IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Exploring the Integration of Edge Computing and Machine Learning in Enhancing Security for IoT Systems: Opportunities and Challenges in Intrusion Detection

Mr. Ramani K, Dr. N Chandrakala, Mr. G Binu
Assistant Professor department of computer science, Assistant Professor and Head, Research Scholar FT
RD National College of Arts and Science, Erode, SSM College of arts and science, Komarapalayalam, Namakkal, Bharathidasan college of Arts and Science, Ellispettai, Affiliated to Bharathiar university

Abstract

The key elements of current cybersecurity approaches involve Intrusion Detection Systems (IDSs), which employ various techniques and architectures for intrusion detection. IDSs can utilize either a signature-based approach, involving cross-checking monitored events with a database of known intrusion patterns, or an anomaly-based approach, where the system learns normal behavior and detects deviations. This study focuses on surveying the utilization of IDSs in Internet of Things (IoT) networks, particularly in conjunction with edge computing to support IDS implementation. The paper identifies emerging challenges in deploying IDS in edge scenarios and proposes solutions. Emphasis is placed on anomaly-based IDSs, highlighting primary techniques for anomaly detection. Additionally, machine learning techniques and their application in the IDS context are explored, along with an analysis of the anticipated advantages and disadvantages associated with specific techniques.

Keywords – Intrusion Detection Systems, Internet of Things, Anomaly Detection, Machine Learning

NOTE: Title, abstract and keywords must be identical to the ones submitted electronically in EDAS – Editor's Assistant. Use the command \ITUnote to achieve the appropriate formatting.

1. INTRODUCTION

An Intrusion Detection System (IDS) refers to a software or hardware element designed to recognize malicious activities within computer systems or networks, ensuring the preservation of security. Host-Based Intrusion Detection Systems (HIDS) are designed for a specific computer system, while Network-Based Intrusion Detection

Systems (NIDS) are tailored for entire networks. NIDS can be either hardware devices or software components strategically placed within a network to analyze the generated traffic from hosts and devices [1]. This study concentrates on NIDS, and henceforth, the term IDS will specifically denote Network-Based Intrusion Detection Systems.

The application of IDS in the context of the Internet of Things (IoT) is not a novel concept, and various solutions have been suggested [2][3]. Traditional IDSs tailored for IoT are typically situated at either the device or gateway level, as depicted in Fig. 1, often relying on cloud computing for operation. However, recent advancements in Edge Computing (EC) have introduced new opportunities for enhancing IoT security. EC extends the Cloud Computing paradigm to the network's edge, allowing intelligent computing at edge devices to mitigate network latency, which is particularly crucial for IoT applications.

Nevertheless, the adoption of edge computing introduces potential vulnerabilities that could be exploited by malicious actors. Edge nodes, especially those deployed in public areas, might be susceptible to unauthorized remote access or physical tampering. If an attacker gains control over an edge node, they could manipulate the traffic passing through it, selectively forwarding packets or injecting new ones to masquerade as a legitimate device. Traditional IDSs placed at the device or gateway level would be unable to detect such attacks, given that the intrusion occurs in a different network segment.

On the contrary, deploying IDSs at the network edge (as indicated by the red box in Figure 1) presents new challenges that impact the reliability of these IDSs. To address these challenges, it is essential to develop specific IDSs tailored for edge environments.

Within this framework, this study aims to achieve three main objectives:

• Establishing a taxonomy for Intrusion Detection Systems (IDSs) and examining their applications in the IoT domain.

• Identifying challenges and opportunities associated with implementing IDS at the Edge.

• Conducting a comprehensive survey of IDSs utilizing Machine Learning approaches specifically designed for IoT, applicable at various levels of IoT architectures.

The subsequent sections of this paper are structured as follows. Section 2 delves into the classification of signature-based and anomaly-based IDSs. Section 3 explores the application of IDSs in IoT environments, while Section 4 focuses on Edge-enabled solutions. Section 5 addresses the novel challenges arising when deploying an IDS in an edge environment, outlining their impact on existing IDS and proposing potential remedies. Finally, Section 6 provides an overview of widely used machine learning techniques applied to IDS. For each technique, we present

a summary of the anticipated advantages and disadvantages that the IDS should exhibit. Conclusions are offered in Section 7.

2. INTRUSION DETECTION SYSTEM TAXONOMY

The primary objective of an Intrusion Detection System (IDS) is to prevent unauthorized access to an information system, as any such access may pose a threat to the confidentiality, integrity, or availability of information. The IDS fulfills this role by scrutinizing network traffic and/or resource usage, issuing an alert when it identifies malicious activities.

IDSs can be broadly classified into two main categories based on the strategy they employ to detect intrusions. The system may either cross-check monitored events with a database of known intrusion techniques or learn the normal behavior of the system and report any occurrence of anomalous events. These strategies are referred to as signature-based and anomaly-based, respectively.

2.1 Signature-based IDS

In essence, Signature-based Intrusion Detection Systems (SIDSs) belong to a category of systems that rely on a database containing "signatures" of known attacks. These signatures, representing ongoing activities, are extracted, and matching methods or protocol conformance checks are employed to compare them with those in the database. If a match is identified, an alarm is triggered. SIDSs can function in both online mode, directly monitoring hosts and generating real-time alarms, and offline mode, where system activity logs are analyzed. This class is also referred to in the literature as Misuse Detection or Knowledge-based Detection [4]. The creation of traffic signatures can be a laborious and time-consuming task, depending on the number and complexity of the considered traffic "features."

Typically, these signatures are manually crafted by experts with detailed knowledge of the exploits the system is designed to detect. Christian Kreibich et al. [5] proposed a system for the automatic generation of malicious traffic signatures, extending the open-source honeypot honeyd [6] with a subsystem inspecting traffic at various protocol hierarchy levels, integrated with existing IDS. Early SIDS approaches focused on analyzing individual network packets and matching them with rules in the database. However, there may be a need to extract and match signatures that span multiple packets, leading to the creation of matching rules based on previously observed packets. Meiners et al. [7] introduced a signature matching method based on finite state machines, implementing a hardware-based regular expression matching approach using small Ternary Content Addressable Memories chips. Lin et al. [8] presented an algorithm that performs matching using Backward Hashing and Aho-Corasick algorithms, partitioning signatures for efficient selection of matching algorithms based on signature features. This system was implemented in the HIDS ClamAV.

Sheikh [9] et al. proposed a signature-based Intrusion Detection System tailored for IoT environments, comprising four cascading subsystems. The system extracts signatures of known attack types stored in a database, regularly updated to enhance detection accuracy. Subsequently, sessions are merged to extract features, and a novel pattern-matching algorithm is applied to compare incoming data with known signatures. Finally, the system produces logs for inspection by the system administrator, providing insight into the current and historical status.

Despite their effectiveness, signature-based IDSs encounter limitations when faced with zero-day attacks, where the attack's signature is absent from the IDS's database. The increasing frequency of zero-day attacks [10] diminishes the overall performance of signature-based IDS. Consequently, anomaly-based IDSs have emerged as a new class, modeling the normal behavior of a computer system and alerting on any significant deviation from the established baseline.

2.2 Anomaly-based IDS

Anomaly-based Intrusion Detection Systems (AIDS) were introduced to address the shortcomings of signature-based IDS. Typically, AIDS undergo a training phase where they construct a model of the system's normal behavior. Once deployed, the IDS monitors computer hosts, comparing their behavior to the established model. When a significant deviation is observed between the hosts' behavior and the model, the IDS may trigger an alert. This strategy potentially enables an anomaly-based IDS to detect zero-day attacks, as it does not rely on matching current host behavior to attack signatures in a database. Another advantage of an anomaly-based IDS is its complexity for attackers to understand the normal behavior of a target host without engaging in transactions, as communication with the target could likely trigger an alert [11], [12]. Furthermore, anomaly-based IDS can serve not only for security purposes but also as a system analysis tool. Anomalies reported by the IDS indicate deviations from baseline conditions, suggesting not only intrusion but also potential bugs in the device's logic.

However, a notable limitation of an AIDS is its tendency to produce a higher rate of false positives compared to a SIDS. During operation, the targeted system may undergo slight or drastic behavior changes without any intrusion occurring. If an AIDS patient is not equipped to recognize such possibilities, it may generate false alerts.

