IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 December 2023, accepted 18 December 2023, date of publication 28 December 2023,
date of current version 16 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3347332

SURVEY

P4 Cybersecurity Solutions: Taxonomy and Open Challenges

CHRISTIAN GARZÓN, SANTIAGO RÍOS-GUIRAL, ERWIN LEAL, SERGIO A. GUTIÉRREZ, AND JUAN F. BOTERO, (Member, IEEE)
Faculty of Engineering, Universidad de Antioquia, Medellín 050010, Colombia

Corresponding author: Christian Garzón (christianc.garzon@udea.edu.co)

This work was supported in part by the Ibero-American Program on Science and Technology for Development through the CYTED Project 519RT0580, and in part by the General System of Royalties from Colombia BPIN code 2020000100381.

ABSTRACT The field of cybersecurity has witnessed a significant shift towards Programmable Data Planes with the emergence of the P4 programming P4. The existing literature lacks a comprehensive taxonomy that provides collaborative classifications and characterization of P4-based cybersecurity solutions. In this paper, we propose a novel taxonomy to better characterize published works in the field of cybersecurity solutions developed leveraging P4 and Programmable Data Planes. Our taxonomy introduces three main categories: detection techniques, mitigation actions, and deployment platforms. Unlike existing classification approaches present in literature, our taxonomy allows categorizing a given work according to different criteria, thus enabling collaborative classifications and the identification of shared features by highlighting intra-category and inter-category relationships that can be established among different works. Through our comprehensive characterization of the works identified in the literature, we present key findings that contribute to extend the current understanding of the field. By identifying the diverse range of techniques employed and the platforms utilized, we aim at addressing the needs to understand the area, which is fundamental for future advancements. Notably, we emphasize the significance of security provisioning through the adoption of cybersecurity functional abstractions. These abstractions enable the emulation of the behavior of conventional security devices, leveraging the capabilities of P4. Furthermore, we discuss several research challenges and possible future directions that have emerged from our analysis. This paper focuses on providing valuable insights and knowledge for researchers and practitioners in the field of cybersecurity solutions based on Programmable Data Planes leveraging P4, with the final goal of opening avenues for new research.

INDEX TERMS Detection, mitigation, security provisioning, machine learning, programmable data plane, P4 programming language.

I. INTRODUCTION

Software-Defined Networking (SDN) is a revolutionary networking paradigm that has introduced essential changes in network management by empowering a programmable network approach for different management and operation tasks. SDN is based on core concepts such as separation of Control and Data planes, network programmability, flow-based traffic management, and a logically centralized network view. These concepts have enabled the development of novel solutions and the revision of existing approaches to tackle different network challenges [1]. For instance, network security has been extensively revisited in the light of SDN. Traditional cybersecurity solutions usually increase communication delays and rely on traffic duplicity, and they might even introduce congestion issues or data loss [2], [3]. Hence, one aspect that can be improved is the response time. This improvement is especially relevant in contexts such as network infrastructures in industrial areas where the risk of loss or duplication of packets when they are sent to a middle-box specialized in cybersecurity can increase delays in time-sensitive systems, which for some use cases is unacceptable [4].

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

As a natural evolution and complete realization of the SDN paradigm, Programmable Data Planes (PDPs) extend the landscape for the development of network management solutions. Different proposals in literature [5], [6], [7], [8], [9] introduce the concept of programmable pipelines that can be deployed at data plane devices. These programmable pipelines make it possible to offload operations associated with traffic processing from the logically centralized control plane toward devices in the data plane. Moreover, the programmable nature of data planes makes it possible to implement functionalities with extended scope and possibilities that can be leveraged to develop sophisticated solutions in addition to those provided in traditional networks. In particular, network security can take advantage of network programmability, for instance, by partially offloading traffic classification and anomaly detection to the data plane.

Currently, Programming Protocol-independent Packet Processor (P4), is the *de facto* standard language for data plane programmability [10], [11], and it is based on the concept of programmable switching architectures [12]. These architectures allow the defining of custom packet formats and parsing sequences within the forwarding devices. Thus, by leveraging this customization, operations of traffic classification or even Machine Learning (ML) models can be deployed in the data plane, which might increase the robustness, scalability, and efficiency of network solutions.

Given the relevance of network security and the interest of industry and academia in developing solutions based on PDPs, in this work, we present a literature survey that offers three distinct contributions that distinguish it from previous works.

- First, we propose a broader, timely, and relevant taxonomy of the solutions available in the literature that address different aspects of network security using PDP: (i) detection techniques, (ii) mitigation actions, and (iii) experimental platforms. Particularly, we address the categorization of solutions based on mitigation, which is a category not frequently considered in the literature. In addition, our taxonomy develops an inter-domain categorization approach rather than the intra-domain approach commonly used in other literature surveys.
- Second, we highlight the potential of P4 to enable the creation and adaptation of specific cybersecurity middle-box functions, which can be deployed in the data plane. Leveraging the capabilities of PDP provides the opportunity to substitute conventional cybersecurity devices with the mentioned adapted functions while maintaining the efficiency of PDP. Additionally, in this contribution, we emphasize the concept of Security Provisioning, aiming to implement proactive cybersecurity measures.
- Third, we present a summary of possible emerging research branches and perceptions of future trends in the field of cybersecurity solutions leveraging PDPs. Our discussion encompasses development limitations, a future view in the deployment of machine learning in data planes, the path to achieve in-network computing and self-driving networks, the improvement of mitigation actions, and finally, intrinsic PDP security challenges.

The remainder of this paper is organized as follows. Section II gives a brief overview of the P4 programming language in a simple way, including fundamental topics such as the program, the compiler, the switch reference architecture, and the P4Runtime as the basic southbound interface. Section III focuses on previous works that present reviews of cybersecurity solutions utilizing P4 and their corresponding taxonomies. Section IV describes our proposed taxonomy, which summarizes the literature review based on our cross-categorical approach. Our taxonomy encompasses detection techniques, mitigation actions, and the experimental platforms used to deploy P4 cybersecurity solutions. Subsequently, Section V emphasizes the importance of proactive responses to threats and highlights the utilization of P4 capabilities for emulating the behavior of recognized security devices. Section VI discusses emerging research branches and future challenges in network security that leverage PDPs based on the findings obtained from this work. Finally, Section VII concludes the paper.

II. P4 IN A NUTSHELL

Traditional networks integrate control and data planes in the same device. This mode of operation became difficult to manage and limited the possibilities for innovation [1], [13]. Consequently, SDN emerged as a paradigm that makes the network dynamic, flexible, and programmable by decoupling the control and data planes. Despite SDN bringing new functionalities and benefits, it also suffers from scalability and performance issues, partly due to the lack of intelligence in forwarding devices [12]. The search for higher performance has motivated the introduction of new approaches, such as PDP, that introduce a promise of better performance when compared to fixed-function switches while introducing additional flexibility for the implementation of solutions [14].

The PDP concept refers to the ability to customize the packet processing behavior of network devices by exposing low-level packet processing logic through higher abstractions, such as standardized Application Programming Interfaces (APIs) [15]. In this matter, PDP enables the creation of custom network functions tailored to fulfill specific requirements defined by network managers. Different platforms can be used to deploy programmable data planes. Some of these platforms are Application-Specific Integrated Circuits (ASICs) [16], [17], the Software-based Behavioral Model version 2 (BMv2) [7], Field-programmable Gate Arrays (FPGAs) [8], and network processors. These platforms offer high performance due to the dedicated and specialized components dealing with network management tasks.

