Survey of Network Intrusion Detection Methods From the Perspective of the Knowledge Discovery in Databases Process

Borja Molina-Coronado, Usue Mori, Alexander Mendiburu, *Member, IEEE*, and Jose Miguel-Alonso, *Member, IEEE*

***Abstract*—The identification of network attacks which target information and communication systems has been a focus of the research community for years. Network intrusion detection is a complex problem which presents a diverse number of challenges. Many attacks currently remain undetected, while newer ones emerge due to the proliferation of connected devices and the evolution of communication technology. In this survey, we review the methods that have been applied to network data with the purpose of developing an intrusion detector, but contrary to previous reviews in the area, we analyze them from the perspective of the Knowledge Discovery in Databases (KDD) process. As such, we discuss the techniques used for the collecion, preprocessing and transformation of the data, as well as the data mining and evaluation methods. We also present the characteristics and motivations behind the use of each of these techniques and propose more adequate and up-to-date taxonomies and definitions for intrusion detectors based on the terminology used in the area of data mining and KDD. Special importance is given to the evaluation procedures followed to assess the detectors, discussing their applicability in current, real networks. Finally, as a result of this literature review, we investigate some open issues which will need to be considered for further research in the area of network security.**

***Index Terms*—Network intrusion detection, anomaly, misuse, data mining, network data, dimensionality reduction, preprocessing, traffic analysis.**

I. Introduction

INTRUSION Detection Systems (IDSs) are deployed to uncover cyberattacks that may harm information systems. An IDS works by analyzing different kinds of data: operating system-related data (Host-based Intrusion Detection System, HIDS) or network-related data (Network Intrusion Detection System, NIDS) [1].

HIDS are designed to detect cyberattacks targeting an individual computer, typically a server. To do so, they process local host data such as operating system calls made by applications, log information or network traffic that reaches the machine [2]. NIDS operate differently, focusing exclusively on network traffic analysis, with the aim of finding malicious patterns targeting all the machines or devices inside the protected network [3]. Throughout this paper, we will focus our attention on NIDS.

Finding attack indicators in network traffic data presents important particularities [4], [5] such as (1) the growing number of attacks, which arise and change as new vulnerabilities are identified; (2) the diverse and evolving nature of network traffic; (3) the continuously growing data volume, which challenges the successful analysis of network traffic; and (4) the increasingly common use of encrypted data, not easily accessible to detectors. These facts, along with the need for near real-time responses, make intrusion detection a very hard problem.

Data mining techniques, which are applied successfully in many fields, have also proven useful to implement NIDS. These techniques can find complex relationships in the data, which can be used to detect network attacks. However, off-the-shelf data mining algorithms cannot be applied directly to network data, as they cannot cope with the particularities listed above. Intrusion detection is the result of a complex process that starts with the collection of network data and continues with the preparation and preprocessing of this data. Only then will a detector be able to deliver meaningful results.

In spite of this, NIDS research normally focuses only on the application of data mining algorithms to network datasets, paying minimum attention (or no attention at all) to questions such as: Which type of data (raw traffic, flows, encrypted/plaintext/no payload, etc.) will the NIDS receive? Can all the data traversing the network be gathered and processed, or just an excerpt of it? Where will the NIDS be located? How good is the performance of the NIDS in terms of both accuracy and response time? Can the detection model be updated easily to detect new threats? In this paper we aim to provide some answers to these questions.

Several surveys of NIDS research have been published already. Table I summarizes their scope. Most of them include

Manuscript received January 28, 2020; revised June 9, 2020; accepted August 10, 2020. Date of publication August 12, 2020; date of current version December 9, 2020. This work has received support from the TIN2016-78365-R (Spanish Ministry of Economy, Industry and Competitiveness http://www.mineco.gob.es/portal/site/mineco) and IT-1244-19 (Basque Government) programs. Borja Molina Coronado holds a predoctoral grant (ref. PRE_2019_2_0022) by the Basque Government. The associate editor coordinating the review of this article and approving it for publication was J.-H. Cho. *(Corresponding author: Borja Molina-Coronado.)*

Borja Molina-Coronado, Alexander Mendiburu, and Jose Miguel-Alonso are with the Department of Computer Architecture and Technology, University of the Basque Country UPV/EHU, 20018 Donostia, Spain (e-mail: borja.molina@ehu.es; alexander.mendiburu@ehu.es; j.miguel@ehu.es).

Usue Mori is with the Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, 20018 Donostia, Spain (e-mail: usue.mori@ehu.es).

Digital Object Identifier 10.1109/TNSM.2020.3016246

TABLE I
SUMMARY OF NIDS SURVEYS AND THEIR COVERAGE OF THE DIFFERENT COMPONENTS THAT CONFORM A FUNCTIONAL NIDS

Survey	Data collection	Data Preprocessing and Transformation	Data mining	Detector updates
Garcia et al., (2009) [6]			✓*	
Sperotto et al., (2010) [7]	✓*		✓*	
Davis et al., (2011) [8]	✓	✓*		
Bhuyan et al., (2014) [9]			✓*	
Ahmed et al., (2015) [10]			✓*	
Buzcak et al., (2016) [11]			✓	
Umer et al., (2017) [12]			✓*	
Moustafa et al., (2019) [13]	✓*	✓*	✓*	
Khraisat et al., (2019) [14]	✓*		✓	
This Survey	✓	✓	✓	✓

*Partial coverage

only a partial coverage of the steps that are necessary to perform intrusion detection by means of network traffic analysis. As can be seen in the table, most published NIDS surveys overlook data collection [9]–[12], [14], [15], while others cover it but only partially, either providing descriptions that are too shallow [13], or focusing on a subset of the possible sources of data [7]. Data preprocessing and transformation phases have received a similar lack of analysis: the number of surveys covering these tasks is small, with limited content and they ignore their associated problems and challenges [8], [13].

In contrast, the data mining phase has been the focus of almost all NIDS surveys. An extensive description of a variety of methods that constitute the "core" of NIDS (the detection engine) is given in [11], [14]. Some papers also focus on this phase but for particular detector types. For example, surveys [6], [9], [10], [13] review only detection techniques based on the discovery of anomalous network traffic, while in [7], [12] the focus is only on approaches that use a specific type of input data, ignoring detectors that use other data sources (such as raw packet data). Finally, we are not aware of any survey addressing detector updates that aim to deal with the evolving nature of traffic and attacks [16], an essential feature of a production NIDS.

We reckon that previous reviews transmit the idea that the only relevant part of a NIDS is the data mining algorithm selected or devised to identify attacks. However, NIDS actually comprise many processes that have not been covered in enough detail, e.g., how data is extracted, preprocessed and transformed; or how the detection engine is evaluated and updated. These additional processes play a fundamental role in the detection ability of a NIDS, and deserve deeper analysis.

In this work, we review NIDS proposals using the Knowledge Discovery in Databases (KDD) process as a guideline. KDD describes the procedures commonly used to find explainable patterns in data, allowing the interpretation or prediction of future events [17], and the data mining phase is only a stage of this process. The main contributions of this paper are:

- We describe the basic blocks that comprise a functional NIDS from the perspective of the KDD process, going beyond the selected data mining algorithm used for network attack detection.
- We review the techniques used in all the steps of this process, paying special attention to the motivation behind their use.
- We propose a taxonomy of NIDS detection methods based on the data mining terminology to avoid the ambiguity of previous classification proposals. To that end, we use two different criteria: (a) the detection approach and (b) the learning approach.
- We review the mechanisms to avoid deterioration in detection ability due to the evolution of network traffic, including attacks.
- We discuss NIDS validation and evaluation procedures, identifying common mistakes such as the use of inadequate metrics based on unrepresentative datasets.
- We discuss current challenges in NIDS research based on the surveyed literature. We analyze the processes and steps followed by authors to build NIDS, identifying common assumptions and mistakes, and highlighting future research lines in the area of network security.

The remainder of this paper is organized as follows. First, we briefly depict some basic concepts about how network attacks can manifest themselves and describe the KDD process and its phases. In the following sections we enumerate and categorize the most representative techniques proposed in the NIDS literature for the different phases of the KDD process. We start with the collection of network traffic data. In Sections IV and V, we analyze methods for the extraction of structured feature records from that data, as well as the transformations that are commonly applied to those features. Techniques to remove data redundancies and accelerate the application of data mining algorithms are discussed in Section VI. In Section VII we pay attention to the data mining phase, analyzing different learning methods and detection approaches. In Section VIII we show the evaluation criteria used by researchers, and also discuss their suitability for NIDS. This panoramic view of NIDS proposals allows us to close the survey in Section IX discussing present and future challenges in NIDS research, identifying common mistakes and unrealistic assumptions in the network security area.

II. BASIC CONCEPTS

In this section we introduce a collection of basic concepts that are necessary to describe how a NIDS operates. First,

we focus on network attacks (those that a NIDS must detect) and the way they manifest if network traffic is analyzed. We also describe the Knowledge Discovery in Databases (KDD) process and its different phases.

A. Networks Under Attack

The CIA triad (Confidentiality, Integrity and Availability) is a widely known set of security properties used in information security management for the definition of security policies and procedures [18]. Confidentiality refers to the efforts to keep data private or secret, controlling access to prevent unauthorized disclosure. Integrity is about guaranteeing that data has not been altered and, therefore, can be trusted: it is correct, authentic and reliable. Availability measures seek to ensure that authorized users have timely and reliable access to resources when they are needed. From the perspective of information security, any attempt to compromise any of the policies and mechanisms set up by an organization to guarantee the CIA properties of a system can be considered an attack [19].

When an attacker uses the organization's network to reach their target, the corresponding action is considered a *network attack* [20]. In this context, the attacker is also known as an *intruder*, and the action as an *intrusion.* An analysis of the traffic managed by the network should help to detect network attacks by finding the leads that the intrusions have left behind. This is precisely what NIDS do: they are fed with network data and search for clues that may uncover malicious activities. There is a large variety of attacks, with different targets and purposes, and the clues and symptoms they leave are not always obvious. Table II lists some broad groups of known network attacks.

As stated before, network intrusion detection is only possible if network attacks leave evident or latent traces in the data collected by the NIDS. These traces may be visible only from some specific vantage points and may appear at different traffic inspection levels [21].[1] Attack manifestations can be classified as.

- **Point**. The malicious activity affects one data sample, e.g., a packet, a connection. In this case, an attack leaves a footprint in a single record, meaning that some of their values are not permitted or constitute an anomaly. For example, SQL injection is carried out by placing code in the parameters of a HTTP request with the aim of manipulating the behavior of a Web application. This attack can be detected by analyzing a single sample of HTTP payload.
- **Contextual**. A particular event is not an anomaly in itself, but may indicate that an attack is happening due to its unexpected appearance given the context. For example, a HTTP (Web) request to a DNS server is not an anomaly by itself, but could be indicative of a service discovery attempt over a server.
- **Collective**. The attack shows up as several characteristic events which, together but not individually, constitute a malicious event. For example, a large number of simultaneous requests to a common service form an anomaly, although each individual connection is not anomalous by itself.

Given the diverse forms in which a malicious event may leave a footprint, different strategies need to be taken during the construction of a NIDS. The successful identification of network attacks is not only related to the choice of a good data mining algorithm, but also to the information available (collected and prepared) to feed it.

B. The Knowledge Discovery in Databases Process

The Knowledge Discovery in Databases (KDD) process describes the procedures commonly used to find explainable patterns in data, allowing the interpretation or prediction of future events [17]. KDD was defined formally in [23] as:

> *"KDD is the organized process of identifying valid, novel, useful, and understandable patterns from large and complex data sets"*

KDD is an iterative process, as depicted in Figure 1, comprising the following steps [17].

- **Data collection** consists of capturing and selecting raw data from their sources. This data should provide the necessary information to solve the problem at hand.
- **Data preprocessing** involves the extraction of a valid and structured set of features required for knowledge extraction. It also includes the application of cleaning methods to avoid factors such as noise, outliers or missing values, and the derivation of new features from others to structure the data in a way appropriate for the application of the algorithms of the following phases.
- **Data reduction** is a necessary step to optimize the application of data mining algorithms, reducing their computation time and/or improving their results. It involves feature subset selection (removal of useless or redundant features), dimensionality reduction (projection of a set of features into a smaller space) or modifications of the set of samples (adding or removing samples).
- **Data mining** is the core of the KDD process, but it can be successful only if the previous steps are performed properly. It consists of selecting and applying techniques, including Machine Learning (ML) algorithms, to extract knowledge from data by finding useful patterns and relations between features. The choice of a specific data mining algorithm depends on the problem at hand and the nature of the available data.
- **Interpretation and evaluation** consists of, using a set of metrics, measuring the performance and effectiveness of the data mining algorithm, and, consequently, of the tasks carried out in previous steps of the KDD process. It also comprises result visualization and reporting.

Given the iterative nature of this process, some steps may have to be reapplied and rethought more than once to achieve satisfactory results [24]. For example, a feature initially excluded during the data transformation step may be

[1] In this introduction we use the classification made by Thottan and Ji [21] to introduce how attacks become apparent at different traffic inspection levels. However, different alternative taxonomies are available, see [18], [20] and [22].

TABLE II
EXAMPLES OF BROAD CLASSES OF NETWORK ATTACKS, INDICATING THE MAIN CIA PROPERTIES THEY COMPROMISE

Attack	Confidentiality	Integrity	Availability	Description
Brute force	✓			Repetitive attempt to carry out some action, such as authentication or resource discovery, often guided by a dictionary.
Injection (SQL, command...)	✓	✓		Exploitation of weaknesses in input field sanitizing mechanisms to execute code or commands on the target system.
Privilege escalation	✓			Attempt to gain unauthorized access from an unprivileged account.
Spoofing	✓	✓		A person or program masquerades as another, falsifying data to gain an illegitimate advantage.
Sniffing	✓			Eavesdropping traffic in order to obtain information.
(Distributed) denial of service			✓	Resource abuse intended to deny access to a service to its legitimate users.

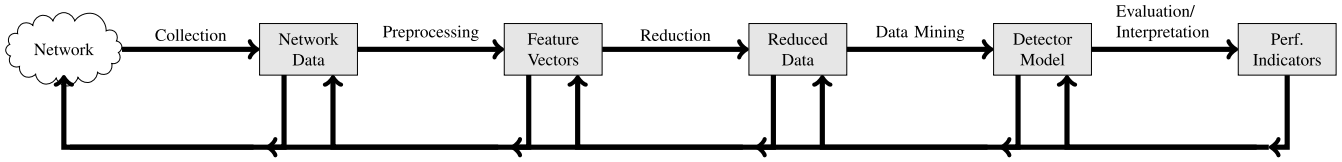


Fig. 1. Illustration of the KDD process for NIDS, showing its iterative nature, the results of each stage, and the possibility of going back and rethinking any previous stage.

added later in order to increase the performance of a particular data mining algorithm.

In the NIDS context, the KDD process starts with the collection of network data. Preprocessing is essential to organize the large and diverse set of collected data, transforming it into a set of structured records with a common set of features. Data mining can be applied to this collection of records, or to a reduced version (after some data reduction procedures), and then evaluated using adequate metrics: detection rate, false alarm rate, etc. With some variations, this should be the NIDS pipeline. However, most reviewed papers start with publicly available datasets, already collected and preprocessed. An advantage of this approach is the reduction in the number of tasks (focusing mostly on the data mining part), and the availability of a common evaluation framework for the comparison of NIDS proposals. This approach also entails several important drawbacks, which will be discussed in detail in the following sections.

III. DATA COLLECTION STAGE

The first phase of the KDD process involves the collection of the data with which network intrusions are detected. Network data[2] can be gathered at different locations.

Commonly, networks are organized in a hierarchical (tree) structure where the devices (typically, routers and switches) have visibility of the traffic entering and leaving the machines connected below them. Higher level (root or core) devices have a complete view of Internet-related traffic going to and coming from the computers (and network devices) located below their level. However, they have a limited view of horizontal communications (those between machines at the same level). The opposite is true for lower level (aggregation and access) devices [25]. Thus, the choice of the level at which to carry out the collection of network data has a bearing on the coverage of a NIDS and, consequently, on the volume of traffic to be analyzed and on the set of network attacks that can be detected.

Data collected from root network nodes corresponds to traffic coming from or going to a large number of network elements. As such, network data obtained at this level could allow Internet-related attacks to be detected by targeting as many network-connected machines as possible. However, some attacks would remain undetected because sometimes malicious activities go horizontally, and does not involve the root nodes. NIDS using root level captures need to process huge data volumes and must support extremely high throughput to avoid discarding traffic and to minimize detection delays. Also, due to the number of network elements they cover, they are easily affected by noise and perturbations, for example the connection of a new device on any segment of the network [26].

In contrast, the analysis of incoming and outgoing traffic of a small portion of the network, and captured from a lower level device, for example a datacenter switch, makes detectors more immune to noise and perturbations which occur at other network segments. Consequently, NIDS using this data require more modest throughput levels [26]. However, by design, their coverage is restricted to events occurring on a specific network segment.

Other scenarios are possible. Distributed collection of network data aims to provide a trade-off between network

[2]Throughout this paper we will use "network data" as a generic term to refer to the information provided to the NIDS to carry out its detection activities. A particular class of network data is "network traffic", with which we refer to a collection of captured packets with none or minimum processing. Note that, with this definition, "network data" may include also higher-level information including, for example, per-connection source/destination devices or average packet inter-arrival times.

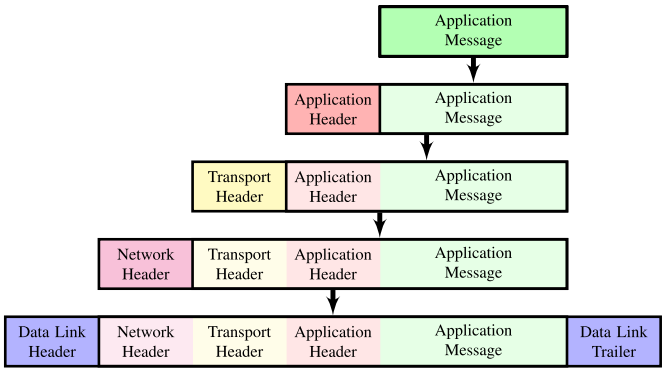


Fig. 2. Sketch of a network packet showing protocol encapsulation. Actual headers (fields, size, options) depend on the actual protocols used at the link layer (Ehternet/PPP/...), at the network layer (IPv4/IPv6), at the transport layer (TCP/UDP/ICMP/...), etc.

coverage, performance and stability. Partial captures and/or measurements may be carried out at several network points and gathered at a central location, overcoming the drawbacks of using any of the two previously described approaches individually [27].