Anomaly-based IDSs can be categorized into three subgroups, each differentiated by its modeling and detection techniques: statistics-based, knowledge-based, and machine learning-based. Both statistics-based and machine learning-based IDSs construct a model of normal host behavior, whereas knowledge-based IDSs concentrate on identifying anomalies based on system data, such as network protocols or patterns in data exchange, as provided by system administrators. The utilization of various IDSs belonging to distinct classes is not mutually exclusive. Different IDS categories have the capacity to detect specific attacks that might go unnoticed by other IDSs using alternative detection techniques. This approach enables the implementation of multi-tier security.

2.2.1 Statistics-based AIDS

In the learning phase, a statistical-based IDS constructs a probability distribution model of the computing system's nominal behavior using statistical techniques. This model is developed by measuring various parameters and events occurring in the computing system. Upon deployment, the IDS assesses the probability of all monitored events within the system and triggers alerts for events with low probabilities. The "Univariate" strategy is the simplest approach to build the statistical model, treating each measurement independently. An advancement of this is the "Multivariate" strategy, identifying correlations and relationships between multiple measurements. Ye et al. [13] proposed a hybrid system combining univariate and multivariate elements, creating profiles for each measurement individually and discovering multivariate correlations to reduce false positives. Tan et al. [14] developed a system for detecting DoS attacks using multivariate correlation analysis (MCA) to understand normal traffic patterns and achieve high accuracy in DoS detection.

Addressing the challenge of high-dimensional data, some systems [15], [16], [17], [18] utilized principal component analysis (PCA) to reduce input vector dimensions before applying standard multivariate statistical techniques. Another statistical technique family utilizes time series data [19], also applied in network anomaly detection [20]. By considering time in probability calculations, these techniques raise alerts for events unlikely to occur at specific times. Viinikka et al. [21] used time series techniques by aggregating individual alerts into an alert flow, enhancing multivariate analysis precision and lowering false positive rates.

Qingtao et al. [22] focused on detecting abrupt-change anomalies, utilizing the Auto-Regressive (AR) process for data modeling and sequential hypothesis testing for anomaly determination. Zhao et al. [23] mined frequent patterns in network traffic, applying time-decay factors to distinguish between newer and older patterns. This approach aids AIDS in updating its system baseline to adapt to users' highly dynamic behavior.

When developing an AIDS, particularly one exploiting time series data, attention must be given to data seasonality—periodic variations influenced by factors like holidays, work hours, or weather. Reddy et al. [24] proposed an algorithm to detect outliers in seasonality-affected time series data using a double pass of Gaussian Mixture Models (GMMs). In the learning phase, time is divided into seasonal time bins, GMMs are trained, and outliers are removed. Another set of GMMs is then built on the cleaned data, enhancing anomaly detection performance.

2.2.2 Knowledge-based AIDS

Knowledge-based Anomaly Intrusion Detection Systems (AIDS) belong to the realm of expert systems. These systems rely on a knowledge source that encapsulates legitimate traffic signatures, treating any event deviating from this profile as an anomaly. Typically, this knowledge is manually crafted and may encompass rules regarding the expected traffic patterns of systems, incorporating Finite State Machines (FSM) applied to internet protocols like IP, TCP, HTTP, etc., to ensure host compliance with these protocols. Walkinshaw et al. [25] applied FSMs to

entire network traffic, representing system activities through states and transitions. The resulting FSM characterizes the system's nominal behavior, and any deviation is considered an attack. Studnia et al. [26] developed a system based on a description language that defines attack characteristics. Knowledge-based AIDSs leverage a precise model of the entire computing system, enabling them to achieve a low rate of false positives compared to other solutions.

However, crafting such a model can be a challenging and cumbersome task, potentially impractical. The model needs to be flexible enough to adapt to dynamic changes in system behavior, limiting the application of this family of AIDS to predictable traffic sources [26]. Ensuring protocol compliance through FSM could be difficult, as it requires modeling the state machine on top of the targeted protocol, which can be complex. Additionally, there is a risk of implementation bugs, especially if the IDS uses the same implementation as the hosts (e.g., open protocol stacks implemented in the Linux kernel), potentially leading to the duplication of bugs. In such cases, the IDS may fail to identify protocol violations and may introduce vulnerabilities to the entire Intrusion Detection System.

2.2.3 Machine Learning-based AIDS

The realm of cybersecurity has extensively utilized Machine Learning (ML) [27]. Various specialized branches of ML, such as Data Mining [28], Deep Learning [29] [30], Deep Reinforcement Learning [31], and more recently, Adversarial Learning [32], have been harnessed for the development of Anomaly Intrusion Detection Systems (AIDS). ML-based AIDSs employ Machine Learning models to autonomously acquire a representation of the typical conditions within the computing system.

When developing a Machine Learning (ML) system, the initial phase involves identifying the features of the data to be analyzed, and this holds true for Intrusion Detection Systems (IDS) as well [33]. Initial efforts are dedicated to assessing the effectiveness of traffic features, utilizing publicly available datasets and applying fundamental ML algorithms. Studies by Khraisat [34], Bajaj [35], and Elhag [36] assess the significance of dataset features using Information Gain (IG), Correlation Attribute Evaluation, and genetic-fuzzy rule mining methods. Through this evaluation, they filter out features with low IG or redundant information. Subsequently, algorithms like C4.5 Decision Tree, Naïve Bayes, NB-Tree, Multi-Layer Perceptron, SVM, and k-means Clustering are applied. Other techniques employed for IDS feature selection include Principal Component Analysis (PCA) [37], [38], and Genetic Algorithms (GA) [39].

Machine Learning models can undergo training either with or without ground-truth labels, with the respective techniques termed Supervised Learning and Unsupervised Learning. In the Supervised Learning approach, the ML algorithm receives both the data and their corresponding true labels (anomaly / not anomaly). Techniques utilizing this training strategy include Support Vector Machines (SVMs), Artificial Neural Networks (ANNs), Decision Trees, among others. An AIDS trained through supervised learning typically exhibits high accuracy,

given that the ML model is acquainted with anomalous events through training on both normal and abnormal occurrences.

However, the availability of true data labels, often found in accessible datasets, may not be present in a production environment. These labels might be occasional or possibly manually crafted. Furthermore, the anomalous data samples need to be sufficiently diverse to encompass all potential anomalies or those that the IDS should have the capability to detect.

The Unsupervised Learning approach addresses the previously mentioned issue by dispensing with the need for input labels during training. Techniques within Unsupervised Learning encompass k-means clustering, Auto Encoders (AEs), Generative Adversarial Networks (GANs), among others. While Unsupervised Learning techniques are more straightforward to deploy in real-world scenarios without relying on ground-truth labels, they may be more susceptible to noise in the data, and the training dataset should be sufficiently extensive to incorporate diverse samples.

Semi-Supervised learning techniques, positioned between Supervised and Unsupervised Learning, have also been proposed for implementing an AIDS. These techniques use only a small amount of labeled data, achieving high accuracy and minimizing the task of data labeling compared to a Supervised Learning system. Overall, Machine Learning-based AIDSs offer a diverse array of techniques, allowing the final IDS to be adaptable to the requirements of the target deployment environment. They demand less knowledge compared to expert systems IDSs, and many ML techniques automatically learn data features, eliminating the need for manually crafting information for the IDS.

Recently, Recurrent Neural Networks (RNNs) and their specializations like Long Short-Term Memory (LSTMs) and Gated Recurrent Units (GRUs) have been proposed, enabling the IDS to analyze complex and unseen patterns in exchanges between hosts—something challenging for knowledge-based or statistics-based AIDS. However, ML-based IDSs tend to impose a significant resource footprint. Depending on the specific technique, the ML model can be computationally expensive, both in terms of memory and CPU usage. If utilizing Deep Learning techniques, a GPU might also be required. This complexity makes it challenging to deploy ML-based IDSs on devices with low computational capabilities, such as IoT devices. Moreover, consideration should be given to the prediction time required by heavy ML models, especially in systems with real-time constraints, where specialized hardware might be necessary.

For an overview of the most used approaches and their requirements, refer to Figure 2.