There is a diverse list of data plane programming solutions for deploying PDPs. Some of these solutions are: Data Plane Development Kit (DPDK) [5], eXpress Data Path

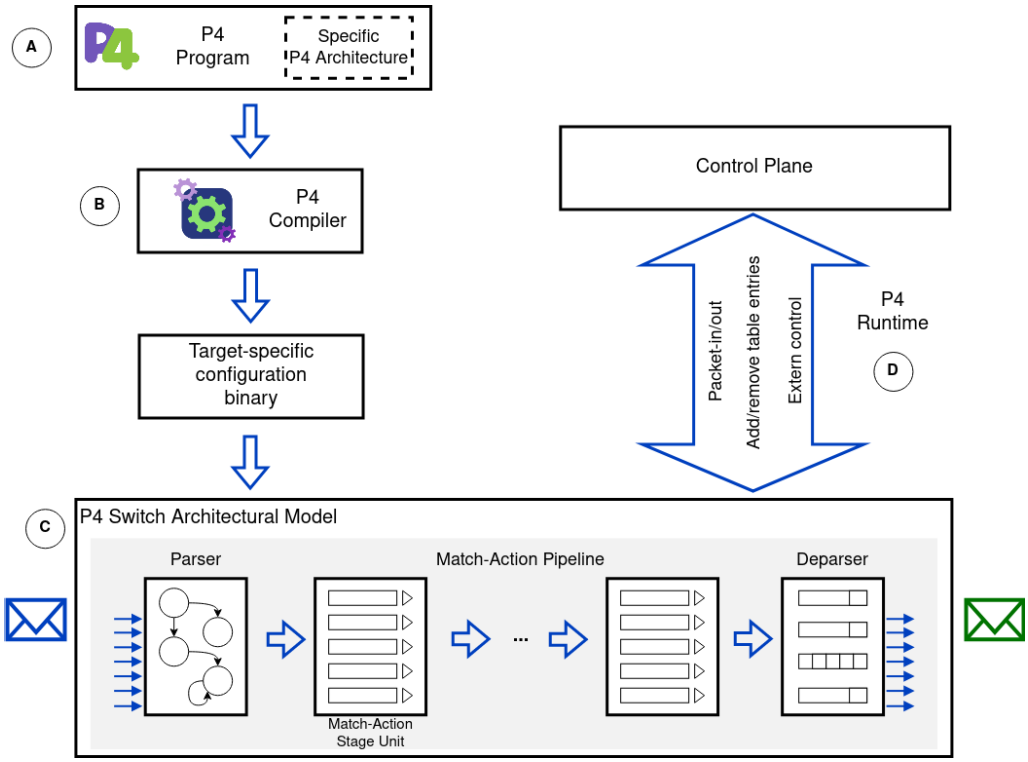

FIGURE 1. Workflow model of the P4 programming language.

(XDP) [18], and P4 [10]. Among these solutions, the P4 programming language has emerged as the most prominent data plane programming language due to its versatility, abstraction level, pipeline usage, similarity with the C programming language, and a broad range of built-in data plane functions, which have captured the attention of academia and industry due to the success of different proposals which leverage it [19].

P4 allows specifying both the configuration of data plane devices and how these devices process packets, thus providing a higher level of abstraction in network programming. The design of P4 has three main goals:

I. **Reconfigurability**: Controllers can redefine in the field the mechanisms for packet parsing and processing.
II. **Protocol independence**: A high-level abstraction that enables protocol-agnostic designs, allowing packet processing regardless of the specific formats or headers of these packets. The ultimate goal is to support a wide range of protocols while adapting to evolving network needs through customizable and flexible processing.
III. **Target independence**: Programmers can describe packet processing functions without the need to know the specifics of the underlying hardware.

Initially, P4 was developed to program software switches using the v1model architecture [7]. However, its applicability has expanded to cover a diverse range of physical data plane devices such as: programmable Network Interface Cards (NICs) based on the Portable NIC Architecture (PNA) [20], switches supporting the Portable Switch Architecture (PSA) [21], and specialized switch brands such as Barefoot Networks and Intel equipped with Tofino Native Architecture (TNA) [22].

The P4 working group has proposed two standards: $P4_{14}$ and $P4_{16}$. The last one introduces new capabilities in response to the shortcomings of the previous version. Some of these new capabilities are strict types, expressions, nested data structures, and support for multiple targets and pipeline architectures. Moreover, $P4_{16}$ provides several base types as well as type operators, different arithmetic algorithms, and external constructs for functions not defined within the P4 core [23]. In [24], the P4 working group has compiled the specifications for the P4 programming language, P4Runtime, and different target architectures. Furthermore, the $P4_{16}$ language specification for the latest version is available at [25] for further information.

The operation and development of functionalities on P4-based switches follow a very well-defined workflow depicted in Figure 1. This workflow integrates four main components: the P4 program itself, the P4 compiler, the architectural reference model of the switch where the program is intended to run, and the P4Runtime interface as a mechanism to communicate control and data planes. In the following subsections, we describe these components in detail.

A. P4 PROGRAM

A P4 program (depicted in Figure 1 - A) defines the logic of the packet processing, which expresses the intended behavior

of the data plane devices. P4 programs are developed for specific P4 architectural models, such as v1model, PNA, PSA, and TNA. A P4 program has five main blocks: Headers, Parsers, Tables, Actions, and the Processing Pipeline. The Header component defines the format of the packets that the P4-enabled switch will be able to recognize. In other words, it specifies the packet fields, their length, and the expected order of these fields within the packet. The parser component, which resembles a finite state machine, defines how packet headers are processed. It specifies the order of the packet header processing and the decisions made according to particular values of these headers. Tables and actions blocks implement the actual decision-making process on the packet flow. Table entries determine which packet header values or variables can be matched to determine which action must be applied to the corresponding packet. Finally, the processing pipeline component integrates the previous elements and determines the order and logic of the operations to be applied to the packets flowing through the switch [23].

B. P4 COMPILER

The P4 Compiler (depicted in Figure 1 - B) is a software tool that translates the P4 code into machine code specific to a target device. The P4 compiler has two main functions: (i) Generating an executable file for the target device in the data plane, indicating the header format and the corresponding operations translated into primitive instructions for the corresponding hardware (i.e. machine code); and (ii) Generating a run-time mapping to allow control and data plane to communicate with each other through the P4Runtime interface. P4-enabled device manufacturers design and provide the P4 compiler for specific P4 target devices.

C. P4 SWITCH ARCHITECTURAL MODEL

One of the goals of the P4 language is to be target-independent, meaning that programmers can describe packet processing functions without knowing the details of the underlying hardware [26]. The P4 target (depicted in Figure 1 - C) has a packet processing pipeline whose structure is target-specific and it corresponds to a particular architectural model. Target independence is achieved by leveraging the notion of the Portable Switch Architecture, which is an open-source framework providing a common interface and abstraction for programming network switches. PSA defines control blocks divided into three components:

I. The parser, which specifies the sequence to analyze different headers in a packet. It resembles the notion of a finite state machine with a state corresponding to a given header and a transition corresponding to given values of a field within the header.
II. The Match/Action block, which contains multiple Match/Action tables used to match packet header parameters according to different user-provided values and defines actions to take whenever a match occurs.
III. The Deparser, which reassembles the headers into packets in regular order and sends them out to the next step, either a queue or an output port.