Data collected by network probes to feed NIDS may have very different characteristics. Sometimes it consists of full copies of all data packets, such as those captured by a network sniffer. Another option is feeding the NIDS with traffic summaries (measurements), normally in terms of per-flow data samples, at the cost of having a limited view of the traffic that is traversing the network. We elaborate on these concepts in the following sub-sections.

A. Capturing Raw Packet Data

Communication networks move data units, which are commonly known as packets, from one node to another. Every packet consists of protocol information arranged in layers which encapsulate the payload, i.e., the actual data being transferred. Each layer contains a set of headers (control information) which is appended to the information of the above levels and required by network devices in order to be delivered to its destination [25].[3] This concept is known as encapsulation. Figure 2 shows the structure of a typical network packet and the encapsulation concept.

Protocol layers are necessary to move a message through the network. But not all the applications use the same protocol sets for this purpose. This means that not all the packets have the same headers with the same fields, and even equivalent fields for two different protocols may be located at different positions within the packet [25]. Also, the nature of the different fields that conform a network packet is heterogeneous, with text (user messages), numbers (lengths, window sizes), categorical information (such as control flags, IP addresses and port numbers), etc. This means that the size of network packets is also variable: from packets containing only control information necessary for protocol operation but without payload, to large application messages that need to be split into several sub-messages (due to limits imposed by some protocols) and are reassembled at the destination. All these factors complicate the creation of a uniform and structured dataset that will be needed to apply data mining methods.

Packet capturing tools (a.k.a. network sniffers) such as *Tcpdump* [28] or *Gulp* [29] can be used to retrieve and store raw network packets. They can run in a single computer, essentially capturing the packets: arriving to/departing from that machine (in-line mode), or from a network device configured to mirror all packets to the port to which the sniffer is connected (mirroring mode). In contrast to in-line mode, mirroring allows traffic from many machines to be captured but has a bearing on the performance of the device [26], which is affected by the increasing number of machines to be monitored and the growing speed of modern networks.

Capturing raw packets is a challenging task that requires high computational and storage requirements [30]. Insufficient bandwidth in the device/link used to carry out the captures, or in the machine in charge of storing/processing them, may cause packet losses [26], that may in turn reduce the detection abilities of the NIDS. In addition to that, if packets are ciphered using, for example, IPSec [31] (available for IPv4 and IPv6), access to the contents of the packets, including the headers of higher-level protocols, is impossible unless additional decryption mechanisms are implemented. Access to this raw packet captures, in contrast with pre-processed data, is of great interest as it enables the extraction of any kind of traffic-related feature that may enhance the detection ability of a NIDS.

B. Capturing Flow-Level Data

A *flow* is a collection of related packets that share a common set of attributes. These include network, transport or application header fields (IP addresses, IP protocol, TCP/UDP source ports, Type of Service...) as well as information related to the handling of packets on the device, such as physical input and output ports. Flows can be either unidirectional or bidirectional; in the later case they are also called *sessions* [32].

Flow-related features can be derived from raw packet data, as will be explained in Sections IV-A2 and IV-A3. However, the collection and reporting of flow-level data records is an increasingly common capability of network devices such as routers or switches. This feature aims to facilitate network management and supervision. Device manufacturers implement their own mechanism to gather and transfer this data, commonly adhering to the IPFIX standard [32], which was derived from Cisco's NetFlow v9 [33]. IPFIX-capable devices compute flow measurements in near real-time, without storing raw traffic, making use of sampling and accounting procedures [34].

Flow records computed in the device are exported to a *collector*, which stores them [32]. Each record contains the basic information about a flow, plus additional elements, including among others[4]: time stamps for flow start/finish times, number of bytes/packets observed in the flow, the set of TCP flags observed over the life of the flow, etc. Note that the payload, the actual end-user application data interchanged by means of

[3]Unless otherwise stated, throughout this document we assume that we are dealing with IP datagrams.

[4]A complete list of IPFIX elements can be found at https://www.iana.org/assignments/ipfix/ipfix.xhtml.

the packets, is an optional field of flow records and, when present, it normally includes just a few octets. Therefore, flow captures may be far more compact than raw packet captures. However, working with flow data adds latency that degrades real-time operation [26], [34].

Network flow data is useful to detect network attacks that leave footprints in traffic patterns such as traffic volume, timings of packets and/or communicating endpoints (sources and destinations of data: machines and/or ports). This helpfulness for NIDS is ultimately dependent on the sampling mechanism used to perform flow measurements. Sampling methods take partial measures to obtain an approximation of the actual values when the number of connections is too high [35]. They reduce the impact on the performance of the device: only 1 out of n packets are checked, n being a manager-selected parameter. Accordingly, flow sampling parameters determine the behavior of the estimation process and play an important role in the extraction of flow records with useful (accurate) information for intrusion detection [36], [37]. For example, choosing a high sampling rate (meaning that a large number of packets per connection are checked) reduces the risk of obtaining poor estimates and the potential loss of vital information for NIDS [36], but there is a performance penalty.

IV. Preprocessing Stage: Derivation of Features

The preprocessing stage involves the derivation and transformation of feature vectors from the collected network data. As a result, a dataset of structured records, each of them composed of a set of explanatory features and usually, a response variable; is obtained.

A. Derivation of Explanatory Variables

Explanatory variables or features describe/represent the characteristics of a traffic observation (packet, connection, session). The derivation of *explanatory variables* or *features* is an important task of KDD because the detection capability of data mining methods highly depends on the information provided by the input variables. In this section, we discuss the different types of explanatory features, the procedures required to obtain them from network collected data and their usefulness for detecting network attacks.

Feature extraction can be done at different levels to enable network traffic analysis (methods to infer information or patterns from network-related data such as timings and counts of network packets) even when encryption is used [38]. We use the scheme proposed by Davis and Clark [8] to describe those levels (using *flow* however, instead of *connection*-based aggregations) and the different classes of variables that are obtained at each level (see Figure 3).

1) Packet Header Features: A raw packet can be converted into a feature vector by parsing the distinct set of header fields of the different protocols in use (from different layers of the TCP/IP stack). Packet header features are useful to detect protocol misbehavior and low level network attacks that aim to provoke protocol malfunction, such as spoofing (impersonation) attacks or attacks that do not leave their footprint in the application message [39].

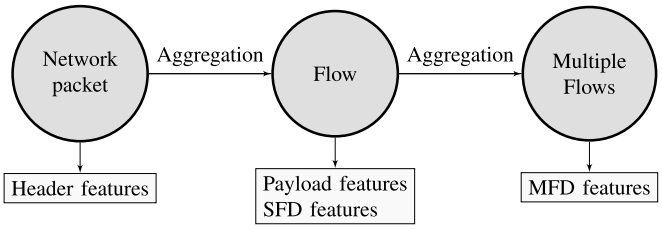


Fig. 3. Feature derivation process.

Given the diversity in the structure of raw packets, and in the data types of the different protocol headers, the derivation of a common vector of features from this data [39] can be done in two different ways: (1) assigning missing values to those features that do not appear in a specific packet, or (2) removing fields that are not common to all protocols at the same layer. While the former leads to a more sparse representation of the data, the latter technique results in the loss of information that may be relevant for the detection of some attacks. In order to deal with both shortcomings, some authors proposed using different detectors: each one working with a particular application protocol (service) [40] or each one analyzing the headers of different protocol layers (for IP and TCP) of the protocol stack [41].

A main issue when working with packets (raw or converted into feature vectors) is not the preprocessing (parsing) procedure, which is straightforward, but the enormous throughput of packets sent to/received from a broadband link. As previously mentioned, capturing, preprocessing and storing the packets in real time is a challenging procedure of growing difficulty, which can be partially addressed through the use of different processing points working in parallel [30]. Also, encryption such as IPsec used at the network layer (either with IPv4 or IPv6) makes extracting higher-level header features a challenging procedure; only network and data link headers can be extracted easily without decryption mechanisms. In these scenarios, a per-packet analysis becomes difficult, but some forms of network traffic analysis (using flow-level data) are still feasible [7].

2) Payload Features: Payload features are those obtained from the content of network messages, that is, the data exchanged between applications. For NIDS, this content is commonly analyzed using techniques based on natural language processing [42], deriving vectors with the frequencies of characters [40], words [41] or n-grams (all subsequences of n consecutive symbols) [43]–[46] that appear in the payload.

Payload features are useful to detect network attacks that may exploit host vulnerabilities, such as injection or shell-code attacks (buffer overflows, privilege escalation, etc.). These attacks do not leave a specific footprint on packet headers, but the attack evidences can be found inside application-level messages [44], [45].

One of the main characteristics of payload features is their high dimensionality. Note that, in general, the feature vectors obtained from payloads are very long, and may require additional dimensionality reduction procedures (performed in posterior phases of the KDD process).

For connection-oriented applications, an application message may be split before transmission into a collection of packets; the number of packets depends on factors such as the message size or the maximum segment size parameter [25]. Therefore, processing methods aiming to extract meaningful features from application payload need to reassemble the packets to reconstruct the original message.

Processing application messages is also challenging in terms of computing capacity, memory and speed. The transmission of high volumes of data at high speed and involving connections between a large number of devices may stress not only the capture mechanisms, but also the feature extraction tasks. Enough capacity (processing speed, RAM, storage) must be guaranteed to properly carry out these tasks [7].

Encryption algorithms, that may be used at any protocol level to provide security (confidentiality, integrity, authentication), make the extraction of payload features even more difficult [47]. Efficient decryption mechanisms are essential in those scenarios if access to plaintext (non-encrypted) payload is a requirement. Such functionality typically requires additional processing infrastructure, such as TLS interception proxies that cut off encrypted traffic to decipher and inspect it before being forwarded to the actual server [48], or modifying the ciphering suites used for communication [49]. Drawbacks of these mechanisms lie in the increase of network latency and in the reduction of service security [50]. In practice, access of NIDS to the contents of encrypted messages is an inconvenient task, and the use of HIDS may be more adequate as they have access to plaintext messages [51].

3) Single Flow Derived Features (SFD): SFD features are those that correspond to a collection of packets sharing any property, i.e., a flow. Packets may be aggregated using different criteria until a given event, such as the end of a TCP connection or a timeout trigger, is detected [52]. Then SFD features are extracted to summarize different properties of this aggregation of packets, that can be either statistical measures or common header fields (from IP and transport layers). Features directly extracted from application messages (payload features) are not considered SFD features. As a result of this derivation, a mixture of nominal and numerical variables for each record (flow) is obtained [52]. Tools to compute SFD features from captures of raw packets include Argus [53] and Bro [52]. However, a more common approach to obtain SFD features uses the flow-related information provided by IPFIX-capable network devices.

A few NIDS papers that make use of SFD features extracted from raw packets superficially describe how these features have been derived. Different header fields are used as aggregation keys, the most common ones being the source IP address or the source-destination IP address pair [16], [54], [55]. In [56] IP address pairs and TCP/UDP ports are used as aggregation keys, and those values are used as SFD features. Other header fields common to all packets in a flow, such as initial sequence numbers and TTL values, have been used as SFD features in [57]–[59], together with aggregated measurements such as: dropped and re-transmitted packets; average, maximum and minimum packet sizes; inter-arrival times; bit/packet rates from source to destination and vice-versa, etc. In [60], features derived from header fields are not considered valid, as they can be spoofed by attackers. They use the same aggregation keys, but instead they analyze average packet inter-arrival times and average packet lengths. A set of more complex SFD features, such as the coefficients from polynomial and FFT decomposition of inbound and outbound packet sizes, is derived in [61], to prove that their utilization makes a NIDS more resilient to some obfuscation attacks, explained at the end of this section.

SFD data sets are much smaller than raw data sets because payload is removed and common header features are summarized. This compactness is fundamental for many NIDS, since it enables the operation in near real-time in high-speed networks [7], although care must be taken when sampling is used, as vital information may be lost, see Section III-B. Also, SFD features are useful for network traffic analysis even when encryption is used, without the need for additional decryption mechanisms [62].

SFD features enable the detection of network attacks that leave a footprint in the summarized statistics of a flow, e.g., affecting the volume, timings or the number of connections. These footprints also include the presence of connections in unexpected contexts, or connections with suspicious/uncommon header values (port numbers, source addresses), etc., [26]. Some works have proposed the use of SFD features to detect network attacks that are normally searched for inside the payload [63], [64]. However, the viability of this approach is questionable, given the high false alarm rates they report.

Basic SFD features can be easily tampered with by intruders by means of obfuscation techniques. These techniques have been proposed as a protection measure to shield network activities against network traffic analysis [65]. They include the insertion of artificial inter-packet delays, and the injection of malformed and spurious packets. Unfortunately, attackers can also use obfuscation, modifying harmless and attack traffic to make their SFD statistics indistinguishable, not allowing detection by the NIDS [66], [67].

4) Multiple Flow Derived Features (MFD): MFD features are derived aggregating information pertaining to multiple flow records. Thus, they contain higher level statistics providing a more abstract insight about the traffic traversing the network, and enabling additional traffic analysis capabilities. To compute MFD data, flows are commonly grouped using criteria such as "flows inside a time window T" or "last n flows". Then, sets of statistics are computed for each group [68]. Note that packets within a flow share many properties (SFD features), while flows aggregated in a group may not share any common characteristic beyond being in temporal and, sometimes, spacial proximity.

Normally, NIDS papers proposing the use of MFD features do not provide enough details about their derivation. The main aggregation criteria used are time windows and flows coming from/going to the same IP address (or to the same set of network prefixes in coarse grain scenarios). In [54], [55] those aggregations are used to extract MFD features such as counts of different ports and hosts, packets per second, ratios of packets with specific TCP flags (RST, SYN) set, etc.

Entropy values that indicate the variability of discrete SFD features (such as source and destination IP addresses and ports) are MFD features used in [5], [68], [69]. Covariance matrices are obtained in [70], [71] to embody the distribution of continuous SFD features. In [39] MFD features are computed using a time window: for every connection, the number of past connections with similar SFD values for source and destination IP addresses, service and protocol flags are computed. Other authors [72] use a fixed-width window of flows as aggregation criterion to subsequently apply the Discrete Wavelet Transform [73] and use the coefficients of this transform as MFD features.

MFD features are useful for identifying attacks that spread their activities across multiple connections, such as flooding attacks (DoS attacks) or network scans [54], [71]. The success of attack detection using MFD features depends on the size of the window used to aggregate flows, as attackers can choose to spread their activities over the time to elude detection. Consequently, the use of different window sizes should be considered [39].

On the downside, when MFD features are used as the sole data source for the detector, it is difficult to trace the source of the attacks, due to the absence of packet or flow identification fields [68]. Furthermore, calculating these features has an additional cost in terms of processing, memory resources and time.

B. Class Labeling: Derivation of Response Variables

Response variables, or labels, are defined to explain the actual nature of a data record: non-malicious or malicious. In a fine-grained scenario, labels may even represent the specific attack type. These variables are usually required for evaluation purposes, to compare the output of a detector with the actual label. Also, the availability of labeled records is mandatory for training supervised data mining approaches such as classifiers (see Sections VII-A and VIII).

As labels are vital to properly construct and evaluate NIDS, the labeling process must be approached carefully to obtain a dataset without false positives or negatives. False negatives correspond to attack records that are labeled as normal traffic because they are not properly identified by the annotator. On the contrary, false positives refer to normal traffic labeled erroneously as malicious. Such mislabeled data leads to detectors raising alarms for normal traffic while overlooking attacks.

There is no single way to add labels to dataset records. Some methods require human intervention, while others are automatic. Manual labeling is done by analyzing each record in the dataset, along with its context, and deciding whether a record is malicious or not. Ideally, this task should be carried out by expert annotators, because the identification of malicious patterns requires domain-specific knowledge, necessary to provide highly reliable labels. However, human experts are a limited and valuable resource that is not always available given the great effort required by this task. For this reason, manual labeling is often considered impractical [74].

Active learning techniques aim to reduce the effort required to manually label records. To do so, only the most informative samples, those providing non-redundant information, are taken as candidates to be manually labeled [75]. Relevancy of samples is computed leveraging on the output of a supervised model trained using that already labeled data, e.g., based on class probabilities, distances, outcome of voting committees, etc. Active learning has proven to be useful to produce labeled datasets that may be of small size but result in detectors that perform better than others trained with larger datasets in which redundancy has not been removed. However, for evaluation purposes, the number of labeled records obtained by means of active learning may be insufficient.

To fully automate the labeling process, external tools can also be used. A NIDS detector specialized in matching signatures for known attacks (misuse detection) is used for labeling in [76], together with public databases of attacks and public lists of blacklisted hosts. The objective is to minimize the number of false negatives to which misuse detectors are prone [77]. Unsupervised labeling by means of anomaly detectors is another alternative used in [78], [79]. They focus on identifying deviations from normal (majority) traffic statistics, but detected events may contain false positives [4]. In [78] a clustering algorithm is used to differentiate normal vs. attack records: samples in small or sparse clusters are labeled as attacks, and the remaining ones as normal. Proposal [79] uses several statistical anomaly detectors, whose output is combined to reduce the risk of false positives. Note that in any case, mislabeled records may still remain in the dataset. In order to deal with these errors (false positives and negatives), some NIDS authors differentiate between "normal" and "background" traffic, understanding that background traffic may contain intrusions not detected by a misuse detector [76]. Other works associate different certainty levels to records labeled as (possible) attacks by an ensemble of unsupervised anomaly detectors [79].

Yet another way of generating labeled datasets is through a synthetic environment in which well identified network attacks are injected into a network at specific, and well controlled, times [74]. The injection schedule is used to do the labeling, by matching attack times with the time stamps of network records. Scheduled labeling ensures the identification of all malicious records while keeping the labeling cost under control. A disadvantage of these process is the lack of realism in the scenario, as both the normal and the malicious traffic is artificially generated.