3. IDS for IOT

As outlined in Section 1, Intrusion Detection Systems (IDSs) designed for the Internet of Things (IoT) can be classified into two main categories: IoT-specific and IoT-agnostic. An IoT-specific IDS is tailored to devices using

a particular communication technology, such as 6LoWPAN, BLE, LoRaWAN, etc. Typically, this type of IDS is deployed on the same network as the IoT devices it monitors. These IDSs base their predictions on messages sent by IoT devices, utilizing control information specific to the technology in use, including checks for protocol compliance. Conversely, IoT-agnostic IDSs are not reliant on any specific IoT technology and utilize information available irrespective of the devices' current communication technology, such as TCP/IP traffic. This category of IDS is well-suited for deployment in edge environments, handling traffic from diverse devices employing different communication technologies.

An advantage of an IoT-specific IDS over its IoT-agnostic counterpart is its capability to detect low-level attacks originating at the device level. However, a single IoT-agnostic IDS can manage multiple IoT devices without the necessity of deploying a specific IDS for each available communication technology.

In the architectural depiction in Figure 1, an IoT-specific IDS typically operates in the network section highlighted in green, while an IoT-agnostic IDS functions in the one highlighted in red. Numerous IoT-specific IDSs have been proposed for various technologies, including Wi-Fi [40] [41] [42] [43], LoRa [44] [45] [46], ZigBee [47] [48], Bluetooth Low Energy [49] [50] [51]. These systems are typically expert systems that inspect the traffic between hosts and verify the compliance of each packet with technology-specific network protocols. Advanced systems can even identify attacks at the physical network layer (PHY), such as jamming. In a PHY attack, an attacker sends bits that deviate from the communication protocol, making the data unreadable by an external IDS and rendering the attack extremely challenging to detect. For instance, in [52], the authors propose a BLE physical layer attack that selectively disrupts the signal on specific channels whenever a device attempts to connect.

Emerging security challenges specific to the Internet of Things (IoT) pertain to network management and operations, encompassing aspects such as routing, topology control, and network maintenance. In the context of routing, novel protocols tailored for devices with constrained resources, such as the Routing Protocol for Low Power Lossy Network (RPL) [53], have been developed. RPL relies on the Destination Oriented Directed Acyclic Graph (DODAG), constructed by devices following the protocol, facilitating point-to-point and point-to-multipoint communications. However, malevolent actors can exploit vulnerabilities by crafting malicious packets to disrupt the protocol execution. Consequently, many Intrusion Detection Systems (IDSs) focus on verifying the proper execution of RPL by connected devices.

One prevalent attack is the Rank Attack [54], where a child node advertises a lower rank value than its actual rank. The rank value is crucial in determining the proximity of nodes to the root node in a multi-hop scenario, with the rank decreasing from the root to the children. In a Rank Attack scenario, messages might be forwarded along loops, deviating from the optimized path. To address this, Perrey et al. proposed TRAIL, a topology authentication scheme for RPL [55]. TRAIL employs cryptography primitives to validate the correct rank value of nodes and,

consequently, the accurate topology of the DODAG. Importantly, these cryptographic functions are computationally efficient, ensuring scalability even on low-power devices.

Chugh et al. explored another form of attack known as the Black Hole attack, wherein a malicious node intentionally drops all or a portion of packets routed through it. Their study on various scenarios in 6LoWPAN networks revealed that reliable detection of the Black Hole attack is challenging. Additionally, a family of RPL-based attacks includes Clone ID and Sybil attacks, where malicious nodes imitate the identity of legitimate nodes. Zhang et al. delved into Sybil attacks, categorizing them based on the attacker's ability and goals. They also presented defense mechanisms categorized into Social Graph-based Sybil Detection (SGSD), Behavior Classification-based Sybil Detection (BCSD), and Mobile Sybil Defend (MSD).

In [58], the authors introduced a system based on the Artificial Immune System (AIS). The Intrusion Detection System (IDS) is decentralized across IoT devices, edge nodes, and cloud nodes. Lightweight detectors are deployed on IoT devices, while on the edge, alerts undergo analysis and processing using Smart Data concepts. Subsequently, the cloud clusters the data and conducts the training of heavy-weight detectors. This approach ensures that the model training for heavy-weight detectors occurs on the cloud, with only the lightweight application being carried out by IoT devices.

Verzegnassi et al. [59] presented a system designed to identify Sybil and jamming attacks by assessing the conformity of network parameters over time. The system passively collects network statistics of IoT devices observed by the gateway, such as average signal strength and average packet rate. These statistics are projected onto subspaces with dimensions representing the number of devices, the number of considered parameters, and the time tick. The algorithm's output is the conformity value of device network parameters across time. Notably, a sudden change in conformity values may indicate the occurrence of an ongoing attack.

Like all expert systems, Intrusion Detection Systems (IDSs) tailored for the Internet of Things (IoT) typically demonstrate high accuracy and low false positive rates. However, they face limitations in detecting zero-days or instances of unconventional utilization of network resources by the hosts. In contrast, IoT-agnostic IDSs operate autonomously, irrespective of the communication technology employed by IoT devices. These IDSs may be implemented on IoT gateways, disregarding Physical (PHY) or Media Access Control (MAC) layer information, or in a separate subnetwork, where they capitalize on TCP/IP traffic features.

4. The Edge-Enabled Approach

The concept of edge computing, introduced to enhance the characteristics and reliability of traditional IoT applications [60], [61], brings several advantages. It allows IoT applications to delegate computational, storage, or management tasks to the edge nodes. This approach is anticipated to improve the quality of services by reducing latency, enabling real-time network management, and enhancing data management. In this context, security applications, including Intrusion Detection Systems (IDS), could be relocated to the edge, as indicated by the red

box in Figure 1. This transition could offer several benefits to an IDS, such as increased computational resources, enabling the use of more sophisticated algorithms, and expanded storage capabilities for storing system logs or conducting memory-intensive procedures.

Furthermore, deploying an IDS on the edge has the potential to provide lower latency compared to the cloud, which is crucial for real-time IoT applications. Additionally, an edge-based IDS should ideally be IoT-agnostic, meaning it does not rely on specific IoT communication technologies. By adopting such an IDS, it can effectively handle a diverse range of devices utilizing different communication technologies in a unified manner, eliminating the need for deploying a dedicated IoT-specific IDS for each device subnetwork.

Eskandari et al. [62] introduced the Passban IDS, a system designed to establish a protective layer on IoT devices directly connected to it. The system focuses on TCP/IP-oriented attacks, excluding those specific to IoT technologies such as Port Scanning, HTTP and SSH brute force, and SYN flood. Notably, the system, which does not require intensive computations, is suitable for deployment on cost-effective edge devices and/or IoT gateways like Raspberry Pis or equivalents. Despite addressing a relatively limited number of attacks, the Passban IDS exhibits a commendably low false positive rate and high accuracy. It stands out as one of the few fully-implemented IDSs, encompassing the detection algorithm and the alerting system with the support of a web user interface.

In another study, researchers [63] delved into the identification of malicious edge devices, given the privileged role these devices play in storing and processing data generated by potentially numerous IoT devices. The proposed framework utilizes a two-stage Markov Model, an anomaly-based IDS, and a Virtual Honeypot Device (VHD). Upon an IDS alert, the first stage of the Markov Model categorizes the specific fog node, while the second stage predicts whether attaching the VHD to the alerted edge node is warranted. The VHD retains logs from all connected edge nodes, offering a resource for later examination by experts. Additionally, in [64], a system was introduced to enhance the detection accuracy of an IDS by implementing fuzzy c-means and Artificial Neural Networks (ANNs) at the edge. The approach demonstrated superior accuracy, even for attacks with low-frequency occurrences, when compared to conventional ANN techniques.

Hafeez et al. [65] presented a system designed for anomaly detection at the network edge gateways. The system characterizes traffic using features independent of the IoT communication technology, relying solely on TCP/IP features observable at the edge. This approach allows multiple systems, each employing diverse IoT communication technologies, to connect to the same IDS. The researchers collected IoT data from a real-world test-bed for their dataset and examined the distribution of the considered features. They noted that a significant proportion of these features conformed well to a heavy-tailed Gaussian distribution. The final anomaly detection is executed using fuzzy clustering, achieving high accuracy and a low false positive rate on their custom dataset.

Schneible et al. [66] introduced a framework for implementing distributed anomaly detection on edge nodes. The system involves deploying Auto Encoder models across multiple edge nodes strategically positioned in various network regions. Anomaly detection follows the conventional Auto Encoder approach, as outlined in Section 6.4.1. Notably, the system exhibits a level of adaptivity: the edge nodes continuously update their models based on new observations, allowing them to identify emerging trends in network traffic. An individual edge node then transmits its updated model to a central authority, which aggregates these models and disseminates the updates to the other edge agents. The authors noted that this approach reduces overhead bandwidth, as the network traffic generated only conveys the Auto Encoders' models rather than all observed data. Auto Encoders were employed in this context for both anomaly detection and as an automated system for feature extraction, compressing observed data to minimize traffic between edge nodes and the central authority.