III. RELATED WORK

The integration between SDN control planes and P4-programmable devices is a hot research topic that has enabled the development of a wide landscape of solutions for different computer network domains. In the cybersecurity area, there is a combined effort between academia and industry in order to develop solutions capable of protecting networks against a growing number of cyberattacks. In that matter, there is a current trend aiming at the development of P4-based cybersecurity solutions that try to deal with more sophisticated attacks. Moreover, by leveraging P4, it is possible to take advantage of network programmability to deploy cybersecurity solutions directly in the data plane or through the collaboration between control and data planes.

In this survey, we propose a taxonomy of the reviewed works based on the exploration of research proposals centered around network security using network programmability. The following articles reviewing cybersecurity solutions based on P4 were handy for the construction of our taxonomy.

Kfoury et al. [14] present an exhaustive survey on P4-enabled programmable data plane switches. This paper analyzes more than 150 articles related to P4-based solutions such as in-band network telemetry, network performance, middlebox functions, accelerated computations, Internet of Things (IoT), cybersecurity, and testing, among others. From these articles, 48 papers focus on the cybersecurity research domain. The classification provided by these authors categorizes P4-based programmable solutions into seven domains covering the most significant research areas. For the cybersecurity domain, the authors propose a taxonomy that classifies P4-based cybersecurity solutions into five security-related categories. The aforementioned categories are heavy-hitter detection, cryptography, anonymity, access control, and miscellaneous defense solutions.

In Gao and Wang [23], authors propose a literature survey addressing solution proposals that leverage P4-based programmable data planes for network security problems. The work is divided into two main sections: first, the authors describe the P4 workflow depicted in Figure 1 while presenting the advantages of P4-based programmable data planes to cope with network security issues. In the second section, the authors provide a classification of P4-based cybersecurity solutions. This review classifies these solutions into four categories according to the perspective of proactive and reactive defense and the combination of multiple enabling technologies. The four network security categories in this review are: access control, privacy and encryption, availability, and integrated defense. These categories define control over incoming traffic, privacy protection, network availability, and defense improvement. Authors present solutions such as Firewalls, cryptography, techniques against Distributed Denial of Service (DDoS) attacks, and ML approaches, among others.

In AlSabeh et al. [27], authors present a survey on cybersecurity solutions deployed in P4-programmable switches. In this survey, authors provide a taxonomy that classifies

TABLE 1. Comparison of different literature surveys on cybersecurity solutions leveraging P4.

Reference	Taxonomy		Defense mechanisms		Cybersecurity Oriented	Future work & Challenges
	Intra-Category	Inter-category	Detection	Mitigation		
Kfoury et al. [14]	●	○	●	○	○	●
Gao et al. [23]	●	○	●	○	●	○
AlSabeh et al. [27]	●	○	●	○	●	●
Kianpisheh et al. [28]	●	○	●	○	○	●
Hauser et al. [29]	●	○	●	○	○	○
This paper	●	●	●	●	●	●

reviewed solutions according to three network security objectives: network availability, anonymity and confidentiality, and operational security provisioning. The categories express the required network goals regarding security, ranging from prevention mechanisms to the deployment of detection and mitigation mechanisms.

In Hauser et al. [29], authors provide a comprehensive survey on data plane programming leveraging P4. The article presents a general overview of P4 by providing information about data plane programming, architectures, compilers, targets, and data plane APIs. The paper explains the evolution of data plane programmability through P4 while summarizing research efforts in order to advance general network solutions taking advantage of P4. The authors provide a concise review of network domains such as: monitoring, traffic management and congestion control, routing and forwarding, advanced networking, and network security, among other topics. In the network security domain, the survey presents different P4-based solutions, namely Firewalls, port knocking, DDoS defense systems, Intrusion Detection Systems (IDSs), and connection security, among other solutions.

Similarly, Kianpisheh et al. [28] present a comprehensive survey on in-network computing using programmable networks. The authors have proposed a systematic classification framework to categorize in-network solutions developed with P4 into different network domains. These domains are: analytics, caching, security, coordination, and new technologies. Regarding in-network security, the study has categorized security solutions into three categories: DDoS defense solutions, Firewalls, and miscellaneous security solutions developed with P4. The authors mainly focus on the analysis of mechanisms for the detection of DDoS. Additionally, the paper introduces P4-based security solutions that use ML and Blockchain, among other technologies.

The previously mentioned surveys offer an in-depth overview of the advancements in P4-based solutions for enhancing network security. In [14], [23], and [27], authors provide taxonomies according to the design and implementation principles that become building blocks of secure systems. These taxonomies classify P4 solutions based on the CIA triad (confidentiality, integrity, availability) [30]. In Hauser et al. [29], authors categorize P4-based security solutions by defining the domains of the taxonomy according to the type of cyberattacks to defend against, and defense applications such as IDS and Firewalls. Likewise, Kianpisheh et al. [28] present a classification for in-network P4-based security solutions in the context of defense solutions against DDoS attacks and implementation of Firewalls.

Nevertheless, the reviewed taxonomies divide the security solutions leveraging P4 into isolated domains. The classification criteria used in these taxonomies divide the solutions into categories based on common research domains. Moreover, some of the considered P4-based security solutions lacking common properties are simply grouped under generic domains like ''other/miscellaneous security solutions''. Consequently, the taxonomy domains are limited since they just apply hierarchical classifications that do not allow determining interceptions between security domains sharing properties or defense objectives.

From the aforementioned surveys, in [23] and [27], the focus is network security leveraging the P4 programming language for the implementation of security solutions. On the other hand, the remaining papers [14], [28], [29] focus on a more general overview of P4. In the security domain, [14], [28], [29] offer a summarized overview of P4-based security solutions leveraging network programmability. Additionally, the reviewed surveys provide an analysis of emerging topics related to P4 and security solutions. Hence, we consider it important to include how the future will look in terms of the development and implementation of security solutions leveraging P4.

Table 1 provides a comparison between the main focus of the aforementioned surveys and the contributions from our work. As can be seen, previous work focuses on the classification of P4-based cybersecurity proposals in intra-domain categories, meaning that the authors of these surveys classify papers according to the identification of common domains or topics. These domains divide cybersecurity solutions, grouping them into entities within a specific cybersecurity area. For instance, domains might be associated with the CIA triad grouping different cybersecurity solutions developed with P4. This division might indicate constraints

on cybersecurity solutions, preventing them from belonging to multiple research domains sharing common properties by leveraging the capabilities of P4. In contrast, in this paper, we propose a taxonomy that classifies cybersecurity solutions leveraging P4 and PDPs into intra-domain and inter-domain categories. This approach enables the classification of P4-based solutions based on specific research domains, such as detection or mitigation, and simultaneously, through inter-domain connections, P4 cybersecurity solutions can belong to different cybersecurity domains by sharing common properties. Our proposed taxonomy is presented in the following section.