Note that labels may be (1) unbalanced, because in a network there is more harmless traffic than attacks, and not all attacks are equally common; and (2) unrealistic when, in order to make it more balanced, the proportion of attacks is artificially high [80]. Both alternatives have a bearing on the performance of data mining algorithms [81]. Several authors modify the datasets in order to have a "fair" label representation for training their NIDS models [82]–[87].

C. Publicly Available Datasets for NIDS Research

From the previous sections it can be seen that collecting and preprocessing realistic data for NIDS research and development are complex tasks, particularly in supervised

TABLE III
A SUMMARY OF DATASETS AVAILABLE FOR THE EVALUATION OF INTRUSION DETECTION SYSTEMS

Dataset	Raw captures	Payload features	SFD	MFD	Labeling method[1]	Duration	Year of Captures	Non-intermittent capture	Real scenario	#Articles
DARPA 1998 [88]	✓				MS	9 weeks	1998	YES	YES	6
DARPA 1999 [89]	✓				MS	5 weeks	1999	YES	YES	5
KDD-CUP 99[2] [39]		✓	✓	✓	MS	9 weeks [1]	1998	YES	YES	36
NSL-KDD[2] [90]		✓	✓	✓	MS	9 weeks [1]	1998	YES	YES	13
Kyoto2006+ [91]			✓		A	34 months	2006-2009	YES	YES	3
MAWILab [79]	✓				A	15min daily traces	2001-	NO	YES	3
ISCX-2012 [74]	✓	✓	✓		S	1 week	2010	YES	NO	3
TUIDS [92]	✓		✓	✓	S	1 week	2011	YES	NO	3
UNSW-NB15 [58]	✓	✓	✓	✓	A	31 hours	2015	NO	NO	2
UGR16 [76]			✓		AS	19 weeks (132days)	2016	NO	YES	-
NGIDS-DS [93]	✓				S	5 days	2016	YES	NO	-

[1]M=Manual labeling, A=Automatic labeling, S=Scheduled attacks
[2]Derivatives of DARPA datasets

scenarios where a ground truth (labeled records) is needed. Many NIDS researchers resort to publicly available datasets that help reduce this effort and also provide a common yardstick which is useful to compare against other proposals. However, their use presents some limitations and drawbacks, which we explore in the following paragraphs.

Table III shows a summary of popular NIDS datasets, the features available in each one, as well as the labeling method used by their creators. It can be seen that most datasets provide already preprocessed records, and only a few of them provide the raw captures from which higher-level (SFD, MFD) features were derived. This is a clear drawback and, ideally, all datasets should include the raw captures, allowing researchers to obtain those header, payload, SFD and MFD features that they find more adequate –instead of being limited by those already included in preprocessed data.

A deeper investigation about the generation process shows that some datasets [76], [79], [88]–[91] are captured in real networks during the operation of real users. Most of these datasets, due to privacy restrictions, need to be anonymized without affecting the original characteristics of the data, which is an important challenge on its own [94]. Oftentimes, network managers are not willing to collaborate in dataset creation as there is a risk of exposing vulnerabilities in the networks they are operating (and, therefore, in their organizations). For this reason, synthetic network environments are the main alternative to obtain captures from network traffic.

Synthetically generated datasets [58], [74], [92], [93] contain traffic captured in synthetic environments, designed with the aim of being as realistic as possible, but with limited infrastructure; hence, these datasets are characterized by including a small number of hosts (sources/destinations of traffic) and limited traffic heterogeneity, which may be easy to model –resulting in biased and scenario-dependent detectors. Moreover, traffic generation methods include programs that try to mimic the operation of the real users [58], [74], [92], [93]. In general, the adequacy of synthetic datasets is in question as they do not truly reflect the operation of actual networks in terms of size and variability of user behavior (legitimate or malicious) [76].

Another issue of these datasets is the short duration of captures, commonly lasting only a few weeks or even days (see Table III). Networks are changing environments where traffic patterns, harmless and anomalous, change throughout time and rarely manifest long stationary periods [4], [16]. Short duration captures over intermittent periods limit the correct evaluation of the adaptability of the models when the network changes. Some commonly used datasets were created more than a decade ago and, therefore, contain outdated traffic, meaning that old protocols and services are present, while current ones are not. Also, ciphered traffic is omitted. In summary, the included traffic patterns do not represent current networks. At any rate, being recent does not guarantee being representative of current networks, especially if the dataset is synthetic [76].

The methods used to label records are also an important factor to take into account when choosing a public dataset (see Table III and Section IV-B). Manual labeling is the most accurate method [74]. Synthetically generated datasets provide good quality gold standards due to the way in which attacks are inserted but, as we mentioned, they may not reflect realistic patterns. Labeling traffic captured from real data is problematic, as normally it is performed using automated tools that do not guarantee the lack of false positives and false negatives. These errors may have a significant effect on the effectiveness of a NIDS proposal [4].

The most used datasets in the literature (see Table III) are the DARPA datasets and their derivatives (KDD-CUP 99 and NSL-KDD). Such databases have been largely criticized along the years due to statistical issues [90], [95]–[97]. Despite this, and that better alternatives have become available in the last few years, the DARPA datasets and their derivatives are still used to develop and test intrusion detectors [98] while more recent datasets have not received enough attention by NIDS authors.

In summary, using publicly available datasets, as most NIDS researchers do, may be convenient, but they are not free of issues of which researchers must be aware. These datasets are not perfect, given the difficulties to emulate a realistic networking scenario or to share data from a real corporate network. They can be very useful in the initial phases of

NIDS research (thus, the efforts of dataset authors must be appraised), but more work needs to be done in dataset creation and evaluation. The traffic that makes up the datasets must be as realistic as possible and associated labels should be of good quality, to give validity to the evaluations of NIDS proposals.

V. Preprocessing Stage: Data Cleaning and Feature Transformation

The main characteristic of the feature vectors (records, samples) obtained from the previous stage is their heterogeneity. They include discrete (categorical and numerical) and continuous (numerical) features. As some data mining methods cannot deal with this heterogeneity, it is common to transform the features to obtain homogeneous records (all numerical or all categorical). Also, data mining algorithms are very sensitive to noise in the data caused either by outliers or errors during the data capturing process. The procedures used during this second preprocessing stage involve the derivation of new features, the modification of existing ones and the elimination of noisy samples, all in order to build a "clean" dataset [23].

Unfortunately, little attention has been paid to this stage by NIDS authors. Most papers do not provide enough details about the exact cleaning and transformation procedures they perform, making their analysis difficult and hindering the reproducibility [59], [99]–[107].

A. Noise Reduction

Network data may be susceptible to errors during processing. Such errors, commonly denominated noise, are reflected in different forms (missing values, outliers...). Noisy data affects the learning capacity of data mining detectors negatively, making them prone to failures [108]. Thus, noise reduction procedures aim to increase the effectiveness and accuracy of data mining algorithms. Noise can be reduced by deleting those samples showing higher distortions with respect to the majority of elements in the dataset, either in a single feature with extreme or missing values, or in several features. Note that, as we describe in the remainder of this section, other transformation methods also contribute to the reduction of noise in an indirect manner.

Only a few NIDS papers explicitly describe the noise reduction techniques they use. Examples are [109] and [55], where outlier removal is done before feeding the dataset to their respective detectors. The former method uses the Mahalanobis distance to rank samples and remove the 0.5% with the largest values, whereas, the latter technique uses an interval for every feature. To do so, limits away from the mean value are calculated taking 10% of the distance between the maximum and minimum feature values.

B. Transformation From Categorical to Numerical

There are different approaches to convert a symbolic feature into a numerical representation. A naive conversion is called *number encoding*. It keeps the dimensionality of the dataset, replacing with a number each possible categorical value. This method has been used to simplify a probabilistic classifier for NIDS in [110] and to cope with the incompatibility of categorical data of the model in [111].

One-hot encoding adds a new binary feature (dummy variable) for each categorical value in the variable, and assigns a positive value (1) to those observations sharing the category represented by the binary feature, and a zero value (0) otherwise. This approach has been commonly used in NIDS proposals to enable the use of distance calculations [83], [112]–[114] without the need to use a heterogeneous measure. Other NIDS, whose models cannot work with categorical values, also use this transformation [72], [115]–[119].

A variant of the previous method is called *frequency encoding*. Each feature value receives a weight according to the frequency of appearance of that value in the data. This method was firstly proposed for NIDS in [84], because it is useful to work with distances. Based on this first approach, frequency encoding was also used later in [86].

In summary, transforming features is a common task in NIDS to deal with the heterogeneity of network data (IP addresses, service types, protocol flags, timings...). Number encoding maintains feature dimensionality, and is useful to simplify value comparison and to reduce memory utilization. However, while distance or order relationships are inherent to numerical data, this is not necessarily true for categories encoded numerically. Therefore, this solution may lead to erroneous conclusions when using certain data mining algorithms, e.g., those using distances. One-hot and frequency encodings work well with data mining methods that make use of the arithmetical properties of features, at the cost of a significant increment of data dimensionality [120]. For example, in the context of a large corporate network, where features with the source and destination of connections may take thousands of different IP address (categorical) values, this transformation results in an extremely wide vector of binary/frequency (numerical) features. Such sparse data slows down and damages the performance of data mining algorithms [121]. Another drawback, common to one-hot and frequency encoding methods, is that they assume that all the values of the categorical variable are known in advance, something that may not be valid in changing environments such as networks (communications may reveal new IP addresses). In this situation, an encoding in use may become invalid.

C. Feature Scaling

Feature scaling, also called feature normalization, is used to limit the range of values of a numerical variable. It has been demonstrated that scaling procedures improve the convergence time of optimization algorithms, as they narrow the solution search space without negatively affecting the accuracy [122], [123]. Applying scaling methods may also reduce the skewness in the data as well as the bias caused to some data mining algorithms [109], [124].

The use of feature scaling has been documented in several NIDS articles. In [68], mean normalization is carried out to scale numerical values before the application of principal component analysis (PCA, see Section VI-A3). The method uses the mean and the difference between maximum and

minimum values to center the data. A logarithmic transformation (replacing a value with its logarithm) is used in [101], [116] to reduce the variability of features with wide value ranges. Min-max normalization uses the minimum and maximum values of a variable to adjust its range to values between 0 and 1; this method is applied in [111], [125] and [115], [116], [119] to reduce the training time of the models used for detection. The same approach is used in [55], [85], [117] to avoid the bias caused by features with wide value ranges in distance-based algorithms. Standardization takes the sample mean and the standard deviation to transform a feature to have zero mean and unit variance; it is used in [83], [84], [113], [114], [126] to improve, again, the behavior of distance-based algorithms. Both standardization and min-max normalization methods are used in [127] to improve the performance of an optimization algorithm that estimates the parameters of their detection model.

Most data mining algorithms, excluding tree-based algorithms, are sensitive to the presence of features with different scales in the data. When data is unequally scaled, algorithms give more weight to variables with higher values, resulting in biased models [128]. Accordingly, the use of methods to re-scale features is often mandatory. The suitability of a scaling method depends on the characteristics of the data mining algorithm and the data itself. Standarization requires knowing the mean and variance of a feature beforehand, and it is recommended for algorithms that assume that data is normally distributed [128]. If this assumption is not made, other scaling procedures may be more appropriate. Min-max and mean normalization re-arrange feature values into smaller intervals; again, a priori knowledge of feature ranges is required. Additionally, for features with most values within a very narrow range, feature scaling methods result in a reduction of the differences between (re-scaled) values when outliers are present, disturbing the operation of some data mining algorithms [117]. Logarithmic scaling skews the data towards large outliers, while standardization, mean and min-max normalization lessen the significance of the inliers [120]. Noise removal, as well as the discretization methods that will be explored in the following section, may help in reducing these side-effects.

The need for previous knowledge about the value ranges of features is a major obstacle in dynamic scenarios such as streaming contexts, as this information may be unknown. Methods to re-estimate feature statistics as new samples arrive may be necessary. Logarithmic scaling is also useful in this context as it does not require any parameters.

D. Feature Discretization

Discretization is used to transform a continuous feature into a discrete one [129]. Although the use of discretization techniques is largely discussed in the machine learning literature [15], it is not that pervasive in the NIDS literature. Discretization works by selecting split or cut points over the continuous space of a variable to create subdivisions (intervals) of this space. Then, each continuous value is replaced by the corresponding interval identifier.

Variables are discretized with the purpose of increasing the learning speed of some detection algorithms, such as those based on induction rules [130]. In decision trees, the use of discretization improves the effectiveness of the learning procedure, resulting in smaller and more accurate trees [131]. Discretization also reduces the complexity of algorithms based on the Bayes theory (by using summation instead of integration terms), while increasing the accuracy of resulting classifiers [129]. Furthermore, it is useful to remove noise from data and to reduce the side-effects of outliers [117], at the cost of some information loss.

The simplest discretization methods are Equal-Width (EWD) and Equal-Frequency discretization (EFD) [15]. EWD splits the original feature range into k intervals of equal width; EFD instead creates as many intervals as needed, in such a way that each interval contains the same number of n samples. Consequently, higher values of n will result in a smaller number k of intervals. A similar method uses K-means clustering, which breaks the range of features into k intervals and replaces the original value of a feature with the centroid of its corresponding range [117]. In all cases, the user must select the parameter that affects the number k of intervals.

Other popular, more complex discretization methods used for NIDS that set the number of intervals automatically are Proportional k-interval discretization (PKID) and Entropy Minimization Discretization (EMD). PKID selects a number k of intervals in such a way that each interval contains a number of samples that is equal to k, trying to find a trade-off between the choice of k and the sparseness of the data [132]. Feature intervals are selected as a function of the conditional probability distribution of the data. As a result of PKID, data is distributed proportionally among the bins, and the resulting discretization is adequate for naive Bayes classifiers [133]. EMD is based on the Minimum Description Length (MDL) principle, which seeks the minimum number of intervals with maximum information [134]. It recursively selects as interval cut points those elements with minimum values of information entropy with respect to the class, i.e., less representative areas of the continuous space of the feature. The intervals created are dominated by a single class [132]. ML algorithms based on decision trees include embedded discretization methods similar to EMD: a continuous variable space is split by selecting the cut point that yields the maximum information gain from the split [131].

Another discretization method, suitable for very dynamic scenarios such as networks, is proposed for NIDS in [117]. In such scenarios, feature ranges may be unknown in advance and applying most of the previous methods is impractical. For dealing with this issue at training time, each dynamic-range feature is clustered into k groups applying K-means. After that, k Gaussian distributions are obtained (one for each group). Next, as new samples arrive, each dynamic range feature is replaced by k new features, obtained using the Gaussian distributions and containing the probability of the original feature value pertaining to each group.

The benefits of using discretization methods for NIDS have been documented in many studies [135], [136]. The main shortcoming of simple methods is the need to choose the

number k of intervals (or the per-interval frequency n for EFD). This is a non-trivial process, as a small k entails the removal of useful information and a large k results in the inclusion of noise [120]. Methods to automatically select the number of intervals based on the class information, such as PKID and EMD can be used, although the computational cost will increase. A common drawback of most discretization approaches is that they do not work properly when processing data on-the-fly, because neither feature ranges nor labels (for those methods requiring them, such as EMD and PKID) are known a priori. Previously unseen values would invalidate the parameters of the methods. These issues may be tackled using self-adaptive, online discretization methods instead [137], [138], but no reference to them has been found in the reviewed NIDS literature.

VI. Data Reduction

In the previous sections we have discussed the derivation of an exhaustive set of features, which later will be used to build the detector. This dataset may be huge in terms of the number of samples, and also in terms of the number of features per sample. In this context, it has been shown that using the full feature set extracted from traffic is ineffective for intrusion detection, due to the inclusion of irrelevant variables that contain redundancies and hinder the learning capacity of data mining algorithms [139].

In this section we discuss methods to reduce the dimensionality of the dataset with the aim of: (1) reducing the computational cost and (2) improving the accuracy of intrusion detectors. We describe some techniques that reduce the number of variables in the feature set, as well as others that reduce the number of samples (records) in the dataset.

A. Feature Dimensionality Reduction

Feature dimensionality reduction procedures try to cut down the number of features considered by the NIDS by discarding useless information or reducing redundancy. There are two main alternatives to do this: removing explanatory variables, or projecting the original explanatory variables onto a lower dimensional space.

1) Manual Feature Removal: The simplest feature reduction method is the manual removal of "irrelevant" features. It requires a profound expert knowledge of the domain, as those features are excluded from the next KDD phases [102].

As previously mentioned, many data mining algorithms cannot work with some types of attributes (categorical, numerical). To deal with this issue, many authors simply opt to remove features instead of applying preprocessing methods (see Section V). For example, in [109], [126], [140]–[143] all categorical variables are removed because their data mining detectors are unable to deal with non-numerical data. Note that this removal may result in important information loss [140].

Manual exclusion of features based on other criteria has also been described in some NIDS papers. In [70], [71] only MFD statistics of the dataset are used for DoS detection, discarding other features. In other proposals, constant features are removed as they do not provide useful information [57], [86]. In [77] features such as IP addresses of attacking machines and attacking times are removed, as they reflect information clearly related with the simulated environment where the dataset was obtained from.

2) Feature Subset Selection (FSS): These techniques select a subset of the whole feature set that is assumed to be the most relevant to solve the problem at hand. FSS methods reduce the cost of data mining models, help to prevent model over-fitting, and enable a better understanding of the data because redundancies are removed [144]. FSS methods can be classified into three main categories: wrapper, filter and embedded methods.

Wrapper FSS approaches measure the benefit that the inclusion of a given feature (or set of features) has over the performance of a predictive model. This can be seen as an optimization procedure that looks for the feature subset that maximizes the predictive capacity of the model. As such, and in order to measure the quality of the predictive model, they require the data to be labeled [120]. Different methods to select feature subsets have been described for NIDS, including sequential greedy methods [46] that add the feature that improves the model accuracy in a greater degree in each step until there is no a remarkable improvement; and methods based on meta-heuristics that guide the search of a solution in the space of subsets of variables. Examples of meta-heuristic FSS algorithms used in NIDS include Genetic Algorithms [77], [145], Firefly Algorithms [146] and Simulated Annealing [147].