While edge nodes possess enhanced computing capabilities compared to IoT devices, they may lack resources for performing resource-intensive tasks like training heavy-weight ML models. Recognizing this challenge, some literature works have proposed systems that circumvent the need for intensive operations. Sudqi et al. [67] introduced a host IDS designed for energy-constrained devices. In a more sophisticated approach, Sedjelmaci et al. [68] devised a system that balances energy consumption and detection accuracy. Their system integrates a signature-based IDS, offering greater energy efficiency but potentially yielding more false positives, and an anomaly-based IDS, demanding more power but delivering a more precise analysis. During normal operation, only the signature-based IDS is active. When an alert is triggered, it is relayed to the anomaly-based IDS, which validates or dismisses it. Furthermore, the system is conceptualized as a security game model, where the anomaly-based IDS makes predictions based on the Nash Equilibrium. A limitation is the reliance on a continuously operational cloud for the system to function effectively. Anomaly detection techniques extend beyond identifying network intrusions; they can also serve to detect bugs in device firmware or deviations from the normal state of a system. In the domain of Industrial IoT (IIoT), initiatives have emerged to detect such anomalies.

Utomo et al. [69] have designed a system for detecting anomalies in sensor readings from power grids. Anomalous alerts serve not only as indicators of potential illegal intrusions but also play a crucial role in ensuring grid safety by preventing failures and blackouts. Due to the significant non-linearity in the readings, they employ an Artificial Neural Network (ANN) based on Long-Short Term Memory (LSTM) cells for anomaly detection. LSTM neural networks, falling under the Recurrent Neural Networks (RNN) category, excel in processing sequential data, making them suitable for analyzing sequences of sensor readings or words, as seen in Natural Language Processing (NLP).

Recognizing the limitations of a single Intrusion Detection System (IDS) running on the network perimeter, Niedermaier et al. [70] proposed a distributed IDS utilizing multiple Industrial Internet of Things (IIoT) agents on

edge devices. The central unit aggregates logs from these agents and performs anomaly detection through one-class classification techniques, assuming a known normal behavior learned by the agents. This IDS is designed to operate efficiently on low-power microcontrollers, as it does not demand intensive computations. Additionally, the authors have implemented a proof-of-concept, a practice not commonly found in similar research endeavors.

4.1 **Device Classification**

Recently, endeavors have been directed towards the identification and categorization of IoT devices by analyzing their network traffic patterns. By scrutinizing network packets, classifiers can be developed to differentiate devices according to their classes (e.g., motion sensors, security cameras, smart bulbs, and plugs). Additionally, these classifiers can learn device signatures, enabling an Intrusion Detection System (IDS) to raise alarms if unauthorized devices attempt to connect to the network. This capability to detect intrusions solely based on network traffic is essential for IDSs intended for deployment on the edge or in networks distinct from the ones they are monitoring [71].

Desai et al. [72] introduced a feature-ranking system designed for the classification of IoT devices. The significance of each feature is determined through statistical methods. To extract features from traffic flows, they employed 15-minute time windows, referred to as "activity periods," representing the duration from the reception of the first packet to the reception of the last packet specific to each device. Depending on the device class, these activity periods may vary in length. Their experiments involved training classifiers with the top-□ features, resulting in notable accuracy. This implies a substantial reduction in computational tasks without compromising accuracy. Thangavelu et al. [73] proposed a decentralized device fingerprinting technique, named DEFT, which identifies IoT device fingerprints. In this system, IoT gateways extract features from devices' traffic sessions, sending these features to central edge nodes for training Machine Learning models and classifiers. The trained classifiers are then transmitted back to the gateways, where the final device identification takes place. The system autonomously recognizes new devices based on their extracted fingerprints, eliminating the need for prior knowledge of traffic signatures. When a new device connects to the network or an existing device alters its typical traffic pattern (e.g., due to a firmware update), the gateway's classifier marks its traffic as low-probability, prompting the transmission of captured features to the edge node. If another device of the same unknown class connects to a different gateway, that gateway also sends features to the edge node. Consequently, the edge node can cluster and identify the new device category. This strategy operates efficiently within a Software Defined Network (SDN) framework, performing classification on a flow-wise basis within fixed time window frames, avoiding packet-wise classification, which can be resource-intensive.

In their work [76], the authors introduced an IoT device classification system based on TCP/IP features. The primary goal of the system is to place a device into one of four specified classes: motion sensors, security cameras, smart bulbs, and smart plugs. The classification relies on bidirectional TCP flows, taking into account selected

features such as packet sizes, inter-arrival times between the first sent and received packets, and utilizing t-Distributed Stochastic Neighbor Embedding (tSNE) to reduce dataset dimensionality. Despite utilizing only fundamental features of the TCP/IP stack, the approach demonstrates commendable accuracy and recall scores. This implies that the classifier does not demand intensive computations in both training and prediction phases. However, it's worth noting that the classifier's effectiveness is indirectly enhanced by the limited number of considered device categories.

Miettinen et al. [77] devised IoT SENTINEL, a system designed to discern the device type of each node. This encompasses details such as the manufacturer's name, model, and software version. The methodology involves constructing a machine learning model to leverage device type information against an external vulnerability database, facilitating the identification of vulnerabilities and the imposition of communication restrictions accordingly. During their evaluation, they determined that the system effectively identifies device types with minimal performance overhead. However, the challenge of determining how to restrict device communication based on identified criticalities remains nontrivial and was not addressed by the authors. Additionally, the system's practicality in real-world scenarios is compromised when custom firmware is utilized in IoT devices, rendering the vulnerability database obsolete.

Bai et al. [74] introduce a device classification methodology to distinguish new and previously unseen devices, marking a departure from the prevailing approach where device recognition typically requires prior training on the specific device. Their classification technique relies on information streams generated by devices and employs an LSTM-CNN model to capture time-dependent patterns in network traffic. Initial steps involve saving features such as timestamp, length, and various addresses for each captured packet. Subsequently, features are extracted by segmenting the traffic into fixed time windows of a specified duration (the exact value is not explicitly mentioned in their experiments and appears to be fixed, not adaptive). They differentiate between incoming and outgoing packets, categorizing them as user (TCP, UDP, MQTT, HTTP) or control packets (ICMP, ARP, DNS). Various statistics of the packets are derived, and the processed data is input into an LSTM network to learn an encoding of the data. This LSTM is then connected to a CNN network that finalizes the classification. The architecture is illustrated in Figure 3, and training is executed using standard backpropagation algorithms. The achieved results exhibit considerable accuracy, although there is potential for further enhancement.

In their work [75], the focus was on distinguishing between IoT and non-IoT device-generated network traffic. The authors opted not to utilize existing datasets; instead, they captured network traffic from campus devices over a three-week period. The feature distribution among the dataset samples is outlined in Figure 4. Seeking to enhance classifier performance, the authors employed clustering techniques for each feature, although the specific algorithm and methodology remain unclear. Despite achieving commendable accuracy in their results, it is noteworthy that the classifier necessitates training with network traffic from each target device. This limitation

renders it unsuitable for the classification of previously unseen devices, posing challenges for systems accommodating a diverse array of IoT devices.

Bikmukhamedov et al. [78] introduced a system designed to scrutinize traffic flows within IoT networks and classify them using a straightforward machine learning model. In this approach, various features, such as statistics pertaining to packet lengths and inter-arrival times, are taken into account for each flow. The researchers trained multiple classification algorithms, encompassing logistic regression, SVM, decision tree, and random forest. To enhance performance, they also applied PCA decomposition to the features. Despite achieving commendable final classification accuracy, it is important to note the absence of implemented flow extraction or feature extraction, a potentially challenging aspect in real-world systems due to the complexity of the utilized features.

