled audit data streams in computer networks," *Knowl.-Based Syst.*, vol. 70, pp. 103–117, Nov. 2014, doi: 10.1016/j.knosys.2014.06.018.

[46] R. Singh, H. Kumar, and R. K. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," *Expert Syst. Appl.*, vol. 42, no. 22, pp. 8609–8624, Dec. 2015, doi: 10.1016/j.eswa.2015.07.015.

[47] S.-Y. Huang, F. Yu, R.-H. Tsaih, and Y. Huang, "Network-traffic anomaly detection with incremental majority learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.

[48] J. Zhang, H. Li, Q. Gao, H. Wang, and Y. Luo, "Detecting anomalies from big network traffic data using an adaptive detection approach," *Inf. Sci.*, vol. 318, pp. 91–110, Oct. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025514007609

[49] M. A. Faisal, Z. Aung, J. R. Williams, and A. Sanchez, "Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: A feasibility study," *IEEE Syst. J.*, vol. 9, no. 1, pp. 31–44, Mar. 2015.

[50] G. Folino, F. S. Pisani, and P. Sabatino, "An incremental ensemble evolved by using genetic programming to efficiently detect drifts in cyber security datasets," in *Proc. Genetic Evol. Comput. Conf. Companion*. New York, NY, USA: Association for Computing Machinery, Jul. 2016, pp. 1103–1110, doi: 10.1145/2908961.2931682.

[51] M.-H. Chen, P.-C. Chang, and J.-L. Wu, "A population-based incremental learning approach with artificial immune system for network intrusion detection," *Eng. Appl. Artif. Intell.*, vol. 51, pp. 171–181, May 2016, doi: 10.1016/j.engappai.2016.01.020.

[52] P. Alaei and F. Noorbehbahani, "Incremental anomaly-based intrusion detection system using limited labeled data," in *Proc. 13rd Int. Conf. Web Res. (ICWR)*, Apr. 2017, pp. 178–184, doi: 10.1109/ICWR.2017.7959324.

[53] V. V. Kumari and P. R. K. Varma, "A semi-supervised intrusion detection system using active learning SVM and fuzzy c-means clustering," in *Proc. Int. Conf. I-SMAC (IoT Social, Mobile, Analytics Cloud) (I-SMAC)*, Feb. 2017, pp. 481–485.

[54] F. Noorbehbahani, A. Fanian, R. Mousavi, and H. Hasannejad, "An incremental intrusion detection system using a new semi-supervised stream classification method," *Int. J. Commun. Syst.*, vol. 30, no. 4, p. e3002, Mar. 2017.

[55] E. Viegas, A. Santin, N. Neves, A. Bessani, and V. Abreu, "A resilient stream learning intrusion detection mechanism for real-time analysis of network traffic," in *Proc. IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.

[56] E. Viegas, A. Santin, V. Abreu, and L. S. Oliveira, "Stream learning and anomaly-based intrusion detection in the adversarial settings," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 773–778.

[57] M. Zolotukhin and T. Hämäläinen, *Data Stream Clustering for Application-Layer DDoS Detection in Encrypted Traffic*. Cham, Switzerland: Springer, 2018, pp. 111–131.

[58] S. Müller, J. Lancrenon, C. Harpes, Y. Le Traon, S. Gombault, and J.-M. Bonnin, "A training-resistant anomaly detection system," *Comput. Secur.*, vol. 76, pp. 1–11, Jul. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016740481830155X

[59] X. Yuan, R. Wang, Y. Zhuang, K. Zhu, and J. Hao, "A concept drift based ensemble incremental learning approach for intrusion detection," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2018, pp. 350–357.

[60] J. L. G. Torres, C. A. Catania, and E. Veas, "Active learning approach to label network traffic datasets," *J. Inf. Secur. Appl.*, vol. 49, Dec. 2019, Art. no. 102388, doi: 10.1016/j.jisa.2019.102388.

[61] S. Park, S. Seo, C. Jeong, and J. Kim, "Online eigenvector transformation reflecting concept drift for improving network intrusion detection," *Expert Syst.*, vol. 37, no. 5, Nov. 2019, Art. no. e12477, doi: 10.1111/exsy.12477.

[62] C. Zhu, X. Wang, and L. Zhu, "Application research of a data stream clustering algorithm in network security defense," *J. Phys., Conf. Ser.*, vol. 1423, no. 1, Dec. 2019, Art. no. 012027, doi: 10.1088/1742-6596/1423/1/012027.