In contrast to wrapper methods, results of *filter FSS* methods do not depend on the predictive model used in the next KDD phase. The relevance of features is computed through an evaluation of their characteristics using correlation, mutual information, consistency, variance or similar metrics [120]. Normally, filter FSS methods are computationally cheaper than wrapper methods, and more appropriate for datasets containing unlabeled data or a large number of features [144].

Correlation-based Feature Selection (CFS) methods [148] choose those features that are highly correlated with the response variable (the label) while showing a low correlation with the remaining features. To compute such correlation, Pearson's correlation coefficient is commonly used, although this coefficient is only able to express linear relations [120]. Other correlation-based filter methods use Mutual Information (MI) (based on entropy), which is able to measure complex relations [149], but can be applied to discrete features only. CFS methods based on Mutual Information have been used for NIDS by selecting the features that maximize feature-class mutual information [101] and include features that maximize feature-class mutual information and minimize the redundancy with respect to other features [111], [125].

Consistency-based methods evaluate the constancy of feature values with respect to class labels [150]. As such, features with stable values in samples belonging to the same class are selected. For NIDS this method has been used in [135], [136] in conjunction with two CFS methods, one based on Pearson's correlation coefficient and the other one based on MI [151].

Analysis of Variance (ANOVA) tests [152] decompose the variance of features among linear components, which can be used to compute the variance of a feature with respect to

others. Then, redundant features containing similar characteristics to others can be discarded. For NIDS, this approach has been used to select those features that provide non-redundant information in [72].

Finally, *embedded FSS* approaches are either filter or wrapper methods integrated into prediction algorithms as part of the training phase. They are dependent on the particular data mining algorithm used, usually trees, as each training step comprises both ranking relevant explanatory variables and generating a model using the top-ranked ones [144]. For example, C4.5 [153] and C5.0 [131], [154] decision tree builders are used in [77], [85], [155] and in [70], respectively, to generate intrusion detectors. Both algorithms have a built-in filter FSS method that weights features using their Gain Ratio (an improvement over MI used to avoid bias towards variables with higher cardinality). Other authors use Random Forests for intrusion detection [82], [156], [157], which includes a wrapper FSS as part of the learning process. The feature relevance metric is based on the misclassification rate produced by the permutation of the values of the input variables [158].

3) Feature Projection: Feature projection techniques do not select a subset of the original feature set. Instead, they are used to obtain a projection of the features into a lower dimensionality space. It is a form of compression, based on the use of statistical and mathematical functions that find linear and non-linear variable combinations that retain most of the information of the original feature vectors [128].

Principal Component Analysis (PCA) [121] is one of the most common dimensionality reduction procedures. PCA applies a linear transformation to the original explanatory variables to map them into a space of linearly uncorrelated components. Dimensionality reduction is performed by selecting a number of components smaller than the original number of features [159]. Ideally, data provided to PCA must be scaled and free of outliers to avoid giving more weight to features with wider value ranges [109]. The main shortcoming of PCA is its inability to capture complex, non-linear relations between features. Several NIDS works use PCA to reduce the feature dimensionality of their data before training their models. In [160] several training and testing executions are performed using PCA in order to select the number of components that result in the highest performance. Proposal [113] selects those components that best discriminate between normality and attack, as indicated by Fisher's Discriminant Ratio.

In contrast to PCA, where the extracted components are orthogonal and ordered by the variance they retain, Independent Component Analysis (ICA) separates a multivariate signal into a set of statistically independent non-Gaussian components or factors of equal importance [161]. Another difference with respect to PCA is that ICA is able to capture non-linear correlations [121]. In NIDS, this method is used in [60] to obtain a representation of the data in which only those independent components that better characterize the malicious class are kept.

Auto-encoders, a type of Neural Networks (NN), have also been proposed for feature projection. They try to reproduce the input values (the original features) at the output layer. Dimensionality reduction is achieved by projecting the input onto a smaller space in the intermediate layers. Auto-encoders can model complex relations in the input data, not limited to linear relations [162]. Different auto-encoder methodologies have been proposed in the NIDS literature, including symmetric [163] and non-symmetric [107] NN architectures, sparse auto-encoders[5] [59], and auto-encoders based on self-taught learning techniques [164]. Four different auto-encoder models are used in [118] to determine the best NN-based projection method for NIDS.

Finally, different feature dimensionality reduction methods can be combined, either by applying projection methods to a subset of the original features obtained by means of FSS, or viceversa. For NIDS, PCA is applied to obtain a reduced projection of the features selected by a filter FSS method based on information gain in [160]. This combination is carried out in order to get a very compact representation of the data at the cost of an increased complexity.

The two main challenges of feature dimensionality reduction methods are: (1) the choice of an adequate number of features, small enough to remove redundancies (and speed up the execution of data mining methods) but without harming the detection performance; and (2) dealing with *concept drift*, that is, adapting to unforeseen changes in the characteristics of input data, something common in networking scenarios [165]. Auto-encoders and FSS wrapper approaches require several costly training and evaluation steps with different dimensions (feature subsets in case of FSS) to find the most adequate size of the feature set [120], [162]. This does not happen with other projection methods (PCA, ICA) and most filter FSS approaches, as the inclusion or exclusion of features is decided by setting a threshold over their quality. The difficulty here is in tuning the threshold to determine an appropriate number of dimensions without discarding valuable information [121]. The availability of labeled data is also important when selecting a method to reduce feature dimensionality. For wrapper methods labels are compulsory. For projection and most filter FSS methods, the set of selected features depends only on the values of the features themselves, and therefore labels are unnecessary [120]. Data projection methods are widely used as they uncover implicit relations between features and may reach better reduction ratios than other methods [120]. However, because projections do not contain the original values extracted from traffic samples, their main shortcoming lies in the lack of explainability of detected attacks.

B. Sample Dimensionality Reduction

The cost of training most data mining algorithms depends heavily on the number of samples used to train the models. This may be a critical issue in scenarios where learning from data is applied under strict time constraints [43]. Techniques that reduce sample dimensionality are sometimes applied with the aim of accelerating the learning step of those algorithms, either by removing duplicate or too similar samples, or by

[5]These projection methods do not result in a reduced projection, instead they produce a sparse representation where the projected data is larger than the original feature set.

selecting representative samples that are used instead of the whole dataset.[6]

In the context of NIDS, proposal [112] uses a tree-based clustering method to reduce the number of samples used to train the model. Instead of training with all the instances, the cluster centroids are used as representatives, thus achieving a significant reduction in the sample dimensionality. A similar method is used in [140], [166] to reduce the cost of performing the distance calculations required by their ML detectors.

VII. Data Mining Stage

Data mining is the core stage of the KDD process, where the preprocessed and reduced data is fed into different algorithms whose objective is to identify patterns in the data and, ultimately, detect attacks. Due to its core role, most NIDS surveys (such as [9], [11]) focus almost exclusively on this stage. In fact, the authors of those surveys categorize NIDS proposals using the type of data mining algorithm used as the main criterion. In this survey we discuss this stage using a slightly different viewpoint, and present a novel taxonomy of NIDS methods based on two criteria: the detection goal and the learning approach.

The first criterion categorizes NIDS methods into three types (misuse-based, anomaly-based and hybrid systems) based on the meaning they assign to the concept of attack and the goal of the detector. The second criterion classifies NIDS proposals according to how they train or build the detector: in batch mode or incrementally.

A. Misuse-Based, Anomaly-Based and Hybrid Detectors

This nomenclature (misuse, anomaly) has already been used in previous surveys such as [11], [13] and, even if the definitions of these terms given in the literature are not always identical, the different versions do have some common grounds. In general, most authors have defined misuse-based systems as NIDS designed to detect well-known attacks using either a knowledge-base of attacks [7], rules written by domain experts [8], attack-signatures [6], [11], [13], [167] or attack-patterns [16], [39]. The authors do not specify the exact meaning of *signature*, *rule*, *pattern* or *knowledge-base*, but clearly some definitions refer to very rigid systems (signature, rules, expert knowledge), while others describe misuse-based NIDS as more flexible systems (patterns). Moreover, in any of these works, it is not clearly stated how this kind of identifying information can be extracted from data.

In the case of anomaly-based detectors, the definitions are vaguer and, oftentimes, with slight differences. These NIDS have been defined as systems whose objective is to detect *exceptional* patterns [9], patterns that *deviate from other observations* [10], patterns that *deviate from the normal* [6]–[9], [11], [13], [39] or patterns that *deviate from the expected* traffic [4], and also as systems *with knowledge of normal behavior* [167]. In this line, in many of the mentioned surveys, their authors state that an anomaly-based NIDS attempts to obtain profiles of the normal traffic, in order to compare the new incoming data with those profiles, identifying observations that deviate significantly from normality.

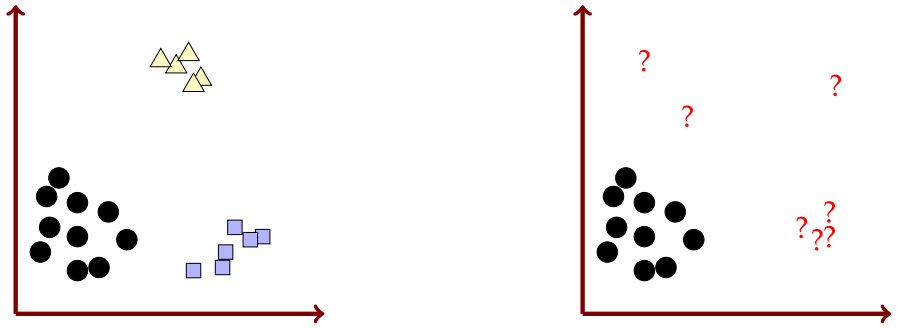
Often, different surveys classify the same NIDS differently, as evidence of the absence of a clear-cut categorization. For example, the classifications made for [45] and [82] in two widely referenced NIDS surveys are incompatible. In [9], Bhuyan *et al.* define the work by Ariu *et al.* [45] as anomaly-based, and the work by Zhang *et al.* [82] is categorized as a hybrid approach. However, it their review, Buczak and Guven [11] state that both proposals are misuse-based methods. In recent surveys, the inconsistency of these definitions has lead to the miscategorization of NIDS that use some data mining methods, such as supervised classifiers [9], [13], [14], identifying them as anomaly-based approaches although they present limitations to detect new and unknown classes of traffic [16], [168].

To avoid further misclassifications, we start laying the foundations for these definitions, providing more specific and up-to-date descriptions of misuse and anomaly based approaches. We focus on three aspects that depart from ideas and concepts of the data mining area, particularly from the area of anomaly/outlier/rare event detection [169]–[171] (see Table IV). Any combination of the characteristics shown in this table will result in a hybrid detector.

- **Misuse-based detectors** understand attacks and normality as categories/classes within the data (see Figure 4(a)). They depart from the assumption that an attack is a minority category with specific patterns/characteristics that can be learned using the training data. Misuse-based systems require fully labeled data (all attacks must be correctly identified) at training time. Once a misuse-based system is built, the response obtained given a new observation is its correspondent or potential class according to the learned characteristics. Therefore, detection ability is limited to the set of attacks and/or normality classes modeled.
- **Anomaly-based detectors** assume that attacks can form categories with specific characteristics, but they can also be isolated observations (outliers) (see Figure 4(b)). With this in mind, these detectors regard that an attack is anything deviating from normal traffic and, therefore, they focus on modeling normality. Anomaly-based NIDS do not require attacks to be explicitly identified in the training data, and the output they produce during operation is vaguer than in misuse systems, since the returned information only states that an observation deviates sufficiently from normal traffic.

This categorization of NIDS is closely related to the degree of supervision required by the data mining algorithm applied. Classical supervised classification algorithms depart from a training dataset in which the instances are labeled as attack (or type of attack) or normal traffic. Their goal is to learn a predictive model from this data, which is then used to identify attacks in new incoming traffic [11]. As such, they are typically associated with misuse-based systems. On the contrary, unsupervised learning algorithms depart from a dataset of instances that are not labeled at all. Their goal is to identify

[6]These techniques are not to be confused with class balancing, whose aim is to reach a proportional, and maybe unrealistic, distribution of samples between the classes.



(a) Misuse Detection problem (b) Anomaly Detection problem

Fig. 4. Visual description of the misuse and anomaly detection problems. In misuse systems, both the normal data (black circles) and the attacks (yellow triangles and purple squares) have specific characteristics and, thus, form classes. In anomaly detection scenarios, normal observations (black circles) are assumed to follow characteristic patterns that constitute well-defined classes, while abnormal, maybe malicious, patterns (red "?" symbols) may or may not have enough entity to form a class.

TABLE IV
DIFFERENCES BETWEEN MISUSE AND ANOMALY DETECTION FROM A DATA MINING PERSPECTIVE

	Do the attacks form classes?	**What is modeled?**	**What is the response obtained from the NIDS?**
Misuse	Yes	Attacks and Normality	Normal observation vs. attack (or type of attack)
Anomaly	Not necessarily	Mainly normality	Normal observation vs. Abnormal observation

the natural groups that underlie in the data [172] and, thus, are typically more related to anomaly-based systems.

In anomaly-based NIDS we can also find different types of weak supervision [173], where some kind of uncertainty is included in the labeling. To the best of our knowledge, two types of weakly supervised learning approaches have been applied to NIDS: semi-supervised learning and one-class classification. In semi-supervised learning [174], the data is partially labeled. Commonly, in NIDS, these methods learn in two steps. In a first step labels are ignored and data is clustered using unsupervised techniques. Afterwards, in the second step, all the instances of each cluster are labeled identically by leveraging the available labels, and assigning, for example, the label that appears most. In one-class classification [175] instead, only some of the instances of one of the classes in the data are labeled. This serves to accurately differentiate between the elements of the labeled class with respect to those from other classes. In general, weakly supervised algorithms do not require as many labeled instances (which are difficult and costly to obtain, see Section IV-B) as fully supervised approaches: one-class approaches only require the maximum number possible of labeled instances from the class of preference, i.e., the one which we are interested to learn from (for anomaly-based NIDS, the normal traffic); while semi-supervised methods work well with few labeled instances of each of the classes in the data [173]. Also, weakly supervised methods have better accuracy than totally unsupervised methods [173].

In the following subsections we describe NIDS works from each of these categories, commenting on the advantages and disadvantages of each approach. We also discuss the degree of supervision required, or typically assumed, by the different detection methods.

1) Misuse-Based NIDS: As mentioned above, misuse-based systems normally leverage classic supervised classification methods. These algorithms have evolved and improved substantially in the past few years and, consequently, they are able to yield very reliable predictions when fed with enough, good quality data. The main advantage of misuse-based systems is that they are able to accurately model the patterns and characteristics of well-known attacks present in the training set, obtaining low rates of false alarms (false positives). However, most of these techniques have important limitations when dealing with new (zero-day) or mutated attack patterns [77], because they are not designed to identify new classes. Thus, they require continuous updates to extract new attack patterns, making them hard to maintain. Also, remember that this type of algorithms require a labeled training set, which is costly and difficult to obtain in real scenarios [120] (see Section IV-B).

The main difference between the various misuse-based systems proposed in the literature is the specific supervised classification algorithm used. Neural Networks (NN) are rather popular and, in NIDS, there are approaches ranging from the simplest network, the Multilayer Perceptron (MLP), [59], [72], [176], to complex Extreme Learning Machines (ELM) [57], Convolutional Neural Networks [119] and Recurrent Neural Networks (RNN) [87], [115], [116].

Other less frequent approaches include Support Vector Machines (SVM), based on finding the maximum margin hyperplane that separates the classes, [46], [72], [102], [111], [125], [147], k-Nearest Neighbor (kNN) approaches, based on distance calculations [84], [106], [114], the Naive Bayes algorithm, based on probabilistic theories [56], [72], [77], [135], [136], [147] and Decision trees, based on constructing tree-like structures that output decision rules [60], [70], [77], [85], [135], [146], [147], [177]. As a more rare approach, a supervised probabilistic variant of the Self-Organizing Maps (SOM) algorithm [178] (which is an unsupervised clustering algorithm), has been applied to NIDS in [105] and [113], assigning to each cluster the most frequent label in its allocated records during the training phase.

The combination of supervised classification methods to form ensemble models has also been proposed for NIDS, since it reduces bias and improves classification accuracy [158], [179] compared to using only one model. The most common ensemble method is the Random Forest [16], [82], [107], [157], which is based on combining many decision trees. However, other heterogeneous ensembles have also been proposed: combinations of Decision Trees and SVMs [155]; or even combinations of SVM, kNN and MLP classifiers [160].

We already mentioned that misuse approaches are only able to recognize what is learnt during training. In this context, in order to provide some flexibility and to avoid overfitting to the training data, some supervised methods include regularization or generalization terms [121]. An alternative approach to provide more flexibility to these methods and attempt to generalize better is proposed in [140]. This misuse-based detector is a variant of kNN, a distance based classifier, that incorporates a genetic approach. This genetic component makes every new sample evolve to perform its classification using a clustered representation of the training data.

We have seen many distinct methods that are used for misuse detection. A comparative analysis of them is not easy due to the lack of a common framework. However, some general conclusions can be extracted through an analysis of the algorithms applied. Among the simplest approaches we can find DT and Bayesian methods, which involve a straightforward learning mechanism. In contrast, NNs and SVMs need a thorough parameter tuning [121]. Most algorithms are costly to train and perform better when they are fed with large datasets. kNN is costly in the detection phase because, to obtain the label for a new sample, it requires the computation of the distance between that sample and the training set [120]. SVMs are good choices for learning from wide datasets (those with few records and many features). Regarding model interpretability, DTs are the most explainable, while the opposite holds true for NNs [121].

2) Anomaly-Based NIDS: These detectors assume that an attack is anything that deviates from normal traffic, and thus, focus mainly on modeling "normality". Deviation of new observations from this normality has been measured in many different ways. As previously mentioned, while misuse-based detectors are commonly based on supervised learning schemes, we can find supervised, weakly supervised and non-supervised approaches for anomaly detection.