Hafeez et al. [79] introduced IOT GUARD, a lightweight method designed to discern between malicious and benign IoT traffic through a semi-supervised approach. Although the approach is primarily unsupervised, a small fraction of labels requires manual verification, rendering the algorithm technically semi-supervised. It relies on Fuzzy C-Mean clustering (FCM). To enhance performance, the authors aggregated features of devices belonging to the same host and service. Notably, this aggregation strategy is based on the latest device connections rather than packet timestamps. This approach offers an advantage by avoiding time-based aggregation, which might be unsuitable for detecting attacks involving time delays between successive connection attempts. In contrast, connection-based aggregation techniques aggregate features over the last "n" connections, accommodating time delays introduced by an attacker. The evaluation utilized a private dataset not disclosed to the public. While the achieved accuracy was commendable, practical comparisons with existing solutions or baseline algorithms were not conducted.

5. Open issues for EDGE-Enabled Architectures

The architecture of an edge-enabled IoT application is depicted in Figure 1, illustrating that the edge network introduces new attack surfaces vulnerable to exploitation by malicious entities. Conventional IoT-focused IDSs are typically positioned at the gateway or device levels, primarily safeguarding against malicious IoT devices. However, attacks may specifically target the edge network, potentially causing an edge node to become compromised. Such compromises could result from remote attacks or physical tampering of the node. Given the pervasive deployment of edge nodes, especially in public areas, attackers may find it easier to tamper with these devices. If an attacker gains control over an edge node, they could manipulate all the traffic flowing through it. This manipulation might involve generating packet streams within the edge, pretending to be a legitimate IoT gateway or device, or selectively forwarding packets of interest while discarding others. Although the management of malicious edge nodes has been explored in the literature [63], it is often not an automated process.

Utilizing existing IDSs in an edge scenario is feasible, but new challenges emerge, impacting the Intrusion Detection System's reliability. These challenges encompass:

1**. Traffic Encryption:** When deployed on edge nodes or IoT gateways, IDSs may encounter encrypted traffic, assuming secure communication between IoT devices and the cloud. If deployed on edge nodes, the IDS observes encrypted traffic, limiting its ability to comprehend the contained information. It can only operate on non-encrypted fields like TCP/IP headers, timestamps, etc.

2. **High Resource Variability:** IDSs employ diverse techniques for detection, varying significantly in required computational resources. Similarly, edge nodes exhibit considerable computational resource variability, ranging from powerful PCs to resource-constrained devices like Raspberry Pi. The challenge lies in ensuring that the IDS's resource requirements align with the capabilities of the executing edge node, avoiding communication latency and optimizing overall system execution.

3. **Distributed IDS Architecture on Edge/IoT:** Addressing the resource variability, IDSs for the network edge necessitate a distributed architecture. A single IDS may comprise multiple subsystems collaborating to ensure proper functionality and enhance detection performance. However, this cooperation introduces complexities inherent in distributed systems, amplifying the IDS's overall intricacy.

4. **Aggregated Traffic** The distinction in protocol stacks between IoT devices and the edge may lead to observers, including IDSs, being unaware of the source end-device of a packet. This occurs when IoT gateways receive packets from end-devices using IoT-specific communication technology and generate new packets using edge protocols like TCP/IP. Further details on this issue and its implications will be elaborated in Section 5.1.

To establish communication schemes resilient to malicious edge nodes, the principles of distributed systems can be applied by considering edge nodes as potentially byzantine nodes [80]. This involves treating each packet passing through the edge as a byzantine consensus problem. However, theorems [80] stipulate that, in a non-authenticated and partially synchronous communication environment, the Byzantine Fault Tolerance (BFT) requires a consensus of \(3f + 1\) parties, where \(f\) is the maximum number of tolerated byzantine nodes, for successful consensus achievement. Yet, implementing this approach would demand the transmission of the same packet from multiple edge nodes. Furthermore, if the packet originated from an IoT device, duplicating the same packet across various edge nodes would result in unnecessary energy and network resource consumption.

A potential resolution to the previously mentioned issue involves the utilization of IPsec [81], a network protocol ensuring security at the network layer by providing packet header and data authenticity, integrity (AH and ESP modes), and encryption (ESP mode only). Through IPsec, malicious edge nodes are prevented from fabricating packets to pose as legitimate network users. However, certain challenges persist:

- IPsec does not safeguard against traffic rerouting or selective-forwarding. Attackers retain the ability to determine which packets to forward or discard (selective-forwarding) and may introduce additional delays when routing packets, potentially impacting the real-time characteristics of IoT applications.

- IPsec does not ensure security in the event of physical tampering. If a device undergoes tampering, attackers could gain access to the private keys of an edge node, thereby compromising the entire IPsec secure communication scheme for the device.

- IPsec introduces increased fractional overhead. In typical IoT applications, where packets sent from IoT devices are small, the payload data-to-header data ratio is low. Incorporating an additional control header further increases the payload data, making communication less efficient in terms of fractional overhead.

5.1 Aggregated traffic

An additional challenge arises when the edge lacks the capability to distinguish traffic flows originating from individual IoT devices. In essence, it may only be able to observe aggregated traffic generated by all devices collectively, treating it as if it originated from a single device. This challenge emerges when IoT devices and the edge utilize different protocol stacks, requiring the gateway to translate IoT protocols to those used by the edge/cloud. The analysis of aggregated traffic poses a significant obstacle for both signature-based and anomaly-based IDS, leading to less reliable predictions.

Let's examine the scenario illustrated in Figure 5. In this setup, IoT devices are linked to gateways through specific communication technologies (such as BLE, LoRa, etc.), while gateways are connected to both the edge and the cloud via TCP/IP. When IoT devices transmit data to the cloud, they dispatch a packet to their respective gateway utilizing their designated IoT communication technology. Subsequently, the gateway generates a new TCP/IP packet and transmits it to both the edge and the cloud. The newly formed packet from the gateway will bear the source IP address of the gateway, irrespective of the originating IoT end-device. Additionally, these packets might share the same IP destination address (pointing to the same application server) and employ identical TCP ports for every IoT end-device. Consequently, any observer situated beyond the gateways, including edge nodes, would find it impossible to identify the source device of an observed packet. The inability to distinguish between TCP flows would lead the edge node to perceive the observed traffic as if it originated from a single device, as it lacks the means to discern the connected devices beyond the gateways.

The aggregated traffic presents challenges for existing IDS, encompassing both signature-based and anomaly-based systems:

- **Signature-based IDSs:** These systems face difficulties in isolating packets associated with individual devices due to the amalgamation of traffic. Consequently, these IDSs struggle to extract patterns from the observed traffic stream, rendering them incapable of recognizing attack signatures. Potential solutions might involve adapting existing signature-based IDSs to mine patterns from cumulative traffic. However, challenges arise as the observed traffic represents the summation of various streams generated by individual devices. In some cases, a signature may be erroneously identified as malicious if the combined flows generate a signature match. This elevates the risk of false positives.

- **Anomaly-based IDSs:** Dealing with the heightened variance in aggregated traffic becomes a significant concern for anomaly-based systems. The cumulative traffic is expected to exhibit greater variance than the individual traffic flows from each IoT device. Anomaly detection requires learning the normal system state, i.e., the condition without anomalies. If the normal state is derived from cumulative traffic, the anomaly detection algorithm may encounter excessive variance. This increased variability introduces the possibility of classifying malicious anomalies as non-malicious deviations from the expected pattern, as these oscillations are deemed acceptable within the variance of the normal system state. This, in turn, raises the likelihood of false negatives.

Table 1 provides a comprehensive overview of the repercussions of cumulative traffic on both existing anomaly-based and signature-based IDS. To exemplify anomaly detection with aggregated traffic, Figure 6 is presented. An anomaly detection algorithm implemented at the edge must derive knowledge about the normal system behavior from the overall network traffic, encompassing all devices, rather than relying on device-specific traffic. However, the cumulative traffic introduces a higher level of variance compared to traffic analyzed on a per-device basis, potentially exerting a significant impact on the effectiveness of the anomaly detection approach.

To enhance anomaly detection at the edge, an initial step involves segmenting the cumulative traffic into individual flows, each corresponding to a distinct IoT device. Following this segmentation, established algorithms can be employed to understand the system's normal behavior, not from the aggregated traffic, but from the distinct flows of each device. It's crucial to note that an edge node alone cannot execute this task as it lacks information about the connected IoT devices beyond the gateways.