[63] G. Li, Y. Shen, P. Zhao, X. Lu, J. Liu, Y. Liu, and S. C. H. Hoi, "Detecting cyberattacks in industrial control systems using online learning algorithms," *Neurocomputing*, vol. 364, pp. 338–348, Oct. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219309762

[64] C. Kim and J. Park, "Designing online network intrusion detection using deep auto-encoder Q-learning," *Comput. Electr. Eng.*, vol. 79, Oct. 2019, Art. no. 106460. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790618334104

[65] E. Viegas, A. Santin, A. Bessani, and N. Neves, "BigFlow: Real-time and reliable anomaly-based intrusion detection for high-speed networks," *Future Gener. Comput. Syst.*, vol. 93, pp. 473–485, Apr. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18307635

[66] C. Yin, S. Zhang, Z. Yin, and J. Wang, "Anomaly detection model based on data stream clustering," *Cluster Comput.*, vol. 22, no. S1, pp. 1729–1738, Jan. 2019.

[67] C. Nixon, M. Sedky, and M. Hassan, "Autoencoders: A low cost anomaly detection method for computer network data streams," in *Proc. 4th Int. Conf. Cloud Big Data Comput.*, New York, NY, USA, Aug. 2020, pp. 58–62, doi: 10.1145/3416921.3416937.

[68] S. D. Çakmakçı, T. Kemmerich, T. Ahmed, and N. Baykal, "Online DDoS attack detection using Mahalanobis distance and kernel-based learning algorithm," *J. Netw. Comput. Appl.*, vol. 168, Oct. 2020, Art. no. 102756. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804520302307

[69] Md. S. Alam, C. Yakopcic, G. Subramanyam, and T. M. Taha, "Memristor based neuromorphic network security system capable of online incremental learning and anomaly detection," in *Proc. 11th Int. Green Sustain. Comput. Workshops (IGSC)*, Oct. 2020, pp. 1–8.

[70] N. Martindale, M. Ismail, and D. A. Talbert, "Ensemble-based online machine learning algorithms for network intrusion detection systems using streaming data," *Information*, vol. 11, no. 6, p. 315, Jun. 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/6/315

[71] Z. Shao, S. Yuan, and Y. Wang, "Adaptive online learning for IoT botnet detection," *Inf. Sci.*, vol. 574, pp. 84–95, Oct. 2021, doi: 10.1016/j.ins.2021.05.076.

[72] C. A. Bollmann, M. Tummala, and J. C. McEachen, "Resilient real-time network anomaly detection using novel non-parametric statistical tests," *Comput. Secur.*, vol. 102, Mar. 2021, Art. no. 102146. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820304193

[73] A. d. R. L. Ribeiro, R. Y. C. Santos, and A. C. A. Nascimento, "Anomaly detection technique for intrusion detection in SDN environment using continuous data stream machine learning algorithms," in *Proc. IEEE Int. Syst. Conf. (SysCon)*, Apr. 2021, pp. 1–7, doi: 10.1109/SysCon48628.2021.9447092.

[74] E. Mahdavi, A. Fanian, A. Mirzaei, and Z. Taghiyarrenani, "ITL-IDS: Incremental transfer learning for intrusion detection systems," *Knowl.-Based Syst.*, vol. 253, Oct. 2022, Art. no. 109542.

[75] B. Nugraha, K. Yadav, P. Patil, and T. Bauschert, "Improving the detection of unknown DDoS attacks through continual learning," in *Proc. IEEE Int. Conf. Cyber Secur. Resilience (CSR)*, Aug. 2023, pp. 477–483.

[76] M. A. Shyaa, Z. Zainol, R. Abdullah, M. Anbar, L. Alzubaidi, and J. Santamaría, "Enhanced intrusion detection with data stream classification and concept drift guided by the incremental learning genetic programming combiner," *Sensors*, vol. 23, no. 7, p. 3736, Apr. 2023.

[77] G. Olímpio, L. Camargos, R. S. Miani, and E. R. Faria, "Model update for intrusion detection: Analyzing the performance of delayed labeling and active learning strategies," *Comput. Secur.*, vol. 134, Aug. 2023, Art. no. 103451. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404823003619

[78] M. F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Comput. Secur.*, vol. 70, pp. 238–254, Sep. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404817301165

[79] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Secur.*, vol. 86, pp. 147–167, Sep. 2019.