A main advantage of anomaly detectors is that they should be able to identify unknown attacks (mutated or zero-day attacks). Labeled data is not a requirement for such systems because normal profiles can be modelled using unlabeled or partially labeled network datasets, which are expected to have a low proportion of attacks [109]. A main drawback of these detectors is that, when deployed in real scenarios, they usually produce a high number of false positives, often related to noise or events that are not attacks (e.g., the connection of a new server in the network). Therefore, anomaly-based NIDS typically require companion methods to explain/categorize the alarms [180].

An anomaly-based NIDS identifies those observations that are very different from the normal majority traffic as attacks. As such, a threshold must be defined to determine if the level of deviation is enough to trigger an alarm. Anomaly-based NIDS proposals differ in the data mining algorithm chosen, as well as in the way deviation is measured. Let us focus first on unsupervised approaches.

The first and most simple unsupervised anomaly-based systems are based on the extraction of frequency profiles from the patterns observed during training. These detectors assume that attacks leave as footprints patterns that do not appear as frequently as normal patterns. For example, in [40], [41], [63], [83], [181], [182] the authors flag infrequent values in traffic features as malicious. More sophisticated is the approach followed in [141]. Traffic of similar characteristics is first clustered, and the frequency of transitions between clusters is analyzed. Attacks are identified when uncommon transition patterns appear, leveraging thus on the temporal relationships among different types of traffic (clusters).

Other approaches are not based on frequencies, but on modeling the correlation structure of the normal data and assuming that attacks follow a different correlation structure. Two main approaches can be identified here: using correlation features directly, or using subspace representations of the correlation features. Examples of the first approach can be seen in [71] and [126]. In the first case the mean of the covariance matrices of normal data is computed. Later, records that deviate from this profile beyond a threshold are flagged as attacks. The second proposal computes the area of the triangle formed by the projection of any pair of variables in the Euclidean 2D space, which is used as an indicator of their correlation. The system generates a normality model by computing the Mahalanobis distance between all normal triangle area records, to obtain the parameters of a univariate Normal distribution. After that, a threshold for admissible normality consisting of n times the standard deviation of the normality profile is set.

Subspace representations of the correlation features by means of projection methods (PCA) have also been used for anomaly detection. For example, in [109] the authors use PCA to extract the major and minor components of normal data. The k major components are those capturing most of the variance. Incoming records with normal correlation structure and extreme values are flagged as anomalous by means of the reconstruction error of their projection using the major components. Similarly, uncorrelated records are detected using the minor components. Other approaches that use PCA to exploit the correlation structure of the data are described in [68] and [5]. Both apply PCA to entropy features, considering the minor components as indicators of how well the normality is represented. In [68], the residual projection of the data obtained by means of the minor components is used to identify anomalies with a high reconstruction error. In all these cases, the number k of major components is selected at training time. In contrast, in [5], the authors use the angle between the PCA projections of two contiguous non-overlaping windows of data to obtain an anomaly score and, to do so, the number of major components k is tuned at run time, selecting the number that minimizes the orthogonality between the normality of both windows.

Time series analysis has also been used for unsupervised anomaly-based NIDS. These statistical methods are used to predict future values for different network features (SFD or MFD), which are compared with actual, measured values. Records with high prediction errors are flagged as anomalies [69], [183]. Time series have also been proposed to model the likelihood of the alarms generated by an unsupervised Long Short Term Memory Neural Network used as anomaly detector [104]. The method filters the number of alarms, indicating that there is an anomaly when this number deviates highly from the expected value.

We end this description of unsupervised anomaly-based systems with clustering methods. They group data that share similar characteristics with the help of similarity or distance measures [184]. The objective is to identify large or dense clusters, which usually correspond with normal data, flagging as suspicious anything that deviates from those clusters. Those methods assume that there is a minority of attack samples, compared to normal traffic. If this were not the case, expert knowledge would be required to interpret the obtained groups. Depending on the clustering method, thresholds can be applied to density measures [54], [55], [83], [100], [118], [142], [143], flagging observations in low density zones as attacks, or to distances [103], [118], [182], flagging farthest observations (or clusters) from the reference normal points as anomalous.

Supervised approaches have also been proposed for anomaly detection in NIDS, although rarely. In [156], a Random Forest classifier is trained with normal data, using the service identifier (TCP, UDP port number) as the class label. In operation, when a new sample arrives, if the service assigned it by the classifier does not match the specific service to which the sample belongs, an alarm is raised.

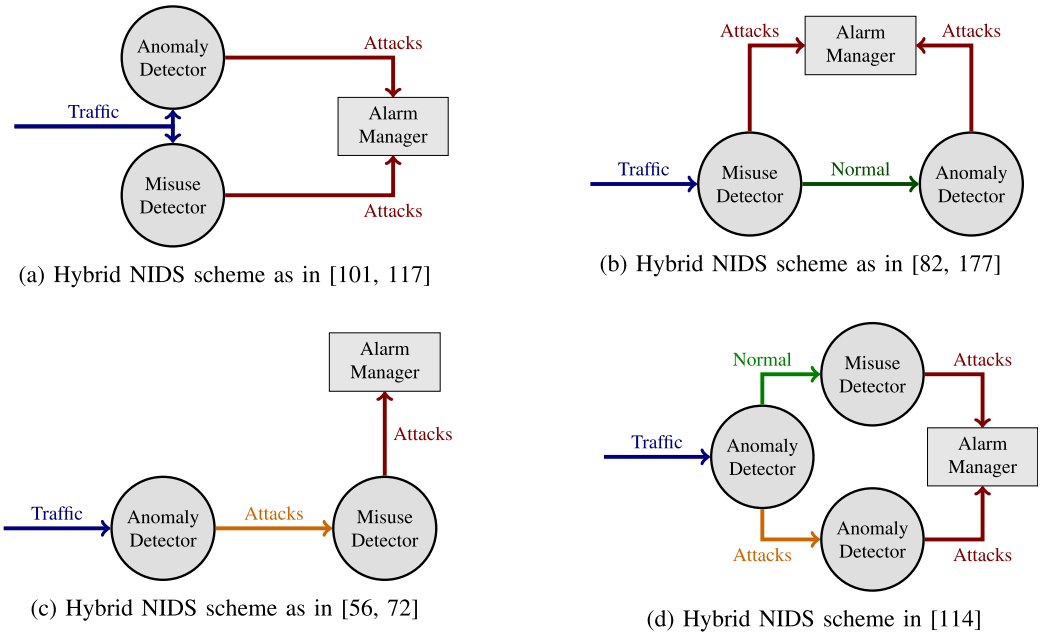
Weakly supervised anomaly detectors for NIDS use partially labeled data, aiming to improve the accuracy of unsupervised methods. Proposals [84], [118] apply a one-class SVM classifier using a dataset in which only some samples are labeled as "normal traffic". In [86], a multi-classifier system formed by various ensembles of one-class SVMs is used to model normal particularities of network services. To do so, each ensemble, formed by two to three models using different feature subsets, is trained for each service. Similarly, proposal [44] also uses an ensemble of one-class SVMs, although it is focused on HTTP attacks. As in the previous approach, each model is trained with a different set of features, and the output of all the models is combined to obtain the final classification result. In [45], an ensemble of five Hidden Markov Models (HMM), a probabilistic classifier, is used in order to predict the probability that a new traffic record is normal. To do that, only payload sequences of normal HTTP traffic are used to train the models. Again, their output is combined to obtain a final score. In [43], a one-class SVDD algorithm that uses active learning strategies is proposed. In that approach, a small portion of very uncertain data is labeled (by an expert) in order to train the detector and increase the detection performance. Proposal [99] uses a semi-supervised variant of SOM [178] in which the class information of the labeled records is used to assign labels to clusters. At training time, when a new sample is fed, it is labeled with the class of its nearest cluster (known as the activated cell). Both that and the remaining clusters that have the same class label are then "rewarded", moving their centroids closer to the sample. Semi-supervision is added to deal with unknown clusters (those that are unlabeled), which are always rewarded. The method is also able to merge and split clusters, respectively merging very pure cells (those with a large number of records of the same class) or creating new clusters from impure cells.

The anomaly-based NIDS methods discussed above present some advantages and limitations which, although they are not enough to perform a comparison among them, they are still worth mentioning. For clustering, the use of feature transformation methods depends on the choice of the distance and the chosen clustering algorithm. Density methods create groups of any shape, while distance based methods form spherical clusters [120]. Correlation methods require numerical features for the computation of covariance or correlation matrices but, as their main shortcoming, they lack explainability when attacks are detected, as those matrices are generated from groups of records. Time series analysis demands using numerical data and existing methods for NIDS are difficult to apply with non-stationary data. However, it is a useful resource to model temporal relations in traffic [120]. Some frequency-based methods are also able to model temporal relations while working with any kind of data, but they may have poor performance in networks with unbalanced types of normality. Finally, weakly supervised methods also require numerical features to work and are not as easily adaptable to incremental approaches as the other methods because they would require constant re-training steps. Instead, they have better accuracy than unsupervised methods because are assisted by labeled data. The cost of obtaining labels is their main shortcoming, but this effect can be reduced by means of active learning techniques [43].

3) Hybrid NIDS: Misuse and anomaly-based methods can be combined into a hybrid detector trying to overcome the limitations of each method separately. At the cost of an increased complexity of the NIDS, this combination can be done in different ways, as depicted in Figure 5.

In Figure 5(a) we can see a simple case in which a misuse-based detector and an anomaly-based detector work in parallel, flagging an observation as malicious if it causes an alarm on any of the two components [117]. The detector described in [101] incorporates both misuse and anomaly approaches in a single data mining method. The system outputs the classification made by the misuse component, and computes the level of deviation from the identified class. Note how this parallel set-up may detect more attacks than a single system, at the cost of an increase in the number of false positives caused by any of the two detectors.

Another way of combining two detectors is concatenating them. The first module of the detector receives traffic records and feeds its output to the second module. Depending on the order in which the misuse and anomaly based detectors are chained, the combination reduces the number of false alarms or increases the detection ability. The architecture proposed in [82], [177] places first the misuse-based system, which receives all network records and identifies



(a) Hybrid NIDS scheme as in [101, 117]

(b) Hybrid NIDS scheme as in [82, 177]

(c) Hybrid NIDS scheme as in [56, 72]

(d) Hybrid NIDS scheme in [114]

Fig. 5. Hybrid-based NIDS schemes.

well-known malicious behaviors (see Figure 5(b)). Afterwards, the anomaly detector receives only those records previously classified as normal in order to discover unusual patterns, increasing the detection ability of the NIDS. The inverse approach has been proposed too [56], [72], as depicted in Figure 5(c), with the intention of reducing false alarms. The anomaly detector flags suspicious traffic, and afterwards the misuse module confirms and explains the flagged anomalies.

The variant depicted in Figure 5(d), and proposed in [114] puts an anomaly detector first. Samples flagged as normal are forwarded to a misuse detector that may identify attacks that passed undetected. Suspicious samples are forwarded to a second, slightly different, anomaly detector, aiming to confirm the attacks and to reduce the number of false alarms. Note that this is the most costly scheme as all input records are analyzed twice.

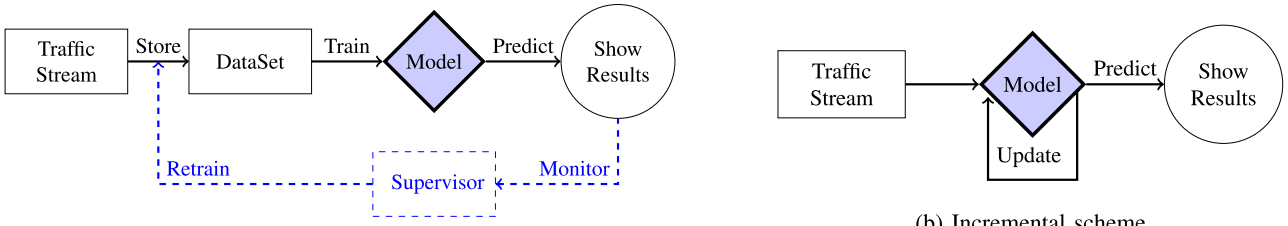
B. Batch vs. Incremental Learning

In our taxonomy, the second NIDS classification criterion refers to the different policies used to learn models and keep them updated. Networks, and network traffic, evolve continuously and, therefore, the characteristics of normal traffic and attacks change over time [4]. In this context, the flexibility and adaptability of the data mining algorithms and their tolerance to previously unseen patterns is crucial [39]. However, this fact is rarely mentioned in the literature. In this section we study the performance and practical usability of two different learning approaches discussed in the NIDS literature: batch and incremental.

1) Batch Learning NIDS: Learning in *batch mode* means that the data mining algorithm is applied once to a set of static (training) data, labeled or unlabeled, and after that, the results or detectors obtained are used to make predictions for new incoming data (see Figure 6(a)). Typically, this type of algorithms require multiple passes over the training data and, since they have not been designed specifically for streaming scenarios, they are usually costly in terms of both time and memory requirements [185]. The effectiveness of these models is high at deployment time, but they do not evolve with time. There is, thus, an implicit assumption of data being generated by a stationary distribution [186]. This is an important drawback, as it seriously challenges the ability to detect previously unseen traffic patterns, that may correspond to new or mutated attacks, or to new kinds of normal traffic [16].

Most classical data mining methods, independently of their degree of supervision, are based on batch learning. Many NIDS proposals leverage off-the-shelf supervised classification or clustering algorithms and, therefore belong to this category. For example, [111] and [160] rely on well-trained models to perform misuse detection. In a similar fashion, [126] and [118] use a batch approach to model normality in their anomaly detectors. None of these proposals are able to update their detectors automatically.

Other proposals try, instead, to somehow capture and adapt to the traffic changes via retraining [187]. This extension of the batch mechanism adds a supervisor component to the detection framework (see dashed elements of the Figure 6(a)) that monitors the status of the network or the response of the system, and decides whether it is necessary to trigger a retraining procedure. Retraining means using the data mining algorithm of choice again, but with a recent batch of stored records, to generate an updated model that replaces the obsolete one [187]. All detectors built using batch learning can be enhanced with retraining. However, this learning mode also has some drawbacks. The selection and storage of the data required to build an updated detector in each retraining phase and, when supervised methods are in use, the labeling of this data (see Section IV-B), are non-trivial tasks that require additional resources [16]. Also, as network traffic changes



(a) Batch scheme (dashed elements are those involved in retraining)

(b) Incremental scheme

Fig. 6. Learning schemes of data mining methods.

at non-predictable times, additional complexity in terms of a monitoring mechanism must be put into the NIDS.

Some examples using retraining have been proposed in NIDS. In [54], basic time series analysis is carried out over statistics of grouped flows in time windows (MFD features) to detect changes in flow patterns. Then, for any time window in which changes are detected, a retraining is done over the flows within the time window, using their SFD features. A density-clustering algorithm is applied to rank records by their distance to the largest cluster. Finally, the records farther than a predefined threshold are flagged as anomalies. In [59] a supervisor is added to the proposed NIDS that monitors network alterations such as changes in open ports (available services) or connected devices. It is assumed that those alterations modify the structure of normal traffic, but attacks do not vary. When a change is detected, an auto-encoder is used to learn a new sparse representation of the data reflecting the new traffic characteristics. The same labeled dataset used for the initial training is transformed with the new auto-encoder to build an updated classifier that replaces the previous one.

2) Incremental Learning NIDS: Incremental learning algorithms assume that (1) data is unbounded and arrives in a continuous manner, and (2) data may be generated by non-stationary processes and, therefore, may evolve over time [186], [188]. Consequently, learning is done continuously with the arrival of new data samples (see Figure 6(b)). The main difference with retraining strategies is that the model or information stored by the algorithm is updated continuously, instead of being rebuilt from scratch. Adaptation to changes is gradual, which better suits highly variable scenarios such as networks, where stationarity is infrequent [21], [187]. In general, the cost is also lower than that for retraining as less data is incorporated to the model. On the downside, since they learn and provide a response in an online fashion, these algorithms can usually only perform a few passes over the data and they have time and memory limitations [187]. Another disadvantage of incremental learning is convergence time: learning about changes requires some time and, meanwhile, accuracy may be poor. This is particularly true at the initial stages of deployment or when abrupt changes occur [189]. Continuous adaptation also makes these systems sensitive to noise and perturbations such as those intentionally generated to evade detectors [190]. As such, incremental approaches require additional and specific evaluation measures that monitor the evolution of the learning process and guarantee its correct behavior before deployment (see Section VIII-A).

As the most common approach, we can find a group of unsupervised methods that analyze if each incoming data point in the stream is anomalous, that is, if it is dissimilar to other data points observed previously in the stream. Since they are incremental algorithms, these methods typically only process each data point once, summarize the stream in a representation with small memory requirements that can be easily updated in an online manner, and compare the incoming points with this representation. A most naive approach within this group is described in [181]. The attribute-value pairs of network records are directly analyzed, flagging data points with infrequent values in certain attributes as anomalies. The stream is thus represented as a table of frequencies of different values for SFD, MFD and payload features, which is updated incrementally with each new sample.

As a slightly more complex approach that follows the same idea, several NIDS papers describe the use of incremental clustering algorithms. These algorithms group the data with similar characteristics and identify attacks as observations that lie far from the existing clusters, or that are located in sparse clusters. Contrary to batch clustering approaches, which group static sets of data into a fixed set of clusters, these methods are designed for streaming environments. As such, the data and cluster structures are represented in different compact manners, i.e., by storing only the objects of the current time window [142], by saving only prototypes or centroids [84], [103], by arranging data into groups of similar data objects (microclusters) represented by a vector of characteristics (mean, variance, etc.) [100], [143], or by using a grid structure in which the data is arranged and summarized using some statistics [55]. Cluster assignations are made online and, additionally, in order to adapt to changes that can happen in the distribution of the data, some algorithms include mechanisms to update cluster assignments by adding and removing clusters as observations arrive. For example, proposal [143] uses a nature-inspired (bird flocking) optimization algorithm to control the evolution of the clusters over time, allowing merge or split operations of microclusters. In [100] microclusters are represented as Gaussian Mixture Models (GMMs) that can be compared using the Kullback–Leibler divergence and merged if they are sufficiently similar. The approach followed in [103] uses the distances of a sample to its two nearest prototypes to measure the uncertainty of the NIDS outcome in an active learning setup. When distances are very similar, the human manager is required to take corrective measures over the system (select the correct prototype, request a new one, merge both prototypes, etc).