6. Machine learning techniques applied to IDSS

In this section we discuss some of the widely used Ma- chine Learning techniques applied to IDS and how they could be leveraged by an IDS deployed at the edge. For each technique, we briefly describe the theory behind it and then we illustrate the expected advantages and disadvantages of an IDS based on it. Requirements on computational power, storage capacity and real time re- sponse of each technique are highlighted, which make an approach suitable or not for an edge-oriented IDS. A comprehensive overview of the techniques applied to IDS is in Table 2.

6.1 Support Vector Machine

Support Vector Machines (SVMs) stand out as a widely used machine learning technique capable of both classification and regression tasks [82]. In classification, the SVM identifies a hyperplane that effectively separates instances with similar labels into distinct regions. This division allows the SVM to predict the label for a new instance $'$ through the following equation:

$$□' = \text{sign}(□ \cdot \ □' - □) \qquad (1)$$

The SVM seeks parameters □ and □ ∈ ℝ□ such that the hyperplane maximizes the separation between the regions. Assuming a standardized dataset for simplicity, with instances sharing the same labels lying on one of these hyperplanes, the points on these hyperplanes are termed Support Vectors. Geometrically, the distance between these hyperplanes is denoted as 2 and is referred to as the margin.

The ultimate optimization problem for SVM can be expressed as follows: ensuring that □ (□ · □ − □) is greater than or equal to 1 for □ = 1, ..., □ (Equation 2). This problem is typically addressed through Quadratic Programming. However, the presence of outliers in the training data can adversely affect the final hyperplane. To mitigate this issue, practitioners often turn to a soft-margin SVM. In this variant, slack variables □ are introduced, allowing certain data points to fall within the margin region, albeit at the expense of an increased error function. The support vectors are defined as the instances where □□ − □ equals 1 or −1.

The optimization problem is formulated as follows: { □□□(≥□ 0· ,□ − □) ≥ 1 − □□, for □ = 1, ..., □ (Equation 3), where □ is a parameter influencing the significance of the slack variables in the loss function. While our previous discussions assumed linear separability in the original data space, this may not always be the case. To address this, the SVM employs the kernel trick, transforming inner products into a higher-dimensional space, allowing for non-linear separability. Depending on the task at hand, suitable kernels can be chosen to make the data linearly separable in the new space.

To illustrate the application of the kernel trick, we implemented this technique on a practical IoT dataset obtained from a LoRaWAN system provided by an Italian service provider (UNIDATA S.p.A.). This LoRaWAN network spans extensive geographical areas in Italy, collecting a vast volume of IoT data from 1862 end devices (EDs) and 138 gateways. In 2019, it amassed a total of 372,119,877 packets. In Figure 8, each dot represents a LoRa packet from two real devices. Suppose we aim to predict the source device of a packet based on its LoRaWAN traffic characteristics; there should be a discernible separation in some feature space. In the plots, each packet is depicted in RSSI-SNR space, two available data features, with different colors indicating the real source device. In Fig. 8a, despite a clear separation, the data is not linearly separable by a single hyperplane. Training a linear SVM on this data leads to the SVM struggling to accurately discriminate between the two classes.

To address this challenge, we can employ a kernel function on the data. Let □1 and □2 represent the two dimensions of the data; the kernel function introduces an additional dimension □3, as depicted in Figure 8b. In this new space, a linear separation of the data exists, facilitating the SVM in learning the separating hyperplane.

In our example, we opted for an ad-hoc kernel tailored to the available data. The selection of an appropriate kernel is crucial for achieving optimal performance from an SVM. However, determining the ideal kernel transformation may not always be straightforward, depending on the characteristics of the data.

SVMs have found extensive use in Intrusion Detection Systems (IDSs). Through the soft-margin learning strategy and the utilization of the kernel trick, SVM proves to be a potent and adaptable tool for Supervised Learning

systems. While SVM demands a considerable amount of computational resources during training—less than an Artificial Neural Network but more than an instance-based classifier—it requires minimal resources for predicting new values, as evidenced in Equation (1). This computational efficiency makes SVM suitable for online real-time IDS applications, minimizing added latency to communication. It also enables devices with limited computational capabilities, such as IoT devices, to perform predictions.

Various approaches involving simple linear and kernelized SVM have been applied to public datasets, demonstrating effective detection performance. For instance, authors in [83] applied basic data preprocessing steps and employed an SVM with a Radial Basis Function (RBF) kernel, where RBF(□1, □2) = exp (−1 / (2□^2)) (Equation 4), and □ is a free parameter. The high accuracy achieved with the RBF kernel highlights its effectiveness as a distance measure.

Pervez et al. [84] proposed a feature-filtering algorithm based on SVM to enhance performance on the NSL-KDD dataset predictions. The algorithm involves training an initial SVM classifier with a set of input features and iteratively modifying the feature space by removing one feature at a time based on a custom policy. The process continues until the accuracy of the new classifier surpasses the previous one, demonstrating high accuracy and recall, but limited generalization to detect new network intrusions.

Chandrasekhar et al. [85] introduced a system combining Fuzzy C-means (FCM), Artificial Neural Networks (ANNs), and SVMs. The dataset is initially partitioned into clusters via FCM, with one cluster representing nominal network traffic. ANNs are then applied per cluster to learn attack patterns, enhancing classifier performance.

KNN-based algorithms have found widespread application in Intrusion Detection Systems (IDSs), particularly due to their ability to handle classes with a substantial number of samples that contribute significantly to the classification "vote." In the context of IDSs, obtaining labeled samples is a non-trivial task, and achieving balance in labeled samples poses additional challenges. To address this, introducing a distance weight in the voting process or exploring various KNN variants, such as Large Margin Nearest Neighbor or Neighbourhood components analysis [88], becomes beneficial.

one advantageous feature of KNN, and instance-based techniques in general, for IDS purposes is their lack of a training phase. This characteristic makes KNN suitable for dynamic systems where conditions change over time. In scenarios like anomaly detection, where the working conditions may shift due to legitimate causes (e.g., seasonality), a new model for anomaly detection might be required. Unlike methods like SVMs or ANNs that necessitate retraining the entire model, building a new KNN instance merely involves feeding the algorithm labeled samples, allowing the system to be promptly operational. However, a drawback of this approach is that predictions may be relatively slower compared to other machine learning techniques, affecting the real-time nature of the IDS.

Exact KNN implementations involve a significant number of operations for a single prediction, and more advanced implementations may leverage Locality Sensitive Hashing [89] to reduce the time complexity. However, the use of hashing functions introduces an element of error and uncertainty to the IDS output, affecting its accuracy.

Another limitation of instance-based algorithms is that the predicting device needs to store the entire dataset (or, in the case of LSH, a hashmap proportional to the dataset size) instead of a trained model like a hyperplane in SVM. This limitation restricts the feasibility of running an IDS on memory-constrained devices.

Researchers have explored hybrid approaches to enhance KNN's performance. Sharifi et al. [90] proposed a hybrid method involving KNN and k-means Clustering, using Probabilistic Principal Component Analysis for data preprocessing. They partitioned the data with k-means clustering, assigning the most frequent label to each cluster, and created a KNN instance using these clusters and labels, demonstrating improved performance compared to a baseline KNN approach.

Shapoorifard et al. [92] extended the traditional KNN by considering both the closest and farthest neighbors along with cluster centers. They combined this approach with a k-farthest neighbors classifier, achieving high accuracy and recall. While effective for attack recognition, the performance of this approach in detecting anomalies in the network was not thoroughly studied.

Meng et al. [93] proposed a knowledge-based expert system to validate incoming alerts. Their multi-tier KNN filters alarms from an existing IDS, employing expert-made custom rating mechanisms. The system, integrated with Snort IDS [94], achieved high accuracy while minimizing loss in recall.

6.2 Decision Tree

A Decision Tree serves as a machine learning technique that constructs a classification tree from input data, enabling rapid predictions. Each node in the tree corresponds to a data feature, each branch represents a value of the feature, and each terminal leaf signifies a potential classification outcome. Typically, the Decision Tree model is built top-down, sequentially selecting features according to a defined policy, such as Information Gain (IG). Information Gain measures the amount of information gained on a random variable by introducing another random variable. In the realm of Decision Trees, it is expressed as the difference in entropy between the prior knowledge and itself with the value of the attribute □ given as known:

\[IG(□ , □□) = H(□) − H(□ |□□) \]

Here, \(H \) represents Shannon’s entropy, and \(H_□ \) is the conditional entropy. Information Gain effectively ranks features, guiding the tree to prioritize features with high discriminative potential. Prominent algorithms for Decision Tree construction, such as ID3, C4.5, and CART [95], rely on Information Gain when creating new tree

nodes. Regardless of the algorithm, the prediction process involves traversing the tree from the root node, selecting branches using if-then-else operations on input features. This computational efficiency makes Decision Tree prediction feasible without requiring specialized hardware.