In addition, it is important to mention that in our literature review, we identified that most of the works focus on the development of detection strategies, and they leave mitigation in second place. Mitigation actions are paramount to alleviating the network, reducing the impact of attacks, safeguarding private information, and preventing further attacks. In that matter, the implementation of appropriate mitigation actions might enable a better reaction to incoming attacks. Our paper covers the study of the implementation of mitigation actions leveraging the P4 programming language, in addition to those covering detection, since we consider these two approaches to be vital in the development of security solutions. Finally, through our work, we provide a novel state-of-the-art of security solutions leveraging P4 while providing an updated overview of future work and challenges for the development of P4 cybersecurity solutions.

IV. TAXONOMY OF CYBERSECURITY IMPLEMENTED PDPs

The findings presented in the previous section lead us to propose a new approach to condense the categorization of P4-based cybersecurity solutions in a simpler and more effective way.

As shown in Figure 2, we propose a taxonomy to classify the different P4-based cybersecurity proposals taking into account **detection** and **mitigation** main categories (inter-domain classification) and their different techniques (intra-domain classification). In our taxonomy, we also propose another main category, **platform**, where we list the different data plane programmable target devices used to implement and deploy cybersecurity solutions.

The following subsections will discuss each category, techniques, actions, and platforms shown in Figure 2. Finally, Tables 3, 4, and 5 will comprise all the solutions studied, with their respective detection technique, mitigation, and deployment platform.

A. DETECTION

Detection aims at identifying any malicious activity that might compromise the network [31]. However, the mentioned malicious activity has the potential to be camouflaged with legitimate intrusion attempts directed towards its ultimate target. Attackers may leverage intrusion attempts to gain unauthorized access, manipulate, or compromise

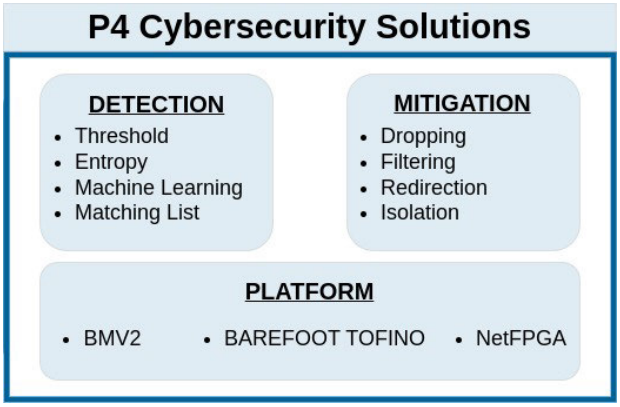

FIGURE 2. Proposed taxonomy to categorize cybersecurity solutions implemented with P4.

specific information [32]. Effectively defending against such intrusions necessitates a critical step involving the analysis of the most relevant activity, whether at end devices or across the network. This can be achieved through the deployment of host-based IDS (HIDS) or network-based IDS (NIDS) [33]. NIDS, in particular, offers significant advantages in cybersecurity applications, encompassing two key aspects: (i) thorough inspection of each desired network packet and (ii) the absence of a requirement for installation on each host. Moreover, the characteristics of PDP address the main drawback of traditional NIDS, as proposed by Khraisat et al. [34], including: (i) Dedicated software; PDP facilitates the creation of a multipurpose programmable switch capable of handling both forwarding and cybersecurity applications. (ii) High-speed network analysis; PDP's distinctive feature of line rate operation ensures efficient performance, even during DPI operations.

Detection techniques can vary from defining a simple threshold or a pre-installed matching list to the use of sophisticated ML algorithms. The following subsections present the review of different works based on each detection technique listed in Figure 2, whereas in Table 3 we present a summary, according to the proposed taxonomy, of the reviewed P4-based solutions that apply detection techniques.

1) THRESHOLD

This technique uses predefined values to trigger specific management actions and responses [35]. Typically, a threshold is associated with a numerical variable and indicates a limit value of that variable. It is often used in conjunction with other detection techniques, such as entropy and matching lists. This section will focus on those solutions that only use thresholds as mechanisms for the detection of potential attacks.

Paolucci et al. [37] used registers in P4 to store information about TCP SYN messages, the number of messages sent (or received), and source port numbers. A threshold is fixed to define a specific limit for the number of consecutive connection attempts, which, in excess, might be a symptom of port scanning.

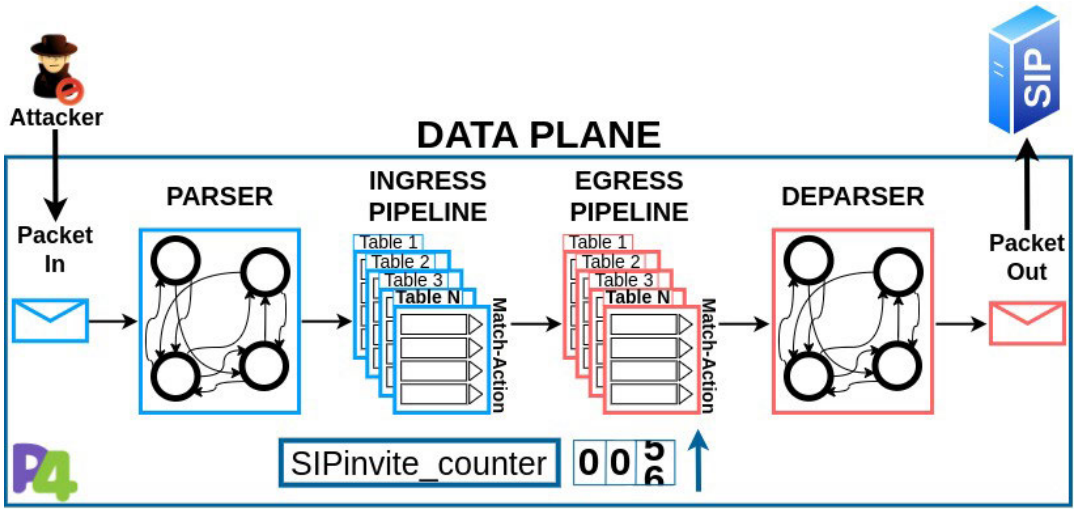

FIGURE 3. Solution based on thresholds and counters, applied to SIP invite messages within P4 switch (adapted from [36]).

Laraba et al. [38] used an Extended Finite State Machine (EFSM) abstraction implemented with P4, subject to the memory constraints of programmable forwarding devices. This EFSM requires capturing and storing specific information from headers of protocols such as TCP, UDP, and ICMP, among others. A set of detection modules is created using the mentioned EFSM abstraction. In this case, a decision module (based on a Petri Net model), varies its response with respect to a synchronized set of other detection modules. Each one of these modules is formed by a 7-tuple based on states, events, actions, initial states, variables, conditions, and transitions. Thus, the ability of P4 to perform DPI makes this analysis possible. Finally, based on the response of the decision module, the detection module uses status registers and counters to determine a set of alert and reaction levels, and subsequently warning of a possible attack attempt if a threshold based on the mentioned levels is reached.

Ding et al. [39] combined the Direct Bitmap [40] and Count-Min Sketch [41] registers into their proposal BACON (a probabilistic data structure implemented and used in P4), which allows estimating the number of distinct flows, thus reducing the memory usage of switches where it is deployed. When a flow is characterized in BACON, it shows the different source IPs attempting to connect to a particular destination IP within a given time frame. If a specific number of connection attempts associated to a defined threshold is exceeded, an alarm is triggered.