A second less common group of unsupervised incremental NIDS uses the temporal information of the stream to identify attacks. Their objective is to incrementally learn and update a model that will somehow describe the normal evolution of the stream. Any observation that does not follow the expected behavior will be flagged as an attack. The difference between the proposals in this group lies mainly in the method used to model the evolution of the stream. Proposal [141] leverages the incremental clustering algorithm introduced in [84], in which a new cluster is created each time that a sample lies outside the limited area (using a fixed radius) of current clusters, and analyzes the transitions between cluster assignments of consecutive traffic samples. Another example is [104], where a Long Short Term Memory (LSTM) network is used to model the evolution of the traffic stream. The LSTM accounts for the transition frequencies provoked by consecutive network samples in the cells of its architecture, updating the transitions with each new sample. In both cases, uncommon transitions are flagged as possible attacks.

Incremental supervised methods have also been proposed in NIDS, although rarely. An example of incremental supervised methods is [102], where an online version of the Adaboost algorithm, a classifier based on an ensemble of weak classifier models, is used. The method updates those weak classifiers with poorest performance each time a new data sample arrives. Labels for those samples are derived by a NIDS classifier that is retrained at fixed intervals. A NIDS model based on Reinforcement Learning is proposed in [119]. The method uses an incremental Neural Network that is updated (rewarded) with every prediction. Other examples include [16], [157], which use an ensemble of Very Fast Decision Trees (VFTD), an algorithm that learns classification trees incrementally using small batches of data under the assumption that classes do not evolve with time [191]. Again, the main drawback of this kind of methods is the requirement of class labels at operation time. Note that proposals [102], [119] assume that the actual label of all the samples is known during the update phase (the reinforcement phase in RL frameworks), something that may be problematic (cost, latencies, mistakes during the labeling, etc.). When labels are not needed for all the new incoming samples, a possible solution is to select the best candidates for labeling (as explained in Section IV-B) by means of active learning techniques [192]. Such techniques have been applied in [16].

In general, incremental approaches (supervised or unsupervised) lack maturity in the early stages of learning (when models have not yet been fed with enough data) [189]. To overcome this problem, some off-line data may be provided before deployment. An initial model is trained using batch learning, thus assuming an initial distribution of the data. After that, the model is incrementally updated as new samples arrive [5], [16], [100], [103], [157]. This approach has the advantage of relying on a well-learned detector in the initial phases, while overcoming the limitations of pure batch processing.

Most incremental NIDS methods are based on unsupervised approaches (see Table V). The main reason is that labels are not always easy to obtain, especially in real time. Unsupervised methods used in NIDS base their detection mechanism on thresholds (over frequencies, densities or distances). Those thresholds are generally fixed, even though these methods assume that traffic data evolves over time. As such, this evolution may not be captured adequately and this may result in a rise in the false positive or negative ratios [103], [141], [142]. In some cases, it is assumed that the thresholds follow some statistical distribution, used to dynamically estimate the confidence interval in which most normal observations should be found [100], [104]. Care must be taken with some distributions, such as the Normal, that are sensitive to outliers, as attackers may use this characteristic to evade detection, e.g., generating a high number of anomalies to widen the threshold and make attacks undetectable.

A small number of NIDS proposals are based on supervised incremental methods (see Table V). They should be able to learn to differentiate normal from malicious activities with the arrival of new (labeled) data. However, some supervised algorithms have difficulties adapting to changes in the distribution of the data [16]: they can adapt to the appearance of new classes in the data, but not to modifications of the characteristics of previously known classes. The adaptation rate depends on the specific characteristics of the algorithm. It can be slow unless a change-detection component is added, which would help to quickly forget the characteristics of old data [187].

VIII. Evaluation Stage

The objective of the evaluation stage is to measure how well the previous stages have performed. Ideally, this evaluation should be as rigorous as possible, using different criteria such as detection accuracy, complexity, adaptability, understand-ability, security, etc.

It is not easy to define valid metrics and comparison criteria for all these characteristics. Let us focus on complexity, very closely related to the ability to deploy a NIDS in a real environment. Only a few papers report complexity computations, and we find them incomplete [45], [46], [103], [177]: computational complexity of the data mining algorithm in use defines only a part of the cost of the system. The remaining steps of the KDD process also have a cost, and implementation issues (such as programming languages used, compilers, libraries, run-time environments etc.) play an important role in the processing ability of the NIDS. A fair evaluation of all these aspects is an extremely complex exercise that requires a comprehensive testing environment –something that is beyond the scope of this survey. Similar limitations could be stated for adaptability, understandability and security.

The predictive performance of a NIDS, however, can be measured, with many different metrics that have been proposed. These metrics are quantitative measures that are commonly used as yardsticks to compare NIDS proposals. This is why most authors have mainly paid attention to this performance criterion, and that is why we do the same in this review. Things are easier when comparing proposals following the same learning paradigm (either batch or incremental). However, note that, different aspects have to be considered when evaluating batch and incremental models. Thus, making their comparison more difficult.

TABLE V
CLASSIFICATION OF SURVEYED NIDS PROPOSALS, ARRANGED BY LEARNING AND DETECTION APPROACH

	Misuse	**Hybrid**	**Anomaly**
Batch	[46, 57, 60, 70, 77, 85, 87, 105–107, 111–113, 115, 116, 125, 127, 135, 136, 140, 145–147, 155, 160, 163, 166, 176]	[56, 72, 82, 101, 114, 117, 177]	[40, 41, 43–45, 64, 68, 69, 71, 83, 84, 86, 99, 109, 118, 126, 156, 181–183, 193]
Re-train	[59]		[54]
Incremental	[16, 102, 119, 157]		[5, 55, 63, 100, 103, 104, 141–143, 168]

A. Validation Frameworks

In a typical batch setting, where a static distribution of the data is assumed, k-fold cross-validation is a common approach to estimate the true predictive error of a model. Cross-validation (CV) is an honest performance estimation framework, where the dataset is randomly split into various subsets or folds, and the error estimate is calculated as the average of the results of the evaluations performed over the different folds [121]. Cross-validation entails several incompatibilities with the online and incremental framework: (1) samples are assumed to be time-dependent and (2) incremental models evolve as new data arrives [194]. As folds in cross-validation are selected at random and samples are assumed to be independent, the progressive evolution and time-dependencies of samples are broken, i.e., the evaluation does not reflect the detector's ability to deal with concept drift [189].

In incremental CV frameworks, the data folds used for training and evaluation are created in an online manner. There are a few variations but, in the most simple approach, each time a sample arrives, it is randomly assigned to one fold. A model is learnt with each fold and incrementally updated every time its fold receives a new sample. As in standard CV, each model is evaluated using all the folds not used to train it. The final performance metric is obtained as an average of the metrics calculated for all the models. A slightly more complex approach assigns the new incoming sample to $k-1$ folds and updates the corresponding $k-1$ models. All models are evaluated on the data that has not been used to learn them (assigned to their fold).

A different approach is called *prequential* evaluation. Every time a new sample arrives, the prediction made by the model is compared against the actual label of the sample. After this evaluation, the model is updated using the sample [194]. The final prequential error is computed as the accumulated sum of all the errors obtained. Incremental cross-validation and prequential evaluation can be combined as k-fold prequential evaluation. The result is a more robust error estimate [194].

All these approaches assume that the labels of samples are known beforehand. They can deal with partially labeled data, computing the error using only the labeled samples [189]. However, note that when very few records contain labels, the estimation of the error might be poor, as the approximation of the error converges to the true error as more labeled data is available [189], [194].

A characteristic problem of the previous incremental error estimation methods is the high errors reported at the initial stages of learning, when the models are immature. Also, errors may increase due to concept drift. To reduce this effect, forgetting mechanisms, such as sliding windows and fading factors, have been proposed. With sliding windows, an error estimate is computed for the most recent k samples, while fading factors apply a weight to discount older information from the computation of the error estimate [189]. In both cases, the chosen window size and the decay factor parameters are critical to perform fast detection, i.e., rapid adaptation of the model to recent changes [189], [195].

TABLE VI
CONFUSION MATRIX. ROWS CORRESPOND TO THE PREDICTED CLASS AND COLUMNS TO THE ACTUAL CLASS. THE VALUE REPRESENTED IN EACH POSITION OF THE MATRIX INDICATES THE NUMBER OF INSTANCES PREDICTED, AS STATED BY THE ROW, THAT ACTUALLY BELONG TO THE CLASS INDICATED BY THE COLUMN

		Actual class	
		Positive	**Negative**
Predicted	**Positive**	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

B. Evaluation Metrics

In terms of ML, predictive performance metrics measure how well the predictions provided by an algorithm match with the true labels of the input samples. The confusion matrix, see Table VI, is used to easily visualize the detection performance of a NIDS. For binary detection problems (normal vs. attack), it is common to see the positive class as attacks, while negative means normal activity. For multiclass problems, where more than two classes of traffic can be detected, it is common to use a *one-vs-all* procedure: the positive class is the one that we want to measure, while the negative class groups all the instances of the remaining classes.

As can be seen in Table VII, several metrics (that receive different names in different works) can be derived from this matrix, each one focusing on a different facet of the prediction capacity of a NIDS. Note that some of them, such as Detection Rate (DR), True Negative Rate (TNR), and Accuracy and Precision, must be interpreted as *the higher, the better*, taking values close to 1 when the detector performs very well. Most of these metrics account for the proportion of samples for which the model predicts its class correctly. In contrast, the best detectors should report values close to 0 for False Positive Rate (FPR) and False Negative Rate (FNR), as they represent failure ratios.

Another common metric is the ROC curve. It visually shows the trade-off between the TPR and FPR of classifiers, by means

TABLE VII
COMMON EVALUATION METRICS DERIVED FROM THE CONFUSION MATRIX

Metric	Formula	Metric	Formula
DR, TPR, Recall, Sensitivity	$\frac{TP}{TP+FN}$	Precision	$\frac{TP}{TP+FP}$
FAR, FPR, Specificity	$\frac{FP}{FP+TN}$	Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
TNR	$\frac{TN}{FP+TN} = 1-FPR$	Error Rate	$\frac{FP+FN}{TP+TN+FP+FN} = 1-Acc$
FNR, Miss Rate	$\frac{FN}{TP+FN} = 1-TPR$	F-score/F-measure	$2 * \frac{Precision * Recall}{Precision + Recall}$

of a bi-dimensional plot, for different values of the decision threshold. The area under the ROC curve (AUC) is also used as a metric to evaluate NIDS.

Not all the metrics used by the machine learning community are meaningful for the specific NIDS problem. Network traffic is a highly unbalanced scenario in which normal traffic is much more common than attacks. Moreover, network attacks may have a severe impact on the victim and, thus, metrics used for NIDS should well reflect the effects of not detecting an attack (false negatives). A robust metric in unbalanced scenarios is the Kappa statistic. It quantifies the behavior of a predictive model in contrast to a random chance detector [196]. The kappa statistic can be seen as a correlation coefficient between the predictions of the model and the true labels, taking a value of 1 when the predictive model always guesses the class of the samples correctly, 0 when it behaves as a random chance detector and -1 when it fails all the predictions.

The most common metrics reported in the surveyed NIDS literature are DR (Detection Rate) and FPR (False Positive Rate), as they inform about complementary detection abilities. DR measures how well the system identifies abnormal behaviours as attacks, while FPR measures the ratio of normal observations incorrectly flagged as attacks. The goal of a good IDS is to maximize the former while minimizing the latter. Summary metrics that combine others (such as the F-measure) are also used, but should be considered as supplementary information due to its poor explainability. ROC curves are useful as they provide a graphical way to compare detectors [55], [100]. However, when plots are very similar, additional metrics may be needed for further clarification. Although being a useful indicator for NIDS, where imbalance is common, the Kappa statistic is not reported in the literature. This fact contrasts with the use of some metrics, such as accuracy, which do not properly reflect the performance in unbalanced scenarios [197], and has been reported in several works [56], [72], [115], [147], [155], [176].

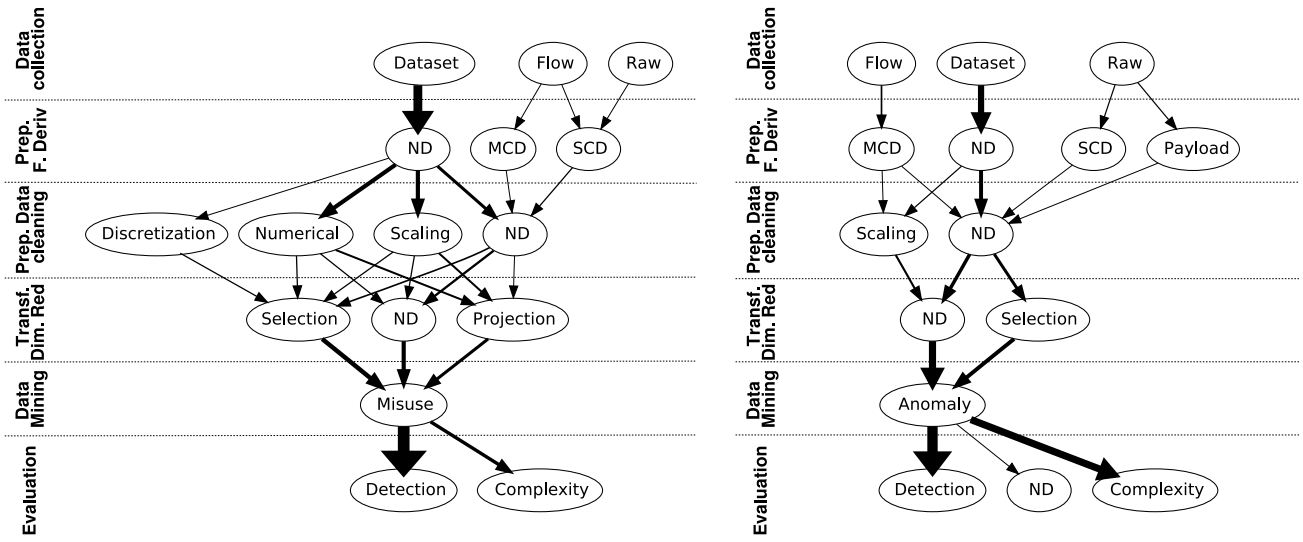
IX. DISCUSSION, CONCLUSION AND OPEN ISSUES

An ideal NIDS should fulfill a large set of desirable characteristics including, among others, the following: early detection of attacks, high attack-detection coverage (ability to detect the maximum number of attacks), high detection rate with low false alarm rate, ability to scale according to network requirements, and robustness to attacks targeting the detector [6], [11]. In this section we summarize the conclusions extracted from our literature review, as well as a collection of open issues and challenges identified, which may constitute future lines of research in the area of network security.

In order to have a general view of the use by NIDS authors of the techniques discussed in this paper, we have rendered two graphs (see Figures 7(a) and 7(b)) that represent the paths followed when designing (and evaluating) a NIDS, for both misuse and anomaly-based systems. In these graphs, each level corresponds to a different phase of the KDD process, and the nodes within each phase correspond to the methods used in the surveyed articles. Nodes are joined by arrows, whose thickness represents the proportion of works that apply a particular (destination) method after the (source) method of the previous phase. To allow a better understanding of the graphs, we have simplified them by removing nodes for methods with low utilization (below 5%). In general, note that main paths (sequences of thick arrows) are characterized by the presence of acronym "ND" (method Not Discussed for that phase), due to two main reasons: (1) many articles do not mention the methods applied to the data, and (2) oftentimes, data mining algorithms are applied directly to "cooked" records obtained from public datasets.

A. Reproducible Research

A first conclusion of our literature review that can be inferred from the previous paragraph is that most papers provide insufficient detail about their pipeline and, more specifically, about the procedures used for feature derivation, transformation and validation. In some cases, this is due to the use of public datasets in which all this work has already been done. However, we have found several articles that apply data mining algorithms that can only deal with numerical data over the set of heterogeneous records provided by a public dataset, without specifying any feature removal or feature transformation method [59], [99], [100], [103]–[106]. In some other cases, transformations are vaguely described: "we transformed categorical variables to a numerical representation", without stating how [101], [107]. Such a lack of details about the procedures carried out to build NIDS hinder their analysis, evaluation and comparison. A fact that has been reflected in our analysis, where some discussions may seem superficial given the difficult to properly find the grounds behind



(a) misuse approaches

(b) anomaly approaches

Fig. 7. Paths along KDD of misuse and anomaly surveyed approaches. Levels represent the different phases of KDD. Nodes represent methods used at each stage. Edge thickness depicts the proportion of the surveyed works using the method below.

the use of some techniques for NIDS. In this sense, as a recommendation for future work in the area, more effort is required towards favouring reproducible research, especially by providing all the details of the whole procedure used to build a NIDS.

B. Data Collection, Feature Extraction and Public Datasets

Graphs 7(a) and 7(b) show how, in the first phase of KDD (data collection), most NIDS use public datasets for both model construction and evaluation purposes. In Section IV-C, we have already explained how these datasets were generated, also pointing out their deficiencies (short capture periods, discontinued captures, uncertainty during the labeling process and lack of realism, among others). A NIDS that works successfully with any of these datasets is not guaranteed to operate equally well in real scenarios. Thus, it is not possible to perform a proper evaluation of NIDS with a single dataset; the use of various datasets, with different characteristics, has to be considered. Ideally, datasets which satisfy most of the mentioned characteristics, and that reflect the current use of the network by applications/nodes, are desirable. These databases should be captured at different network locations, and should thus provide different views of the network.

Another potential weakness related to the use of public datasets lies in that each one provides a particular set of already derived features, and most researchers work only with those. Available features may be useless when trying to detect newer attacks, or may contain data that is impossible to obtain in real scenarios, for example features annotated by human experts. To guarantee the derivation of custom features and provide more flexibility to researchers, the availability of raw packet data in public datasets is a must. Indeed, a few NIDS authors derive their own set of features from raw data (see Figures 7(a) and 7(b)), normally SFD, MFD or payload features, that have shown to improve the accuracy of the detector [61].