However, the primary challenge with Decision Trees is the risk of overfitting to the training data. This issue is more pronounced in Decision Trees due to their independent comparison of features without considering associations. To mitigate overfitting, techniques like post-pruning can be employed. Post-pruning involves removing internal nodes from a tree to enhance model generalization by eliminating training set-specific checks. Determining an optimal pruning threshold is non-trivial, and strategies like alternating pruning and evaluation on a test set may introduce bias. A more unbiased approach involves partitioning the dataset into multiple sets, using k-fold cross-validation.

Despite being widely used in existing IDSs due to their simplicity and rapid build and prediction speeds, Decision Trees face challenges. The demand for ample training data to avoid overfitting is particularly challenging in the context of IDSs, where labeling data manually is arduous. Additionally, modeling complex relationships between features proves difficult, as each feature is treated independently during the prediction phase. This limitation can lead to misclassifications of certain inputs, and one potential solution is to preprocess the data through Principal Component Analysis (PCA).

An important limitation concerning Decision Trees in IDS applications is their inability to identify patterns extending across different data points. Decision Trees treat each input independently, lacking an internal state of previous predictions. Consequently, IDSs relying solely on Decision Trees struggle to detect patterns in the exchanged packets among hosts, hindering their ability to identify a broad range of potentially malicious messages. This limitation makes Decision Trees less suited for identifying zero-day attacks and performing anomaly detection, although some literature explores enhancing Decision Trees with other ML techniques [97], [98], [99].

Various approaches have been proposed to address these challenges. Malik et al. [100] suggested an IDS based on Decision Trees, incorporating Particle Swarm Optimization (PSO) for tree pruning. Rai et al. [101] developed a technique to build a Decision Tree-based IDS, addressing split value and feature selection issues. Azad et al. [102] proposed a hybrid IDS combining Decision Trees and Genetic Algorithm (GA) to address the Small Disjoint problem, which arises when nodes close to the leafs discriminate only a small number of instances, leading to overfitting.

6.3 Artificial Neural Network

Artificial Neural Networks (ANNs) are mathematical constructs capable of approximating any continuous function \(□\) to \(□\). Typically, Neural Networks consist of multiple neurons organized in layers. Each neuron

takes outputs from all neurons in the preceding layer and performs the operation \(□(□) = □ (∑ □□□□ + □)\) (Equation 6). The parameters of these neurons are embedded in kernels that execute convolutions over the input. While multiple kernels are usually applied in a single layer, the number of parameters is significantly lower compared to a fully-connected network. This results in improved performance during both training and prediction phases. Additionally, parameter sharing allows the network to learn the target function, rendering it invariant to deformations like translation, scaling, or tilting—particularly advantageous in tasks involving image or text recognition.

Mathematical theorems falling under the category of Universal Approximation Theorems [104] affirm the capability of a neural network to approximate a continuous function. One notable theorem is Cybenko's theorem [105], asserting that any function can be approximated by a fully-connected neural network with a single hidden layer.

Neural Networks, particularly Convolutional Neural Networks (CNNs), have found application in Intrusion Detection Systems (IDSs) for the identification of attacks within network traffic. Wang et al. [109] introduced a CNN-based approach to detect malicious network traffic. They devised an encoding scheme to transform data samples into one-channel 2D images, enhancing CNN generalization without relying on hand-crafted or application-specific knowledge. This encoding is applicable to raw packets in any IDS. While their system achieves reasonable accuracy, there is room for further improvement. In another work, Wang et al. [110] proposed a 1D CNN for classifying encrypted traffic. The capability to perform machine learning operations on encrypted traffic allows the IDS to be deployed on another network of the target system, where incoming traffic is predominantly encrypted due to TLS or other encryption schemes. The challenges of deploying an IDS in Edge environments will be discussed in Sections 3 and 5.

Despite CNNs having a higher number of trainable parameters than other machine learning techniques, making training computationally intensive, they are usually trained using specialized hardware such as high-end GPUs capable of parallel floating-point calculations. However, post-training, the network can make predictions using only the CPU. In the context of IDSs, this implies that a network can be trained offline, even off-site (e.g., using a cluster of GPUs remotely from a cloud provider), without the need for specialized hardware installation within the targeted system.

While CNNs have been widely employed for attack recognition, their utilization for detecting network anomalies has been limited [111]. In theory, it might be possible to train a CNN on labeled data with "normal" and "anomaly" labels, but the diverse range of anomaly attacks necessitates an extensive training set that covers all possible anomalies detectable by the system. Section 6.4.1 will explore the conventional approach to anomaly detection with a neural network, involving the use of an Auto Encoder—a specific network architecture that simplifies one-class classification tasks.

6.4 Auto Encoder

Autoencoders (AEs) represent a distinctive architecture within Artificial Neural Networks (ANNs), adept at learning an encoding of input data through unsupervised learning. This encoding serves purposes such as dimensionality reduction, noise cancellation, and anomaly detection. Comprising two neural networks, namely the encoder and the decoder, AEs operate in an unsupervised manner. The encoder, denoted as \(E: \mathbb{R}^m \rightarrow \mathbb{R}^n\), learns an encoding \(D: \mathbb{R}^n \rightarrow \mathbb{R}^m\) such that \(||□ - D(E(□))||_2^2\) is minimized.

Various types of AEs have been developed, each offering specific advantages:

- Variational Autoencoder [113], introducing continuity to the encoding by pushing the space to follow a normal distribution through an additional term in the reconstruction loss.

- Denoising Autoencoder [114], learning to reconstruct a sample from a corrupted input during the stochastic process \(□ˆ\sim □(□|□)\), enabling the model to denoise corrupted inputs.

- Sparse Autoencoder [115], featuring a bottleneck with more neurons but activating only a few, depending on the input, due to a penalization term in the loss function. This sparse encoding is reported to enhance performance in classification tasks.

Anomaly detection, a one-class classification task, involves training a machine learning model to predict whether an input belongs to a specific class. AEs are commonly employed for this purpose. The typical strategy involves training the model solely on non-anomalous samples. During deployment, real samples are encoded and decoded, and when the reconstruction error surpasses a predefined threshold, an anomaly alert is triggered.

Vanilla AEs may lack predictability due to system variability, making variants like Variational AEs more suitable. Setting the threshold precisely is non-trivial, often depending on the system's rate of change. If system conditions vary significantly, training the AE from scratch may be necessary, introducing challenges associated with heavy-weight ANN-based systems.

The unsupervised training of an IDS presents a significant advantage, as it eliminates the need for manually labeling the dataset. Normal data samples can be automatically collected during intrusion-free periods, assuming the use of another IDS for validation. Abolhasanzadeh et al. [121] proposed a system to enhance IDS detection performance, utilizing the encoding of a vanilla AE for dimensionality reduction. They achieved a notable reduction in prediction latency, a concern for ANN-based IDSs. Their AE-based dimensionality reduction outperformed PCA, factor analysis, and non-linear kernelized PCA [122] in terms of accuracy.

In another approach, Aminanto et al. [123] introduced an AE-based IDS designed specifically for detecting Wi-Fi impersonation attacks. They employed stacked Autoencoders, a subclass with multiple encoding layers, for feature

extraction. ML techniques like SVM, ANN, and Decision Tree were then applied for feature weighting, and the final classification was performed through an ANN.

Conclusion

This study explores Intrusion Detection Systems (IDS) for the Internet of Things (IoT) from both an architectural standpoint and the methodologies employed to detect anomalies and cyber attacks. In terms of architecture, while conventional IoT IDS are typically deployed at the device or gateway level, the growing interest in edge computing introduces new vulnerabilities that malicious actors can exploit. We delve into these emerging issues and present solutions designed to address them. The discussion then shifts to new IDSs specifically tailored for edge environments. Subsequently, we focus on the integration of Machine Learning (ML) techniques within IDS frameworks. For each ML technique, we provide a theoretical overview and discuss the anticipated advantages and disadvantages. The study also highlights the computational power, storage capacity, and real-time response requirements associated with each technique, determining their suitability for an edge-oriented IDS.