Febro et al. [36] investigated the detection of INVITE flooding attacks in the Session Initiation Protocol (SIP). The authors used DPI leveraging programmable switches. Using a counter-based approach, they basically count the number of times that a SIP flow is matched. Based on the assumption that a human can not initiate more than ten calls in one second, this approach relies on a controller that evaluates a threshold associated with the number of call attempts. In addition, the authors used P4 to process various protocol header fields, including those in Ethernet, IPv4, UDP, and SIP, to facilitate the analysis. The approach proposed by the authors uses a dedicated table called "`sipinvite_table`" which incorporates match conditions for ingress port and SIP INVITE identification and a counter integrated into the table. An overview of the counter attached to Match-action tables is illustrated in Figure 3. In this work, the authors provided insights into enhancing network security and protecting even application protocols against threats by employing DPI and counter-based thresholds implemented at the data plane.

2) ENTROPY

The concept of entropy, which comes from Shannon's Information Theory, has been used by some authors as a relevant feature to be assessed on network flows. Entropy can be used to measure the amount of disorder, randomness, or variability in data elements, making it a suitable feature for the analysis of values of different protocol headers, such as IP addresses, source and destination ports, or protocol identification. By measuring the entropy of protocol header values in packets belonging to a flow, it is possible to detect anomalies or patterns that allow distinguishing between legitimate and compromised traffic. Hence, an increment in the entropy measure might be an indication of a security problem [35], [44], [45].

Ding et al. [44] proposed to normalize the entropy value of destination IP addresses in packets extracted from captured network traffic. The authors also adapted the theoretical entropy calculation process to accommodate the fact that P4 does not perform floating point arithmetic operations. Notably, authors utilized entropy as a threshold, which can be adaptive in specific scenarios. Each entropy calculation is compared with an adaptive threshold for a time interval, thus enabling the effective detection of anomalous patterns.

Lapolli et al. [42] presented a real-time DDoS attack detection approach that focuses on calculating the Shannon entropy of source and destination IP addresses in order to identify anomalies. Built on this concept, Silveira Ilha et al. [43]

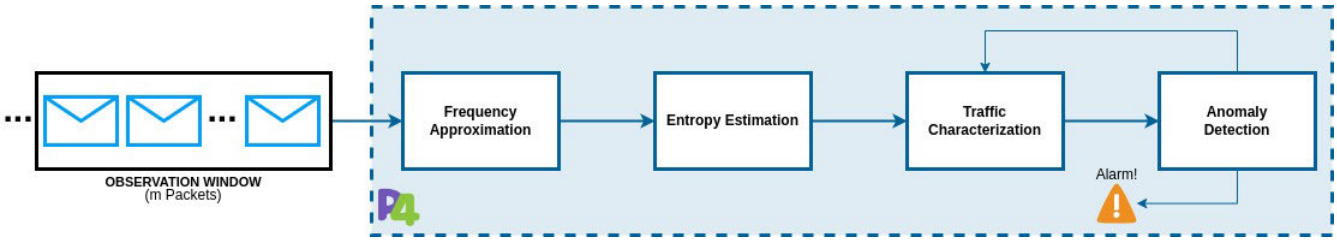

FIGURE 4. Solution based on entropy and Anti-DDoS detection mechanism (adapted from [42], [43]).

incorporated the entropy calculation technique into their proposal. Their approach incorporates, as well as [42], the calculation of entropies of IP addresses, an approximation to the measurement of IP packet frequencies, and an introduction of a smoothing coefficient to support the detection process, avoiding short-term fluctuation. Both approaches differ mainly in the deployment platform, as we will detail in Section IV-C.

Authors in [42] and [43] employed a fixed-sized observation window (OW) as the starting point for their respective detection techniques. Figure 4 displays a visual representation of the general operation steps used in these solutions. Within a single OW, a *frequency approximation* operation is utilized to count the occurrences of every distinct source and destination IP addresses in the traffic. This frequency approximation then serves as an input parameter for an *IP addresses Shannon entropy estimation* process. The mentioned entropy estimation builds a statistical model to represent benign network traffic conditions. Then, the statistical model is updated with the counts obtained within the OW as a mean to accommodate changes in traffic patterns over time. Later, the *anomaly detection* stage involves calculating the entropy of the IP addresses to compare it to the statistical model built as indicated above. Any significant deviations from the expected entropy values indicate unusual changes in traffic behavior. In response, the system generates an alarm or updates the statistical model accordingly. The approach employed in [42] and [43] showcases a framework as shown in Figure 4. This framework represents an interesting approach to the detection of DDoS attacks, produces valuable insights for network traffic analysis, and provides a basis for proactive identification and response to anomalous events.

3) MACHINE LEARNING (ML)

ML has emerged as a powerful tool for extracting insights and making predictions from large amounts of data. In recent years, the adoption of ML has been extended to the networking domain, ultimately within PDPs. As previously mentioned, P4 language enables the development of flexible and customizable packet processing, and it can even enable the direct implementation of custom algorithms and techniques (including ML) within packet forwarding devices in the network [46]. There are four main advantages that emerge from the integration of ML capabilities into PDPs. First, ML algorithms can analyze and learn from network data in real-time, thus enabling intelligent decision-making and adaptive behavior in packet processing. Second, the ability to process and extract relevant features directly at the data plane reduces reliance on centralized control and improves network efficiency. Third, ML algorithms implemented in PDP facilitate the detection of complex patterns, anomalies, and security threats, providing new and outstanding approaches to network security and performance improvement. Finally, the location of Programmable Forwarding Devices at certain vantage points might represent an ideal position for some use cases, such as the mitigation of attacks (which can be performed closer to the attack source). The integration of ML and PDP holds immense potential to transform network operations and enhance network management and performance. Indeed, some PDP researchers have been tackling this exploitable area in recent years, as is evident in the literature [47] and [48].

The integration of ML models into programmable switches is not a trivial task. The first thing to consider is the trade-off between the resource consumption associated with the ML model and the one used for the packet forwarding task. Despite the potential for functions such as in-band attack detection, it is important to preserve the line-rate performance [48]. Additionally, given the limitations of the P4 programming language, the incorporation of ML models might require certain abstractions in order to map the model artifacts to the switch pipeline structures (e.g. Match/Action tables). Moreover, despite the switch program itself might not change during the operation, it is possible that table structures do need to change as a mechanism of adaptation to the variations in traffic dynamics. Hence, this adaptation also needs to be supported. In our taxonomy, we analyze works implementing ML-based classifiers published since 2019 which are either totally or partially implemented in the data plane. Table 2 presents a summary of the reviewed works.

Macías et al. [49] proposed ORACLE, a collaborative solution integrating data and control planes to detect DDoS attacks. The architecture of ORACLE efficiently detects attacks with fine-grained inspection leveraging the PDP for feature extraction. The authors addressed controller overloading and data plane feature extraction challenges. Arithmetic conversions resolved operational limitations in P4, particularly the square roots and division operations. Flow information and statistics are stored using registers and hash

functions in data structures. Flow registers are created by hashing 5-tuple inputs, distinguishing bidirectional (BD) and unidirectional (UD) flows. Figure 5 depicts the workflow of ORACLE. ORACLE implements with P4 the extraction of flow features, which are not feasible to be extracted with a traditional control plane approach using OpenFlow. Also, ORACLE uses registers and hash functions to index flow data in a *data structure*, feeding a *statistics collector* component that manages packet features. The 5-tuple inputs are stored in registers and then processed by hash functions to discern BD/UD flows and features. The features extracted by the *pipeline handler* include flow duration (BD), flow inter-arrival time standard deviation (BD), backward packet payload sizes (UD), and average packet payload size per flow (BD). This flow recognition functionality implemented with P4 facilitates mathematical aggregation in the PDP, thus allowing the extraction of additional features: total packet count, payload size, size square (backward), and normal/square inter-arrival times. An *information transmitter* forwards organized features to the control plane where a process of classification is performed with a trained *ML model*.