[80] H. Hindy, D. Brosset, E. Bayne, A. K. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104650–104675, 2020.

[81] Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, and H. Han, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Comput. Secur.*, vol. 116, May 2022, Art. no. 102675.

[82] A. Kenyon, L. Deka, and D. Elizondo, "Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets," *Comput. Secur.*, vol. 99, Dec. 2020, Art. no. 102022.

[83] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014, doi: 10.1016/j.compeleceng.2013.11.024.

[84] M. Grzenda, H. M. Gomes, and A. Bifet, "Delayed labelling evaluation for data streams," *Data Mining Knowl. Discovery*, vol. 34, no. 5, pp. 1237–1266, Sep. 2020, doi: 10.1007/s10618-019-00654-y.

[85] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "MOA: Massive online analysis, a framework for stream classification and clustering.," in *Proc. 1st Workshop Appl. Pattern Anal.*, Sep. 2010, pp. 44–50.

[86] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, no. 3, pp. 317–346, Mar. 2013, doi: 10.1007/s10994-012-5320-9.

[87] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," *Expert Syst. Appl.*, vol. 29, no. 4, pp. 713–722, Nov. 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417405000989

[88] Z. Cataltepe, U. Ekmekci, T. Cataltepe, and I. Kelebek, "Online feature selected semi-supervised decision trees for network intrusion detection," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp.*, Apr. 2016, pp. 1085–1088.

[89] N. Kunhare, R. Tiwari, and J. Dhar, "Particle swarm optimization and feature selection for intrusion detection system," *Sādhanā*, vol. 45, no. 1, pp. 1–14, Dec. 2020.

[90] S. McElwee, "Active learning intrusion detection using k-means clustering selection," in *Proc. SoutheastCon*, Mar. 2017, pp. 1–7.

[91] Q.-V. Dang, "Active learning for intrusion detection systems," in *Proc. RIVF Int. Conf. Comput. Commun. Technol. (RIVF)*, Oct. 2020, pp. 1–3.

[92] N. Capuano, G. Fenza, V. Loia, and C. Stanzione, "Explainable artificial intelligence in CyberSecurity: A survey," *IEEE Access*, vol. 10, pp. 93575–93600, 2022.

[93] G. De Francisci Morales, "SAMOA: A platform for mining big data streams," in *Proc. 22nd Int. Conf. World Wide Web*, May 2013, pp. 777–778, doi: 10.1145/2487788.2488042.

[94] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, and A. Bifet, "River: Machine learning for streaming data in Python," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 1–10, Jan. 2020.

[95] A. Bifet, S. Maniu, J. Qian, G. Tian, C. He, and W. Fan, "StreamDM: Advanced data mining in spark streaming," in *Proc. IEEE Int. Conf. Data Mining Workshop (ICDMW)*, Atlantic City, NJ, USA, Nov. 2015, pp. 1608–1611, doi: 10.1109/ICDMW.2015.140.

[96] P. Reutemann and J. Vanschoren, "Scientific workflow management with Adams," in *Machine Learning and Knowledge Discovery in Databases*, P. A. Flach, T. De Bie, and N. Cristianini, Eds., Berlin, Germany: Springer, 2012, pp. 833–837.

[97] V. Jacobson, C. Leres, and S. McCanne, "The TCPDUMP manual page," Lawrence Berkeley Lab., Berkeley, CA, USA, Tech. Rep. 1, 1989, vol. 143, p. 117.

[98] C. Satten, "Lossless gigabit remote packet capture with Linux. University of Washington network systems," Netw. Syst., Univ. Washington, Tech. Rep. 1, 2008.

[99] International Computer Science Institute (ICSI). (2023). *The Zeek Network Security Monitor*. Accessed: Mar. 23, 2023. [Online]. Available: https://zeek.org/

[100] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy*, 2016, pp. 407–414.

[101] C. Bullard. (2023). *Argus*. Accessed: Mar. 29, 2023. [Online]. Available: https://openargus.org/

[102] Z. Aouini and A. Pekar, "NFStream: A flexible network data analysis framework," *Comput. Netw.*, vol. 204, Jan. 2022, Art. no. 108719.

[103] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. Journal: A Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, Apr. 2016.

[104] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: The CICIDS2017 case study," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2021, pp. 7–12.