Regarding payload features, we have already mentioned that the trend is that most applications interchange data in a secure, encrypted form. Nonetheless, most public datasets do not contain this kind of traffic (another reason that makes them unrealistic). This absence of encryption has allowed the extraction of payload features, useful to detect some attacks that leave their footprint in the content of a connection. However, either ciphered or in cleartext, obtaining these features is barely possible in real world high-speed networking scenarios. Efficient capture and decryption mechanisms are required to extract the payload of plaintext connections with low overheads. A step towards a solution may be the use of probes installed at the hosts (that is, once the data is received and decrypted), but at the cost of increasing network traffic (payload still has to be sent to the NIDS for analysis) and losing some early detection capabilities. In this sense, HIDS can directly manage this data at the host [51].

Precisely because dealing with payload data is anything but easy, many authors have limited their detectors to use SFD and MFD features in order to identify point and collective malicious patterns. These derived features are usually very simple: flow, packet or byte counts and average rates. We think that more sophisticated variables could be useful to increase detection performance, widen the spectrum of identified attacks, harden the security of a NIDS and identify evasion attempts. For example, features that summarize the additional traffic activity caused by a network connection (number of different service requests, such as DNS, ARP, etc.), together with indicators of causal relationships, may also be useful for the detection of contextual patterns of network attacks. In general, the extraction and use of additional SFD and MFD features needs to be further explored, when dealing with different kinds of attacks and evasion scenarios [198].

C. Feature Transformation and Dimensionality Reduction

Our graphs (see Figures 7(a) and 7(b) again) show how feature scaling methods or transformations of categorical features into numerical representations are more common in misuse systems than in anomaly detectors. The main reason is that some data mining algorithms (mainly supervised classifiers) require datasets which are free of outliers, balanced in terms of labels and homogeneous in terms of data type natures (numerical or categorical) to work properly.

Numerical transformations of categorical data allow the use of data mining algorithms that can only deal with numerical data. Some of these representations, such as dummy features or frequency encoding, substantially increase the length of the feature vectors. Consequently, dimensionality reduction methods, such as FSS and projections, become mandatory. Some anomaly-based NIDS performs this reduction through the manual removal of unwanted features.

This combination of techniques is used to enable the use of data mining algorithms and also to increase their detection accuracy. While this may be a good solution for static environments, what happens when network traffic changes? New threats may leave traces in features incorrectly modified (discretized or scaled, with unknown categories...) or selected for removal, as this was done using criteria based on obsolete traffic patterns. Most surveyed NIDS are based on techniques devised for stationary approaches, making them unrealistic. As a solution to this problem, dynamic preprocessing and transformation techniques need to be explored.

D. Data Mining Algorithms and Learning Schemes

As regards the data mining phase, misuse systems are mainly based on supervised classifiers, while unsupervised and weakly supervised methods are the choice for anomaly detection. Based on our review of the literature, we have seen that many works based on supervised classifiers have erroneously used the term "anomaly detection" to refer to their detection approaches. In view of this, we have concluded that the existing definitions for the concepts of misuse-based and anomaly-based NIDS do not allow a clear categorization of the existing NIDS, and in this work we have provided a new taxonomy to avoid such miscategorizations.

Also, we have pointed out the limitations of batch learning methods when dealing with evolving traffic patterns. Despite this, we have shown that most published works use batch learning, implicitly assuming static and unrealistic traffic patterns. Sometimes retraining mechanisms are implemented and the training of the new model is done while the NIDS is operating with the obsolete model, which in fact is dangerous as the number of false alarms may increase, while attacks may pass undetected [11]. The number of incremental learning methods that consider networks as continuously evolving scenarios is very small for both misuse and anomaly detectors. For misuse incremental detectors, human intervention is often required to label new or unknown records before updating the model. Active learning can be of help in this sense. Anomaly-based incremental detectors are prone to generating many false alarms and may be easily evaded. In general, we think that incremental learning approaches for NIDS, taking into account their risks and benefits, should be investigated further.

The small interest paid by the authors to temporal dependencies in the input data when designing detectors also needs to be highlighted. Indeed, we have briefly discussed this issue as a missing aspect in the feature derivation step (see Section IX-B for this discussion). A few works apply elementary time series analysis to exploit temporal relations, but a thorough study of the temporal relations is missing. Most proposals analyze traffic data once connection statistics have been computed, i.e., detection is carried out over connections that have already occurred; therefore, an attack is detected after it starts taking place. Performing early detection of malicious traffic with models that exploit temporal patterns and dependencies among different communication protocols could be of interest since it can limit attack consequences. For example, the analysis of ARP flows could be used to detect some man-in-the-middle attacks at an early stage. In general, we think that these approaches require further attention.

E. NIDS Evaluation Beyond Accuracy

A common issue when comparing NIDS papers is the lack of an adequate framework to evaluate and compare different proposals. Authors often use unfit metrics (we have already discussed how some metrics that are commonplace in ML literature are meaningless in NIDS) and obviate the evaluation of important properties beyond those related to accuracy. Complexity, adaptation ability and resistance to attacks towards the NIDS itself should be assessed.

Complexity is sometimes discussed in the NIDS literature (see the lower part of Figures 7(a) and 7(b)) but, in our opinion, this issue deserves further research efforts. A theoretically excellent detector is not practical if response times are high and attacks are reported only after the attacker has fulfilled his/her purpose. An effort to collect the *training* complexity of NIDS proposals was made in [11], but the computational cost associated to this phase is only critical for retraining procedures. In addition, and as we have seen throughout this survey, the operational cost of NIDS detectors may be dependent of any other steps apart from detection (data preprocessing, data transformation...). Such costs should be reported, as they would indicate whether or not a NIDS may work in real time, and with how many resources.

The evaluation of the adaptability of incremental learning approaches is also an open issue [199]. These learning mechanisms need to be robust to noise (or attacks) while being able to adapt to newer traffic patterns. In other words, a balance between sensitivity and robustness has to be found by means of a proper evaluation of different variability scenarios, a fact barely discussed in NIDS that use incremental learning.

Finally, assessments of the security features of NIDS proposals are also infrequent. As can be seen in Figures 7(a) and 7(b), there is no node representing this aspect of the evaluation. A NIDS plays an important role in the protection of the corporate network, and should be resistant against attacks that target it [167]. To the best of our knowledge, the first

article discussing this risk dates from 2006 [200]; posterior research also tackles this issue [61], [201]–[203]. After that, a reduced number of NIDS proposals incorporate some protection mechanism, but of limited scope or without performing any evaluation [45], [69], [157]. More effort must be placed on researching NIDS security, as well as on applying a framework to evaluate it [204].

REFERENCES

[1] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Comput. Netw.*, vol. 31, no. 8, pp. 805–822, 1999.

[2] C. V. Brown, *IS Management Handbook*. Boca Raton, FL, USA: CRC Press, 1999.

[3] R. Heady, G. F. Luger, A. Maccabe, and M. Servilla, *The Architecture of a Network Level Intrusion Detection System*. Washington, DC, USA: Univ. New Mexico, 1990.

[4] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 305–316.

[5] T. Huang, H. Sethu, and N. Kandasamy, "A new approach to dimensionality reduction for anomaly detection in data traffic," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 651–665, Sep. 2016.

[6] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Security*, vol. 28, nos. 1–2, pp. 18–28, 2009.

[7] 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.

[8] J. J. Davis and A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: A review," *Comput. Security*, vol. 30, nos. 6–7, pp. 353–375, 2011.

[9] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart., 2014.

[10] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *J. Netw. Comput. Appl.*, vol. 60, pp. 19–31, Jan. 2016.

[11] 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.

[12] M. F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Comput. Security*, vol. 70, pp. 238–254, Sep. 2017.

[13] N. Moustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 128, pp. 33–55, Feb. 2019.

[14] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.

[15] S. Garcia, J. Luengo, J. A. Saez, V. Lopez, and F. Herrera, "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 734–750, Apr. 2012.

[16] 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.

[17] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The kdd process for extracting useful knowledge from volumes of data," *Commun. ACM*, vol. 39, no. 11, pp. 27–34, 1996.

[18] E. Maiwald, *Network Security: A Beginner's Guide*. New York, NY, USA: McGraw-Hill Professional, 2001.

[19] B. A. Forouzan, *Cryptography & Network Security*. New Delhi, India: McGraw-Hill, Inc., 2007.

[20] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," *Comput. Security*, vol. 24, no. 1, pp. 31–43, 2005.

[21] M. Thottan and C. Ji, "Anomaly detection in IP networks," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2191–2204, Aug. 2003.

[22] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita, "Network attacks: Taxonomy, tools and systems," *J. Netw. Comput. Appl.*, vol. 40, pp. 307–324, Apr. 2014.

[23] G. Piateski and W. Frawley, *Knowledge Discovery in Databases*. Cambridge, MA, USA: MIT Press, 1991.

[24] P. M. Domingos, "A few useful things to know about machine learning.," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[25] W. Stallings, *Computer Networking With Internet Protocols and Technology*. Upper Saddle River, NJ, USA: Pearson/Prentice-Hall, 2004.

[26] R. Hofstede *et al.*, "Flow monitoring explained: From packet capture to data analysis with netflow and IPFIX," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2037–2064, 4th Quart., 2014.

[27] L. Degioanni, S. McCanne, C. J. White, and D. S. Vlachos, "Distributed network traffic data collection and storage," U.S. Patent 8 971 196, Mar. 3, 2015.

[28] V. Jacobson, C. Leres, and S. McCanne, *The TCPDUMP Manual Page*, Lawrence Berkeley Lab., Berkeley, CA, USA, vol. 143, 1989.

[29] C. Satten. (2008). *Lossless Gigabit Remote Packet Capture With Linux*. [Online]. Available: https://staff.washington.edu/corey/gulp/

[30] S. Campbell and J. Lee, "Intrusion detection at 100g," in *Proc. Int. Conf. High Perform. Comput. Netw. Stor. Anal. (SC')*, 2011, pp. 1–9.

[31] S. Frankel and S. Krishnan, "IP security (IPSEC) and Internet key exchange (IKE) document roadmap," IETF, RFC 6071, Feb. 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6071.txt

[32] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information," IETF, RFC 7011, pp. 2070–2721, 2013.

[33] B. Claise, "Cisco systems netflow services export version 9," IETF, RFC 3954, Oct. 2004.

[34] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP flow information export (IPFIX)," IETF, RFC 3917, 2004.

[35] N. Duffield *et al.*, "Sampling for passive Internet measurement: A review," *Stat. Sci.*, vol. 19, no. 3, pp. 472–498, 2004.

[36] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?" in *Proc. 6th ACM SIGCOMM Conf. Internet Meas.*, 2006, pp. 165–176.

[37] F. Raspall, "Efficient packet sampling for accurate traffic measurements," *Comput. Netw.*, vol. 56, no. 6, pp. 1667–1684, 2012.

[38] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proc. IEEE Symp. Security Privacy (SP)*, 2005, pp. 183–195.

[39] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM Trans Inf. Syst. Security*, vol. 3, no. 4, pp. 227–261, 2000.

[40] C. Krüegel, T. Toth, and E. Kirda, "Service specific anomaly detection for network intrusion detection," in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2002, pp. 201–208.

[41] M. V. Mahoney and P. K. Chan, "Learning rules for anomaly detection of hostile network traffic," in *Proc. 3rd IEEE Int. Conf. Data Min.*, Nov. 2003, pp. 601–604.

[42] A. Nenkova and K. McKeown, *A Survey of Text Summarization Techniques*. Boston, MA, USA: Springer, 2012, pp. 43–76.

[43] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld, "Active learning for network intrusion detection," in *Proc. 2nd ACM Workshop Security Artif. Intell.*, 2009, pp. 47–54.

[44] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "MCPAD: A multiple classifier system for accurate payload-based anomaly detection," *Comput. Netw.*, vol. 53, no. 6, pp. 864–881, 2009.

[45] D. Ariu, R. Tronci, and G. Giacinto, "Hmmpayl: An intrusion detection system based on hidden Markov models," *Comput. Security*, vol. 30, no. 4, pp. 221–241, 2011.

[46] T. Hamed, R. Dara, and S. C. Kremer, "Network intrusion detection system based on recursive feature addition and bigram technique," *Comput. Security*, vol. 73, pp. 137–155, Mar. 2018.

[47] J. Oltsik, "White paper: Network encryption and its impact on network security," Enterprise Strategy Group, Inc., Milford, MA, USA, Rep., Feb. 2019.

[48] J. B. Rubin, J. Besehanic, and R. P. Borland, "Intercepting encrypted network traffic for Internet usage monitoring," U.S. Patent 8 914 629, Dec. 2014.

[49] J. Li, R. Chen, J. Su, X. Huang, and X. Wang, "ME-TLS: Middlebox-enhanced TLS for Internet-of-Things devices," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1216–1229, Feb. 2020.

[50] Z. Durumeric *et al.*, "The security impact of https interception," in *Proc. NDSS*, 2017, pp. 1–14.

[51] V. Bukac, P. Tucek, and M. Deutsch, "Advances and challenges in standalone host-based intrusion detection systems," in *Proc. Int. Conf. Trust Privacy Security Digit. Bus.*, 2012, pp. 105–117.

[52] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, nos. 23–24, pp. 2435–2463, 1999.

[53] Q. LLC. (2019). *Argus: Auditing Network Activity*. [Online]. Available: https://qosient.com/argus/index.shtml

[54] P. Casas, J. Mazel, and P. Owezarski, "Unada: Unsupervised network anomaly detection using sub-space outliers ranking," in *Proc. Int. Conf. Res. Netw.*, 2011, pp. 40–51.

[55] J. Dromard, G. Roudière, and P. Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 1, pp. 34–47, Mar. 2017.

[56] D. Barbara, N. Wu, and S. Jajodia, "Detecting novel network intrusions using Bayes estimators," in *Proc. SIAM Int. Conf. Data Min.*, 2001, pp. 1–17.

[57] B. G. Atli, Y. Miche, A. Kalliola, I. Oliver, S. Holtmanns, and A. Lendasse, "Anomaly-based intrusion detection using extreme learning machine and aggregation of network traffic statistics in probability space," *Cogn. Comput.*, vol. 10, no. 5, pp. 848–863, 2018.

[58] N. Moustafa and J. Slay, "Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *Proc. Military Commun. Inf. Syst. Conf. (MilCIS)*, 2015, pp. 1–6.

[59] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Introducing deep learning self-adaptive misuse network intrusion detection systems," *IEEE Access*, vol. 7, pp. 13546–13560, 2019.

[60] F. Palmieri, U. Fiore, and A. Castiglione, "A distributed approach to network anomaly detection based on independent component analysis," *Concurrency Comput. Pract. Exp.*, vol. 26, no. 5, pp. 1113–1129, 2014.

[61] I. Homoliak, M. Teknos, M. Ochoa, D. Breitenbacher, S. Hosseini, and P. Hanacek, "Improving network intrusion detection classifiAers by non-payload-based exploit-independent obfuscations: An adversarial approach," *EAI Endorsed Trans. Security Safety*, vol. 5, no. 17, p. e4, Dec. 2018.

[62] T. Kovanen, G. David, and T. Hämäläinen, "Survey: Intrusion detection systems in encrypted traffic," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. Cham, Switzerland: Springer, 2016, pp. 281–293.

[63] A. Yamada, Y. Miyake, K. Takemori, A. Studer, and A. Perrig, "Intrusion detection for encrypted Web accesses," in *Proc. IEEE 21st Int. Conf. Adv. Inf. Netw. Appl. Workshop (AINAW)*, vol. 1, 2007, pp. 569–576.

[64] R. Koch, M. Golling, and G. D. Rodosek, "Behavior-based intrusion detection in encrypted environments," *IEEE Comm. Mag.*, vol. 52, no. 7, pp. 124–131, Jul. 2014.

[65] M. R. Stytz and S. B. Banks, "Method and apparatus for preventing network traffic analysis," U.S. Patent 6 917 974, Jul. 12, 2005.

[66] T. H. Ptacek and T. N. Newsham, "Insertion, evasion, and denial of service: Eluding network intrusion detection," Secure Netw., Inc., Calgary, AB, Canada, Rep., 1998.

[67] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis," in *Proc. NDSS*, vol. 9, 2009, pp. 1–14.

[68] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 217–228, 2005.

[69] P. Winter, H. Lampesberger, M. Zeilinger, and E. Hermann, "On detecting abrupt changes in network entropy time series," in *Proc. IFIP Int. Conf. Commun. Multimedia Security*, 2011, pp. 194–205.

[70] S. Jin, D. S. Yeung, and X. Wang, "Network intrusion detection in covariance feature space," *Pattern Recognit.*, vol. 40, no. 8, pp. 2185–2197, 2007.

[71] D. S. Yeung, S. Jin, and X. Wang, "Covariance-matrix modeling and detecting various flooding attacks," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 2, pp. 157–169, Mar. 2007.

[72] S.-Y. Ji, B.-K. Jeong, S. Choi, and D. H. Jeong, "A multi-level intrusion detection method for abnormal network behaviors," *J. Netw. Comput. Appl.*, vol. 62, pp. 9–17, Feb. 2016.

[73] A. Jensen and A. la Cour-Harbo, *Ripples in Mathematics: The Discrete Wavelet Transform*. Heidelberg, Germany: Springer, 2001.

[74] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Security*, vol. 31, no. 3, pp. 357–374, 2012.

[75] M. Almgren and E. Jonsson, "Using active learning in intrusion detection," in *Proc. 17th IEEE Comput. Security Found. Workshop*, 2004, pp. 88–98.

[76] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "Ugr'16: A new dataset for the evaluation of cyclostationarity-based network IDSs," *Comput. Security*, vol. 73, pp. 411–424, Mar. 2018.

[77] E. K. Viegas, A. O. Santin, and L. S. Oliveira, "Toward a reliable anomaly-based intrusion detection in real-world environments," *Comput. Netw.*, vol. 127, pp. 200–216, Nov. 2017.

[78] S. Baek, D. Kwon, J. Kim, S. C. Suh, H. Kim, and I. Kim, "Unsupervised labeling for supervised anomaly detection in enterprise and cloud networks," in *Proc. IEEE 4th Int. Conf. Cyber Security Cloud Comput. (CSCloud)*, 2017, pp. 205–210.

[79] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proc. 6th Int. Conf.*, 2010, p. 8.