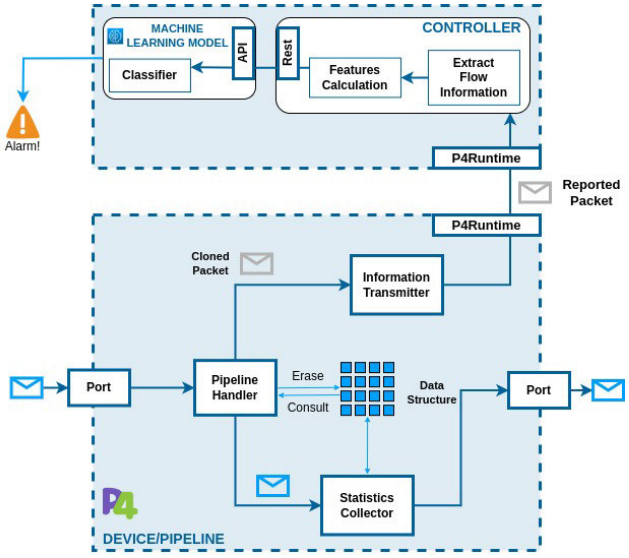

FIGURE 5. ORACLE workflow (adapted from [49]).

Barradas et al. [50] introduced FlowLens, an approach performing real-time classification of malicious packets using a collaboration between data and control planes. The data plane collects compact features and sends them to the control plane which classifies using ML techniques. The authors employed well-known ML algorithms such as XGBoost (Gradient Boosting), Multinomial Naïve-Bayes, and Random Forest (RF) for the classification task. The authors consider the frequency distribution intervals of the packet flows, and from these intervals, the most relevant features are extracted for subsequent classification. Notably, the authors proposed a methodology that manages to reduce and optimize the use of memory and processing time in forwarding devices. The design and implementation of this ML-based solution aims for accurate and resource-efficient detection of malicious network traffic, thereby contributing to enhance the deployment of effective network security measures.

Musumeci et al. [51] proposed an innovative solution that leverages ML techniques for the direct detection of DDoS attacks (specifically TCP SYN flood) within the data plane. Their approach aimed at minimizing data forwarding latency while mitigating the transmission of malicious packets toward a detection module. To reach the solution, the authors employed a feature extractor module and a binary classifier based on ML algorithms applied during Observation Windows (OW) for practical analysis. The extracted features include average packet length, SYN flags, and ratios (the percentage of packets containing particular values of given header values out of the total packets observed during the OW), such as TCP ratio, UDP ratio, and TCP-UDP ratio. The ML algorithms utilized included RF, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM).

Paolucci et al. [52] explored the potential of the P4 programming language for implementing an Artificial Neural Network (NN) directly within a programmable switch. Authors demonstrated that protocol header fields or metadata could serve as practical input features to NNs. In this proposal, the neurons were implemented as integer variables of 8 bits. The NN was trained using a subset of features extracted from the UNSW-NB15 [53], [54] data set, which can be easily extracted with primitives of the P4 language. The output of each neuron of the NN was computed as the weighted sum of all the feature inputs, followed by the application of a Rectified Linear Unit (ReLU) activation function. The weights of the NN layers were updated through interactions with the control plane, seeking for excitatory or inhibitory stimuli. This work demonstrates the potential of P4 to enable the deployment of NNs within programmable switches, paving the way for intelligent and adaptive packet processing in network environments.

Lee and Singh [55] proposed embedding an RF algorithm into a programmable switch for efficient in-network data analysis and real-time attack detection in SDN/PDP environments. This approach utilizes Match/Action tables to process traffic features in stateless and stateful ways. Authors harnessed the P4 language to extract these traffic features right at the data plane. The RF model was trained using a subset of the top 12 features available in the UNSW-NB15 dataset [53], [54], considering memory and capacity constraints common in programmable forwarding devices. The hyperparameters of the RF classifier were determined through offline training and testing, considering the trade-off between possible overfitting and memory utilization. The implementation of the RF algorithm within the switch involves embedding Decision Trees (DT) into Match/Action tables. Each decision tree level becomes represented by a table with keys and actions based on conditional evaluations. For instance, as shown in Algorithm 1, *node_id* indicates the

location of a node within the tree, *prevFeature* denotes the ID of a previous feature, and *isTrue* indicates whether the last condition has been satisfied. Actions can include parameters like the evaluation of, for instance, thresholds associated with packet counts. The Match/Action tables implement the verification of decision tree features using an action called *CheckFeature* or triggering an action called *SetClass* located at the final node. The ultimate classification is the result obtained from a majority vote among decision trees used in the implementation of the RF algorithm.

Algorithm 1 Example of a P4 Match/Action Table Defined for the *n-th* Decision Tree Level [55]

```
Table level_n {
    key = {
        meta.node_id: exact;
        meta.node_prevFeature: exact;
        meta.node_isTrue: exact;
    }
    actions = {
        NoAction;
        CheckFeature;
        SetClass;
    }
    size = 1024;
}
```

TABLE 2. ML algorithms used in the P4 solutions described.

P4 Solution	Year	ML Algorithm						
		DT	RF	XGBoost	KNN	SVM	NN	Others
Lee et al. [55]	2020	●	●	○	○	○	○	○
Musumeci et al. [51]	2020	○	●	○	●	●	○	○
Macías et al. [49]	2021	○	●	○	●	○	○	○
Paolucci et al. [52]	2021	○	○	○	○	○	●	○
Zhang et al. [56]	2021	●	○	○	○	○	○	○
Barradas et al. [50]	2021	○	●	●	○	○	○	●
Zheng et al. [57]	2022	●	●	●	●	●	●	●
Hardegen et al. [58]	2022	○	●	○	○	○	○	○
Ganesan et al. [59]	2022	●	○	○	○	○	○	●
Heggi et al. [60]	2022	○	○	○	○	○	○	●
Zang et al. [61]	2022	○	●	●	○	○	○	●
Carvalho et al. [62]	2022	○	●	○	●	●	○	○
Coelho et al. [63]	2022	○	●	○	○	○	○	○
Roshani et al. [64]	2022	○	●	○	●	●	●	○
Gaikar et al. [65]	2023	●	○	○	○	○	○	○
Al Sadi et al. [66]	2023	○	○	○	○	○	●	○
Zang et al. [67]	2023	○	○	○	○	○	○	●
González et al. [68]	2023	○	●	○	○	○	○	●
Doriguzzi et al. [69]	2023	○	○	○	○	○	●	○
Wang et al. [70]	2023	○	○	○	○	○	●	○
Khedr et al. [71]	2023	●	●	○	●	●	○	●

4) MATCHING LIST

Several P4-based cybersecurity solutions rely on special lists that define specific treatments for incoming packets. In the P4 programming language context, special lists are implemented as a set of rules indicating matches. Utilizing matching techniques can be enhanced through the implementation of DPI strategies. This approach entails a thorough inspection of both header and payload contents for every packet processed by the programmable switch acting as a NIDS. The NIDS, equipped with the capability to identify predefined patterns, requires a defined decision based on inspection recognition. This decision is then implemented as the action of a Match-Action table defined in the P4 language. Hence, a matching list is a data structure that exploits DPI fundamentals and is used to compare incoming packet fields against a predefined set of values or patterns expressed as entries into the Match/Action table. This comparison is based on identifying the total or partial similarities of given fields within packet headers [35]. The primary objective of the matching process is to efficiently classify incoming packets and facilitate their processing through network functions [25].