REFERENCES

[1]Anukool Lakhina, Mark Crovella, and Christophe Diot. "Diagnosing network-wide traffic anomalies". In: ***ACM SIGCOMM Com- puter Communication Review*** **34.4 (Oct. 2004),**

p. 219. DOI: 10.1145/1030194.1015492.

[2] A. L. Buczak and E. Guven. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection". In: ***IEEE Communications Surveys Tutorials*** **18.2 (2016),**
pp. 1153–1176. DOI: 10 . 1109 / COMST . 2015 . 2494502.

[3] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani. "A Survey of Ma- chine and Deep Learning Methods for Internet of Things (IoT) Security". In: ***IEEE Communi- cations Surveys Tutorials*** **22.3 (2020), pp. 1646–**
1685.

[4] Ansam Khraisat, Iqbal Gondal, and Peter Vam- plew. "An Anomaly Intrusion Detection System Using C5 Decision Tree Classifier". In: ***Lecture Notes in Computer Science*****. Springer Interna- tional Publishing, 2018, pp. 149–155. DOI: 10. 1007/978-3-030-04503-6_14.**

[5] Christian Kreibich and Jon Crowcroft. "Honey- comb - Creating Intrusion Detection Signatures Using Honeypots". In: ***Computer Communication Review*** **34 (Jan. 2004), pp. 51–56.**

[6] Niels Provos. "A Virtual Honeypot Frame- work". In: ***Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*****. SSYM'04. San Diego, CA: USENIX Association, 2004, p. 1.**

[7] Chad Meiners, Jignesh Patel, Eric Norige, Eric Torng, and Alex Liu. "Fast Regular Expression Matching Using Small TCAMs for Network In- trusion Detection and Prevention Systems." In: Sept. 2010, pp. 111–126.

[8] Po-Ching Lin, Ying-Dar Lin, and Yuan-Cheng Lai. "A Hybrid Algorithm of Backward Hashing and Automaton Tracking for Virus Scanning". In: *IEEE Transactions on Computers* 60 (2011),

pp. 594–601.

[9] Nazim Uddin Sheikh, Hasina Rahman, Shashwat Vikram, and Hamed AlQahtani. "A Lightweight Signature-Based IDS for IoT Environment". In: *ArXiv e-prints* (2018).

[10] Symantec. *Internet security threat report 2017*.

[11] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. "Survey of intrusion detection systems: techniques, datasets and chal- lenges". In: *Cybersecurity* 2.1 (July 2019). DOI: 10.1186/s42400-019-0038-7.

[12] Hung-Jen Liao, Chun-Hung Richard Lin, Ying- Chih Lin, and Kuang-Yuan Tung. "Intrusion detection system: A comprehensive review". In: *Journal of Network and Computer Applications* 36.1 (Jan. 2013), pp. 16–24. DOI: 10.1016/j. jnca.2012.09.004.

[13] N. Ye, S. M. Emran, Q. Chen, and S. Vilbert. "Multivariate statistical analysis of audit trails for host-based intrusion detection". In: *IEEE Transactions on Computers* 51.7 (2002), pp. 810– 820.

[14] Zhiyuan Tan, Aruna Jamdagni, Xiangjian He, Priyadarsi Nanda, and Ren Ping Liu. "A Sys- tem for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis". In: *IEEE Transactions on Parallel and Distributed Systems* 25.2 (Feb. 2014), pp. 447–456. DOI: 10 .1109 / tpds.2013.146.

[15] Roberto Magán-Carrión, José Camacho, Gabriel Maciá-Fernández, and Angel Ruíz-Zafra. "Multivariate Statistical Network Monitoring–Sen- sor: An effective tool for real-time monitor- ing and anomaly detection in complex net- works and systems". In: *International Jour- nal of Distributed Sensor Networks* 16.5 (May 2020), p. 155014772092130. DOI: 10 . 1177 / 1550147720921309.

[16] V. Chatzigiannakis, S. Papavassiliou, and G. An- droulidakis. "Improving network anomaly detection effectiveness via an integrated multi-metric- multi-link (M3L) PCA-based approach". In: *Se- curity and Communication Networks* 2.3 (May 2009), pp. 289–304. DOI: 10.1002/sec.69.

[17] D. Brauckhoff, K. Salamatian, and M. May. "Ap- plying PCA for Traffic Anomaly Detection: Problems and Solutions". In: *IEEE INFOCOM 2009 - The 28th Conference on Computer Communica- tions*. IEEE, Apr. 2009. DOI: 10.1109/infcom. 2009.5062248.

[18] Haakon Ringberg, Augustin Soule, Jennifer Rex- ford, and Christophe Diot. "Sensitivity of PCA for traffic anomaly detection". In: *Proceedings of the 2007 ACM SIGMETRICS international con- ference on Measurement and modeling of com- puter systems - SIGMETRICS '07*. ACM Press, 2007. DOI: 10.1145/1254882.1254895.

[19] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. "A review on out- lier/anomaly detection in time series data". In: *ArXiv e-prints* (2020). eprint: 2002.04236.

[20] Kieran Flanagan, Enda Fallon, Paul Connolly, and Abir Awad. "Network anomaly detection in time series using distance based outlier de- tection with cluster density analysis". In: *2017 Internet Technologies and Applications (ITA)*. IEEE, Sept. 2017. DOI: 10.1109/itecha.2017. 8101921.

[21] Jouni Viinikka, Hervé Debar, Ludovic Mé, Anssi Lehikoinen, and Mika Tarvainen. "Processing in- trusion detection alert aggregates with time se- ries modeling". In: *Information Fusion* 10.4 (Oct. 2009), pp. 312–324. DOI: 10.1016/j.inffus. 2009.01.003.

[22] Qingtao Wu and Zhiqing Shao. "Network Anomaly Detection Using Time Series Analy- sis". In: *Joint International Conference on Au- tonomic and Autonomous Systems and Interna- tional Conference on Networking and Services - (icas-isns'05)*. 2005, pp. 42–42.

[23] Ying Zhao, Junjun Chen, Di Wu, Jian Teng, Nabin Sharma, Atul Sajjanhar, and Michael Blu- menstein. "Network Anomaly Detection by Using a Time-Decay Closed Frequent Pattern". In: *In- formation* 10 (Aug. 2019), p. 262. DOI: 10.3390/ info10080262.

[24] Aarthi Reddy, Meredith Ordway-West, Melissa Lee, Matt Dugan, Joshua Whitney, Ronen Kahana, Brad Ford, Johan Muedsam, Austin Henslee, and Max Rao. "Using Gaussian Mix- ture Models to Detect Outliers in Seasonal Uni- variate Network Traffic". In: *2017 IEEE Secu- rity and Privacy Workshops (SPW)*. IEEE, May 2017. DOI: 10.1109/spw.2017.9.

[25] Neil Walkinshaw, Ramsay Taylor, and John Der- rick. "Inferring extended finite state machine models from software executions". In: *Empirical Software Engineering* 21.3 (Mar. 2015), pp. 811–853. DOI: 10.1007/s10664-015-9367-7.

[26] Youssef Laarouchi, Mohamed Kaaniche, Vincent Nicomette, Ivan Studnia, and Eric Alata. "A language-based intrusion detection approach for automotive embedded networks". In: *Interna- tional Journal of Embedded Systems* 10 (Jan. 2018), p. 1. DOI: 10.1504/IJES.2018.10010488.

[27] Idan Amit, John Matherly, William Hewlett, Zhi Xu, Yinnon Meshi, and Yigal Weinberger. "Machine Learning in Cyber-Security -Problems, Challenges and Data Sets". In: *ArXiv e-prints* (Feb. 2019).

[28] Sumeet Dua and Xian Du. *Data Mining and Ma- chine Learning in Cybersecurity*. Auerbach Pub- lications, Apr. 2016. DOI: 10.1201/b10867.

[29] Yang Xin, Lingshuang Kong, Zhi Liu, Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao, Haixia Hou, and Chunhua Wang. "Machine Learning and Deep Learning Methods for Cyber- security". In: *IEEE Access* 6 (2018), pp. 35365–

35381. DOI: 10.1109/access.2018.2836950.

[30] Arwa Aldweesh, Abdelouahid Derhab, and Ahmed Z. Emam. "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues". In: *Knowledge-Based Systems* 189 (Feb. 2020),

p. 105124. DOI: 10 . 1016 / j . knosys . 2019 . 105124.