[80] C. Wheelus, E. Bou-Harb, and X. Zhu, "Tackling class imbalance in cyber security datasets," in *Proc. IEEE Int. Conf. Inf. Reuse Integr. (IRI)*, 2018, pp. 229–232.

[81] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[82] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 5, pp. 649–659, Sep. 2008.

[83] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *Proc. ACM CSS Workshop Data Min. Appl. Security (DMSA)*, 2001, pp. 1–25.

[84] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of Data Mining in Computer Security*. Boston, MA, USA: Springer, 2002, pp. 77–101.

[85] H. H. Nguyen, N. Harbi, and J. Darmont, "An efficient local region and clustering-based ensemble system for intrusion detection," in *Proc. 15th Symp. Int. Database Eng. Appl.*, 2011, pp. 185–191.

[86] G. Giacinto, R. Perdisci, M. Del Rio, and F. Roli, "Intrusion detection in computer networks by a modular ensemble of one-class classifiers," *Inf. Fus.*, vol. 9, no. 1, pp. 69–82, 2008.

[87] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, 2016, pp. 1–5.

[88] K. R. Kendall, *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*, Ph.D. dissertation, Dept. Elect. Eng., Massachusetts Inst. Technol., Cambridge, MA, USA, 1999.

[89] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Comput. Netw.*, vol. 34, pp. 579–595, Oct. 2000.

[90] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD Cup 99 data set," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl. (CISDA)*, 2009, pp. 1–6.

[91] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation," in *Proc. 1st Workshop Build. Anal. Datasets Gathering Exp. Returns Security*, 2011, pp. 29–36.

[92] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards generating real-life datasets for network intrusion detection," *Int. J. Netw. Security*, vol. 17, no. 6, pp. 683–701, 2015.

[93] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," *J. Netw. Comput. Appl.*, vol. 87, pp. 185–192, Jun. 2017.

[94] K. Mivule and B. Anderson, "A study of usability-aware network trace anonymization," in *Proc. IEEE Sci. Inf. Conf. (SAI)*, 2015, pp. 1293–1304.

[95] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Security*, vol. 3, no. 4, pp. 262–294, 2000.

[96] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection," in *Proc. Int. Workshop Recent Adv. Intrusion Detect.*, 2003, pp. 220–237.

[97] C. Brown, A. Cowperthwaite, A. Hijazi, and A. Somayaji, "Analysis of the 1999 darpa/lincoln laboratory ids evaluation data with netadhict," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, Jul. 2009, pp. 1–7.

[98] K. Siddique, Z. Akhtar, F. A. Khan, and Y. Kim, "KDD Cup 99 data sets: A perspective on the role of data sets in network intrusion detection research," *Computer*, vol. 52, no. 2, pp. 41–51, 2019.

[99] J. Z. Lei and A. A. Ghorbani, "Improved competitive learning neural networks for network intrusion and fraud detection," *Neurocomputing*, vol. 75, no. 1, pp. 135–145, 2012.

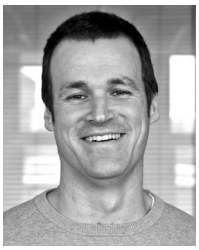
[100] E. Bigdeli, M. Mohammadi, B. Raahemi, and S. Matwin, "Incremental anomaly detection using two-layer cluster-based structure," *Inf. Sci.*, vol. 429, pp. 315–331, Mar. 2018.

[101] M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "A multi-step outlier-based anomaly detection approach to network-wide traffic," *Inf. Sci.*, vol. 348, pp. 243–271, Jun. 2016.
[102] W. Hu, J. Gao, Y. Wang, O. Wu, and S. Maybank, "Online adaboost-based parameterized methods for dynamic distributed network intrusion detection," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 66–82, Jan. 2014.
[103] S. Roshan, Y. Miche, A. Akusok, and A. Lendasse, "Adaptive and online network intrusion detection system using clustering and extreme learning machines," *J. Franklin Inst.*, vol. 355, no. 4, pp. 1752–1779, 2018.
[104] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, Nov. 2017.
[105] S. T. Sarasamma, Q. A. Zhu, and J. Huff, "Hierarchical kohonenen net for anomaly detection in network security," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 2, pp. 302–312, Apr. 2005.
[106] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowl. Based Syst.*, vol. 78, pp. 13–21, Apr. 2015.
[107] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[108] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artif. Intell. Rev.*, vol. 22, no. 3, pp. 177–210, 2004.
[109] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," in *Proc. Int. Conf. Data Min.*, Jan. 2003, pp. 1–10.
[110] D. Barbara, J. Couto, S. Jajodia, L. Popyack, and N. Wu, "Adam: Detecting intrusions by data mining," in *Proc. IEEE Workshop Inf. Assurance Security*, 2001, pp. 1–6.
[111] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 2986–2998, Oct. 2016.
[112] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *VLDB J.*, vol. 16, no. 4, pp. 507–521, 2007.
[113] E. De la Hoz, E. De La Hoz, A. Ortiz, J. Ortega, and B. Prieto, "PCA filtering and probabilistic som for network intrusion detection," *Neurocomputing*, vol. 164, pp. 71–81, Sep. 2015.
[114] C. Guo, Y. Ping, N. Liu, and S.-S. Luo, "A two-level hybrid approach for intrusion detection," *Neurocomputing*, vol. 214, pp. 391–400, Nov. 2016.
[115] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[116] M. Sheikhan, Z. Jadidi, and A. Farrokhi, "Intrusion detection using reduced-size RNN based on feature grouping," *Neural Comput. Appl.*, vol. 21, no. 6, pp. 1185–1190, 2012.
[117] 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, 2005.
[118] V. L. Cao, M. Nicolau, and J. McDermott, "Learning neural representations for network anomaly detection," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 3074–3087, Aug. 2019.
[119] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112963.
[120] C. C. Aggarwal, *Data Mining: The Textbook*. Cham, Switzerland: Springer, 2015.
[121] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY, USA: Springer, 2009.
[122] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. Heidelberg, Germany: Springer, 1998, pp. 9–50.
[123] S. Wiesler and H. Ney, "A convergence analysis of log-linear training," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 657–665.
[124] D. Tax and R. Duin, "Feature scaling in support vector data descriptions," in *Proc. Learn. Imbalanced Datasets*, 2000, pp. 25–30.
[125] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1184–1199, 2011.
[126] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "A system for denial-of-service attack detection based on multivariate correlation analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 447–456, 2013.
[127] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2016, pp. 195–200.
[128] M. J. Zaki and J. W. Meira, *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, May 2014.
[129] H. Liu, F. Hussain, C. L. Tan, and M. Dash, "Discretization: An enabling technique," *Data Min. Knowl. Disc.*, vol. 6, no. 4, pp. 393–423, 2002.
[130] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 194–202.
[131] J. R. Quinlan, "Improved use of continuous attributes in C4.5," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 77–90, 1996.
[132] Y. Yang and G. I. Webb, "Proportional k-interval discretization for naive-bayes classifiers," in *Proc. Eur. Conf. Mach. Learn.*, 2001, pp. 564–575.
[133] Y. Yang and G. I. Webb, "A comparative study of discretization methods for naive-bayes classifiers," in *Proc. PKAW*, 2002, pp. 159–173.
[134] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proc. IJCAI*, 1993, pp. 1022–1029.
[135] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A combination of discretization and filter methods for improving classification performance in KDD Cup 99 dataset," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2009, pp. 359–366.
[136] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a hidden naïve bayes multiclass classifier," *Expert Syst. Appl.*, vol. 39, no. 18, pp. 13492–13500, 2012.
[137] J. Gama and C. Pinto, "Discretization from data streams: Applications to histograms and data mining," in *Proc. ACM Symp. Appl. Comput.*, 2006, pp. 662–667.
[138] S. Ramírez-Gallego, S. García, and F. Herrera, "Online entropy-based discretization for data streaming classification," *Future Gener. Comput. Syst.*, vol. 86, pp. 59–70, Sep. 2018.
[139] J. P. Early and C. E. Brodley, "Behavioral features for network anomaly detection," in *Machine Learning and Data Mining for Computer Security*. London, U.K.: Springer, 2006, pp. 107–124.
[140] M. S. Hoque, M. A. Mukit, and M. A. N. Bikas, "An implementation of intrusion detection system using genetic algorithm," *Int. J. Netw. Security Appl.*, vol. 4, no. 2, p. 109, 2012.
[141] M. Hahsler and M. H. Dunham, "Temporal structure learning for clustering massive data streams in real-time," in *Proc. SIAM Int. Conf. Data Min.*, 2011, pp. 664–675.
[142] F. Angiulli and F. Fassetti, "Detecting distance-based outliers in streams of data," in *Proc. 16th ACM Conf. Inf. Knowl. Manag.*, 2007, pp. 811–820.
[143] A. Forestiero, "Self-organizing anomaly detection in data streams," *Inf. Sci.*, vol. 373, pp. 321–336, Dec. 2016.
[144] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.
[145] D. S. Kim, H.-N. Nguyen, and J. S. Park, "Genetic algorithm to improve svm based network intrusion detection system," in *Proc. IEEE 19th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, vol. 2, 2005, pp. 155–158.
[146] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," *Comput. Security*, vol. 81, pp. 148–155, Mar. 2019.
[147] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee, "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection," *Appl. Soft Comput.*, vol. 12, no. 10, pp. 3285–3290, 2012.
[148] M. A. Hall, *Correlation-Based Feature Selection for Machine Learning*, Ph.D. dissertation, Dept. Comput. Sci., Univ. Waikato, Hamilton, New Zealand, 1999.
[149] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Somerset, U.K.: Wiley, 2012.
[150] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artif. Intell.*, vol. 151, nos. 1–2, pp. 155–176, 2003.
[151] Z. Zhao and H. Liu, "Searching for interacting features," in *Proc. 20th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2007, pp. 1156–1161.
[152] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*. Pacific Grove, CA, USA: Cengage Learn., 2011.
[153] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann, 1993.
[154] J. R. Quinlan, "Boosting, bagging, and C4.5," in *Proc. 13th Nat. Conf. Artif. Intell.*, 1996, pp. 725–730.

[155] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas, "Modeling intrusion detection system using hybrid intelligent systems," *J. Netw. Comput. Appl.*, vol. 30, no. 1, pp. 114–132, 2007.
[156] J. Zhang and M. Zulkernine, "Anomaly based network intrusion detection with unsupervised outlier detection," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 5, 2006, pp. 2388–2393.
[157] 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. (GLOBECOM)*, 2017, pp. 1–6.
[158] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[159] I. Jolliffe, "Principal component analysis," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Heidelberg, Germany: Springer, 2011, pp. 1094–1096.
[160] F. Salo, A. B. Nassif, and A. Essex, "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection," *Comput. Netw.*, vol. 148, pp. 164–175, Jan. 2019.
[161] A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Netw.*, vol. 13, nos. 4–5, pp. 411–430, 2000.
[162] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.
[163] V. A. Golovko, L. U. Vaitsekhovich, P. A. Kochurko, and U. S. Rubanau, "Dimensionality reduction and attack recognition using neural network approaches," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2007, pp. 2734–2739.
[164] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 759–766.
[165] I. Zliobaite and B. Gabrys, "Adaptive preprocessing for streaming data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 309–321, Feb. 2014.
[166] C.-F. Tsai and C.-Y. Lin, "A triangle area based nearest neighbors approach to intrusion detection," *Pattern Recognit.*, vol. 43, no. 1, pp. 222–229, 2010.
[167] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Dept. Comput. Eng., Chalmers Univ. Technol., Göteborg, Sweden, Rep., 2000.
[168] P. Casas, P. Mulinka, and J. Vanerio, "Should i (re) learn or should i go (on)?: Stream machine learning for adaptive defense against network attacks," in *Proc. 6th ACM Workshop Moving Target Defense*, 2019, pp. 79–88.
[169] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artif. Intell. Rev.*, vol. 22, no. 2, pp. 85–126, 2004.
[170] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Process.*, vol. 99, pp. 215–249, Jun. 2014.
[171] A. Carreño, I. Inza, and J. A. Lozano, "Analyzing rare event, anomaly, novelty and outlier detection terms under the supervised classification framework," *Artif. Intell. Rev.*, vol. 55, pp. 3575–3594, Oct. 2019.
[172] O. Maimon and L. Rokach, "Introduction to knowledge discovery and data mining," in *Data Mining and Knowledge Discovery Handbook*. Boston, MA, USA: Springer, 2009, pp. 1–15.
[173] J. Hernández-González, I. Inza, and J. A. Lozano, "Weak supervision and other non-standard classification problems: A taxonomy," *Pattern Recognit. Lett.*, vol. 69, pp. 49–55, Jan. 2016.
[174] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-Supervised Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 2010.
[175] D. M. J. Tax, "One-class classification. Concept-learning in the absence of counter-examples," Ph.D. dissertation, Comput. Sci., Delft Univ. Technol., Delft, The Netherlands, 2001.
[176] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 2, 2002, pp. 1702–1707.
[177] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Syst. Appl.* , vol. 41, no. 4, pp. 1690–1700, 2014.
[178] T. Kohonen, *Self-Organization and Associative Memory*, vol. 8. Heidelberg, Germany: Springer, 2012.
[179] G. Seni and J. F. Elder, "Ensemble methods in data mining: Improving accuracy through combining predictions," *Synth. Lectures Data Min. Knowl. Disc.*, vol. 2, no. 1, pp. 1–126, 2010.
[180] J. R. Goodall *et al.*, "Situ: Identifying and explaining suspicious behavior in networks," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 204–214, Jan. 2019.
[181] M. E. Otey, A. Ghoting, and S. Parthasarathy, "Fast distributed outlier detection in mixed-attribute data sets," *Data Min. Knowl. Disc.*, vol. 12, nos. 2–3, pp. 203–228, 2006.
[182] A. Koufakou and M. Georgiopoulos, "A fast outlier detection strategy for distributed high-dimensional data sets with mixed attributes," *Data Min. Knowl. Disc.*, vol. 20, no. 2, pp. 259–289, 2010.
[183] J. D. Brutlag, "Aberrant behavior detection in time series for network monitoring," in *Proc. LISA*, vol. 14, 2000, pp. 139–146.
[184] D. J. Weller-Fahy, B. J. Borghetti, and A. A. Sodemann, "A survey of distance and similarity measures used within network intrusion anomaly detection," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 70–91, 1st Quart., 2015.
[185] M. J. Kearns, *The Computational Complexity of Machine Learning*, Ph.D. dissertation, Harvard Univ., Cambridge, MA, USA, 1990.
[186] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. De Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Comput. Surveys*, vol. 46, no. 1, p. 13, 2013.
[187] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surveys*, vol. 46, no. 4, p. 44, 2014.
[188] H.-L. Nguyen, Y.-K. Woon, and W.-K. Ng, "A survey on data stream clustering and classification," *Knowl. Inf. Syst.*, vol. 45, no. 3, pp. 535–569, 2015.
[189] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, no. 3, pp. 317–346, 2013.
[190] M. Kloft and P. Laskov, "Online anomaly detection under adversarial impact," in *Proc. 13th Int. Conf. Artif. Intell. Stat.*, 2010, pp. 405–412.
[191] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2001, pp. 97–106.
[192] S. J. Huang, R. Jin, and Z.-H. Zhou, "Active learning by querying informative and representative examples," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 892–900.
[193] T. Lane and C. E. Brodley, "Temporal sequence learning and data reduction for anomaly detection," *ACM Trans. Inf. Syst. Security*, vol. 2, no. 3, pp. 295–331, 1999.
[194] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2015, pp. 59–68.
[195] J. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2009, pp. 329–338.
[196] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.
[197] G. Santafe, I. Inza, and J. A. Lozano, "Dealing with the evaluation of supervised classification algorithms," *Artif. Intell. Rev.*, vol. 44, no. 4, pp. 467–508, 2015.
[198] I. Homoliak and P. Hanacek. (2019). *ASNM Datasets: A Collection of Network Traffic Features for Testing of Adversarial Classifiers and Network Intrusion Detectors*. [Online]. Available: https://arxiv.org/abs/1910.10528
[199] G. Krempl *et al.*, "Open challenges for data stream mining research," *SIGKDD Explore Newslett.*, vol. 16, pp. 1–10, Sep. 2014.
[200] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proc. ACM Symp. Inf. Comput. Commun. Security*, 2006, pp. 16–25.
[201] J. Newsome, B. Karp, and D. Song, "Paragraph: Thwarting signature learning by training maliciously," in *Proc. Int. Workshop Recent Adv. Intrusion Detect.*, 2006, pp. 81–105.
[202] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Disc. Databases*, 2013, pp. 387–402.
[203] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Security*, 2017, pp. 506–519.
[204] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 984–996, Apr. 2014.

Borja Molina-Coronado received the B.Sc. degree in computer engineering from the Technical University of Valencia, in 2015, and the M.Sc. degree in computer science from the University of the Basque Country UPV/EHU in 2017, where he is currently pursuing the Ph.D. degree with the Department of Computer Architecture and Technology. His main research interests include machine learning and network security.

Usue Mori received the M.Sc. degree in mathematics, and the Ph.D. degree in computer science from the University of the Basque Country UPV/EHU, Spain, in 2010 and 2015, respectively, where she has been working as a Lecturer with the Department of Computer Science and Artificial Intelligence since 2019. Her main research interests include clustering and classification of time series.



Alexander Mendiburu (Member, IEEE) received the B.Sc. degree in computer science and the Ph.D. degree from the University of the Basque Country, Spain, in 1995 and 2006, respectively. He is a Full Professor with the Department of Computer Architecture and Technology, University of the Basque Country UPV/EHU, where he has been working since 1999. His main research areas are evolutionary computation, time series, probabilistic graphical models, and parallel computing.

Jose Miguel-Alonso (Member, IEEE) received the M.Sc. degree in computer science and the Ph.D. degree in computer science from the University of the Basque Country UPV/EHU in 1989 and 1996, respectively, where he is a Full Professor with the Department of Computer Architecture and Technology. Formerly, he was a Visiting Assistant Professor with Purdue University. He carries out research related to parallel and distributed systems, in areas, such as network security, performance modeling and resource management in large-scale computing systems. He is a member of the IEEE Computer Society and the HiPEAC European Network of Excellence.