Regularly, the implementation of matching lists with P4 uses two common approaches to handle incoming packets, *whitelists* and *blacklists*. A *whitelists* is a list of specific and authorized network parameters expressed as values for protocol headers, indicating the packet on which some predefined actions may be performed. On the other hand, a *blacklists* is a list of unauthorized network parameters, also expressed as values of protocol headers, indicating packets that might be associated with malicious activities [35].

Vörös and Kiss [72] introduced a pioneering security middleware deployed in the data plane, marking the first implementation of a P4-based Firewall. This work holds significant importance in network security, influencing subsequent research and practical implementations. The proposal integrates protocol filters, port filters, and flood attack detection, leveraging the ability of P4 to process custom packet headers. This aligns with second-generation Firewalls, extending to the fourth layer and utilizing stateful memory. The implementation focuses on the parsing of Ethernet headers, with emphasis on the *ethertype* field indicating encapsulated *protocols* like IPv4, IPv6, TCP, or UDP, which is used to guide the protocol parsing at the P4 program implementing the Firewall. The solution operates based on a *blacklist* approach, utilizing a Matching List implemented with stateful memory to check a ban list. Counters are also used to measure packet rates per IP (DoS attacks), unsuccessful connection attempts (Portscan or SYN floods), and transferred bytes. Match/Action tables are used to enforce rules based on exact matches on protocol numbers, IP, or MAC addresses and Longest Prefix Match for IP addresses. These matches are used to indicate either forwarding or dropping actions. The foundational insights of this work significantly influenced the evolution of P4-based Matching List solutions and Firewall designs, paving the way for advancements in network security.

Liu et al. [73] proposed JAQEN. This proposal aims at effectively detecting DDoS attacks by employing universal sketches [74]. The proposed detection methodology consists of two components, one in the data plane and the other in the control plane. Universal sketches represent a category of approximation algorithms designed to estimate several network statistics concurrently. These are traditionally addressed by specific algorithms such as high impact, entropy, and determination of distinct flows, which are applied in response

to commands received from a central controller or triggered by specific events. The control plane implements an API that allows the configuration of these sketches, applies decisions associated with the detection tasks, and analyzes various metrics derived from the data plane, such as heavy hitters, flow measures, and the mentioned entropy. An important advancement in this research involves the utilization of a network-wide resource manager. This manager strategically places detection modules across network devices, enhancing their efficiency and efficacy in identifying DDoS attacks.

Datta et al. [75] created P4GUARD, a software-based configurable Firewall that operates in the data plane of the network. The Firewall is implemented by utilizing a controller, which orchestrates the Firewall functionality leveraging P4 tables in the data plane devices. The control plane populates the tables within switches with new entries using information derived from an application running on it. Also, this Firewall provides statistics gathered at the data plane devices and transmitted to the controller. The Firewall operates based on policies defined within Match/Action tables that allow or block traffic based on features such as protocol numbers, IP addresses, port numbers, and switch IDs.

B. MITIGATION

At present, security solutions seek to implement mechanisms in order to defend computer networks against cyberthreats. These actions include strategies for monitoring, traffic analysis, detection, and mitigation, among others [94]. In that matter, emerging technologies offer functionalities to boost the network defense and strengthen the response against more sophisticated and complex attacks [2].

The concept of PDP has paved the way for the development of innovative cybersecurity strategies. By leveraging P4, the cybersecurity community has increasingly focused on developing security solutions, with a particular emphasis on the detection process. However, mitigation is also a paramount research domain that requires more attention.

Mitigation refers to the decision, action, or practice planned to reduce the level of risk associated with a threat scenario or vulnerability [95]. Mitigation strategies aim to develop robust systems to remediate the network after the detection of an attack. In this stage, the deployed defense systems evaluate the detection decision output and take the appropriate actions to alleviate the network and reestablish the correct network operation. In this paper, we classify the mitigation process according to the implemented actions leveraged by the P4 programming language. These actions are applied by the data plane devices when an attack is detected. Additionally, it is possible to simultaneously implement different mitigation actions according to security policies and system's defense specifications.

Furthermore, by leveraging the P4 capabilities and functionalities for packet customization, reconfiguration to define packet parsing, and the definition of specific packet formats, it is possible to implement sophisticated mitigation strategies. These strategies can be deployed entirely in the data plane without the controller intervention, but they could also establish collaborative actions among data plane devices.

In this survey, we present four categories to classify mitigation actions: dropping, filtering, redirection, and isolation. Table 4 presents an overview of the reviewed solutions in our proposed taxonomy, which are implemented with P4 and apply mitigation actions.

1) DROPPING

Dropping is a fast mitigation action that involves identifying malicious incoming traffic from a known attacker and subsequently discarding it. Dropping action can be implemented by blocking a network port or by using a *blacklist* that immediately drops packets associated with possible attacks. This action requires a high level of certainty in the detection process to avoid dropping legitimate traffic [96]. It is important to consider the impact of implementing dropping as a mitigation action by blocking a data plane device port. For instance, when blocking a compromised port, the dropping action might discard both malicious and benign traffic. Despite the possible impact against benign traffic, this is the most prominent mitigation action implemented in cybersecurity solutions leveraging the P4 language.

Simsek et al. [82] proposed DroPPPP, a modular defense system. When an attack is detected, an attack flag is set, indicating the beginning of the mitigation process. If the flag is active and the difference between timestamps of packets coming from a possibly malicious source is less than a predefined time window threshold, these packets are set to be dropped by the switch. When the attack finishes, the flag is reset, and the switch stores the timestamp of the last occurrence of an attack. To implement dropping, DroPPPP employs a Two-Rate-Three-Color marker. Packets marked *green* are forwarded normally, packets marked *yellow* are forwarded normally but a digest message is sent to the controller, and packets marked *red* are dropped. Likewise, Shen et al. [86] propose dropping as a mitigating action against TCP SYN and UDP flooding attacks. For TCP SYN flood attacks, the defense mechanism generates a *blacklist* after an attack is detected. On the other hand, for UDP attacks, authors propose a Two-Rate-Three-Color marker process to classify traffic. Packets marked *red* are dropped, while the ones marked *yellow* are sent to the controller for further inspection. Furthermore, in P4-NSAF [89], packets marked *red* are dropped when the count of these packets surpasses a rate threshold. Also, the defense system implements modules to countermeasure spoofing and flooding attacks against IPv6 networks.

Khooi et al. [83] proposed DIDA as a solution to mitigate DDoS attacks. Figure 6 presents an overview of the implemented detection and mitigation techniques in DIDA. For detection, the authors propose the use of a threshold associated with the number of requests and responses provided by an Intrusion Prevention System (IPS) network and a DNS resolver. In (a) and (b) steps, there is

TABLE 3. Overview of detection techniques leveraging P4.