[105] 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. 22351–22370, 2021.

[106] G. Y. Sakurai, J. F. Lopes, B. B. Zarpelão, and S. Barbon Junior, "Benchmarking change detector algorithms from different concept drift perspectives," *Future Internet*, vol. 15, no. 5, p. 169, Apr. 2023.

[107] H. Sun, W. Huang, J. Weng, Z. Liu, W. Tan, Z. He, M. Chen, B. Wu, L. Li, and X. Peng, "CCID-CAN: Cross-chain intrusion detection on CAN bus for autonomous vehicles," *IEEE Internet Things J.*, vol. 11, no. 15, pp. 26146–26159, May 2024.

[108] T. Zhang, C. Du, Y. Zhou, Q. Guan, Z. Liu, X. Huang, Z. Gong, L. Deng, and Y. Li, "Hybrid transfer and self-supervised learning approaches in neural networks for intelligent vehicle intrusion detection and analysis," *IEEE Internet Things J.*, vol. 12, no. 7, pp. 7677–7692, Jul. 2024.

[109] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, May 2017.

[110] B.-J. Hou, L. Zhang, and Z.-H. Zhou, "Prediction with unpredictable feature evolution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5706–5715, Oct. 2022.

[111] E. R. Faria, I. J. C. R. Gonçalves, A. C. P. L. F. de Carvalho, and J. Gama, "Novelty detection in data streams," *Artif. Intell. Rev.*, vol. 45, no. 2, pp. 235–269, Feb. 2016.

[112] S. Prasath, K. Sethi, D. Mohanty, P. Bera, and S. R. Samantaray, "Analysis of continual learning models for intrusion detection system," *IEEE Access*, vol. 10, pp. 121444–121464, 2022.

RODRIGO SANCHES MIANI received the master's and Ph.D. degrees in electrical and computer engineering from the State University of Campinas (UNICAMP), including an Internship period with the University of Maryland. Since 2013, he has been a Professor with the Faculty of Computing (FACOM), Federal University of Uberlândia (UFU). He is currently the Creator and the Coordinator of the Cybersecurity Specialization Course, FACOM/UFU, the Founder of the Cybersecurity Center (NuSec), FACOM/UFU, and a member of the steering committee of the Special Commission on Cybersecurity (CESeg) of the Brazilian Computer Society (SBC). He is also a Permanent Member of the Graduate Program in Computer Science (PPGCO), FACOM/UFU, and the Coordinator of several research and development projects in cybersecurity and related fields. His research focuses on the intersection of cybersecurity and artificial intelligence, with interests that include the development and evaluation of machine learning-based intrusion detection systems, static and dynamic malware analysis, and cyber threat intelligence.

GUSTAVO DI GIOVANNI BERNARDO is currently pursuing the Ph.D. degree with the Faculty of Computing, Federal University of Uberlândia. His research interests include intrusion detection systems, machine learning, and data stream mining.

GUILHERME WEIGERT CASSALES received the Ph.D. degree in computer science from the Graduate Program in Computer Science, Federal University of São Carlos (UFSCar), in 2021. During his Ph.D., he conducted part of his research with the University of Waikato, New Zealand. He is currently a Postdoctoral Fellow with the AI Institute, within the Division of Health, Engineering, Computing, and Science (HECS), University of Waikato. His research interests include dataflow learning and distributed systems.

HERMES SENGER is currently an Associate Professor with the Federal University of São Carlos, Brazil. He is an Enthusiastic Researcher for improving the efficiency of applications across different computer architectures, from tiny devices with limited capabilities to supercomputers and distributed systems. The research goals may vary, from improving existing algorithms and developing new ones to enhancing the efficiency of applications on either existing or new architectures.

ELAINE RIBEIRO DE FARIA received the bachelor's and master's degrees in computer science from the Federal University of Uberlândia (UFU), in 2002 and 2006, respectively, and the Ph.D. degree in computer science from the University of São Paulo, in 2014. She was a Ph.D. Sandwich Internship with the University of Porto, Portugal, from 2012 to 2013. Since 2010, she has been with UFU, where she is currently an Associate Professor. From May 2024 to April 2025, she was a Visiting Researcher/a Professor with the University of Birmingham, supported by CAPES/PRINT funding. She is a Permanent Faculty Member of the Graduate Program in Computer Science and coordinates the extension project #includegirls. Her research interests include machine learning, data mining, and streaming data.