P4 Solution	Year	Detection Techniques				Core Idea
		Threshold	Entropy	Machine Learning	Matching List	
Vörös et al. [72]	2016	●	○	○	●	Developing a P4 Firewall solutions that utilizes policies represented by rules in Match-action tables.
Datta et al. [75]	2018	●	○	○	●	Embedding a Firewall service in data plane. Controller establishes policies using P4 Match-action tables with calculated features and statistics.
Ndonda et al. [76]	2018	●	○	○	●	Parsing Modbus layer-7 protocol and create *whitelists* tables in a P4 switch.
Grigoryan et al. [77]	2018	●	○	○	●	Displaying the IP address of the attacker in custom fields once an attack is detected.
Cao et al. [78]	2018	○	○	○	●	Calculating hash collisions in an external server using switch information to identify new forwarding commands.
Paolucci et al. [37]	2019	●	○	○	○	Storing P4 registers information about SYN sessions, with a specified limit for consecutive connections.
Febro et al. [36]	2019	●	○	○	○	Implementing a counter with a fixed threshold of 10 SIP INVITE packets per second.
Lapolli et al. [42]	2019	●	●	○	○	Performing entropy calculations on source and destination IP addresses to identify anomalies.
Kuka et al. [79]	2019	●	○	○	●	Detecting DDoS attacks in an ISP backbone by imposing a traffic bps rate limit threshold on the top-N sources.
Narayanan et al. [80]	2019	○	○	○	●	Implementing anti-spoofing mechanisms using P4 with white and black Match-action tables across various headers.
Friday et al. [81]	2020	●	○	○	○	adjusting dynamically a threshold based on traffic distribution statistics obtained from Bloom Filters.
Simsek et al. [82]	2020	●	○	○	●	Detecting spoofing and volumetric DDoS attacks using P4 operations involving hash functions, timestamps, registers, tables, and trTCM.
Khooi et al. [83]	2020	●	○	○	●	Sharing counter info between border and access routers, synchronizing them at a specified time interval τ to populate multiple ACLs.
Gondaliya et al. [84]	2020	○	○	○	●	Assessing the feasibility of applying traditional anti-spoofing mechanisms using P4 Match-action tables.
Musumeci et al. [51]	2020	○	○	●	○	Obtaining features like packet length, TCP/UDP ratio, and SYN flags. RF, KNN, and SVM algorithms are utilized for analysis.
Lee et al. [55]	2020	○	○	●	○	Embedding a RF algorithm with pre-selected and suitable features from the UNSW-NB15 dataset, for PDP analysis.
Macías et al. [49]	2020	○	○	●	○	Collaborating data and control planes for DDoS attack detection, using PDP for feature extraction, and ML models for classification.
Zhang et al. [56]	2021	●	○	●	○	Using DT algorithm for predicting heavy flows in base of PDP feasible features extracted from flow management.
Carvalho et al. [62]	2021	○	●	●	○	Collecting flow statistics within an OW to determine the flow class using RF, KNN, and SVM models.
Barradas et al. [50]	2021	○	○	●	○	Optimizing efficiently memory and processing with flow frequency distribution intervals.
Paolucci et al. [52]	2021	○	○	●	○	Implementing a neural network within a programmable switch for enhanced functionality.
Silveira Ilha et al. [43]	2021	●	●	○	○	Calculating IP address entropy and frequency, and a scaling coefficient to employ for further analysis.
Ding et al. [39]	2021	●	○	○	○	The system tracks source IPs connecting to a destination IP and alarms if the threshold is exceeded.
Liu et al. [73]	2021	●	●	○	●	Estimating attack volume using programmable switches and sort it for availability in the control plane.
González et al. [45]	2021	●	●	○	●	Implementing a dynamic pushback mechanism to calculate IP address entropy and compare it with a threshold.
Li et al. [85]	2021	●	○	○	●	Detecting procedures to the article is out-of-scope, it assumes a coarse-grained and traffic statistics procedures.
Shen et al. [86]	2021	●	○	○	●	Detecting based on counters for TCP and Two Rate Three Color Market (trTCM) for UDP.
Zheng et al. [57]	2022	○	○	●	○	Training ML models and mapping it to PDP, showcasing their feasibility, performance, and resource efficiency.
Hardegen et al. [58]	2022	●	○	●	○	Activating a specific RF classifier system, based on thresholds defined for links and flows.
Ganesan et al. [59]	2022	○	○	●	○	Performing a detection using the RF and logistic regression algorithms.
B. Coelho et al. [63]	2022	○	○	●	○	Mapping RF algorithm into Match-action tables to detect DDoS attacks using features extracted from IP, TCP, and UDP headers.
SR. Heggi et al. [60]	2022	○	○	●	○	Comparing the performance of LSTM, Naïve Bayes, ANN, and LSTM+NB models using feasible features extracted by P4.
M. Zang et al. [61]	2022	○	○	●	○	Processing features in-band and send them in UDP packets to a controller for classification using ML models.
R. Roshani et al. [64]	2022	○	○	●	○	Detecting volumetric DDoS attacks through collaborative detection between the data plane and ML classifiers with diverse ML models.
Laraba et al. [38]	2022	●	○	○	○	EFSM responses may trigger adjustments to targeted counters or registers, aiming to detect potential ongoing attacks.
Ding et al. [44]	2022	●	●	○	○	The theoretical entropy calculation process is adapted, normalized, and adjusted to accommodate the limitations of P4.
Febro et al. [87]	2022	●	○	○	●	Developing a DPI to verify exceeds thought counters.
Zhang et al. [88]	2022	●	○	○	●	NetHCF is a collaborative system utilizing Heavy Hitters and an IP2HC table in the data plane, to detect spoofed IP packets.
Li et al. [89]	2022	●	○	○	●	Detecting ICMPv6-based attacks by utilizing Match-action tables and Count-Min Sketch with an IPv6 frequency threshold.
Oh et al. [90]	2023	●	○	○	○	From the BACON sketch, it generates a top-k list of IP sources transmitting packets towards a potential victim.
Gaikar et al. [65]	2023	○	○	●	○	Applying a DT classification algorithm to detect malware attacks in IoT networks by encoding DT rules as P4 Match-action tables.
Smyth et al. [91]	2023	○	○	○	●	Using P4 mechanisms to detect topology poisoning attacks through DPI of Ethernet, IP, ARP, and LLDP headers.
Al Sadi et al. [66]	2023	○	○	●	○	Detecting DDoS attacks with Convolutional Neural Networks (CNN) through collaboration between control and data planes.
Zang et al. [67]	2023	○	○	●	○	Using Federated Learning with tree-based models on IoT edge, based on local models and parameters the system designs a global model.
Clemens et al. [92]	2023	●	○	○	○	Combing HYPERLOGLOG and COUNTMIN sketches to compare number of source IPs and packet counts to destination with specific thresholds.
González et al. [68]	2023	●	●	●	○	Detecting suspicious traffic with an entropy threshold and sending the value to the control plane for classification with ML techniques.
Doriguzzi et al. [69]	2023	○	○	●	○	Leveraging the PDP to extract features for detecting DDoS attacks and sending reports to the control plane for traffic classification using NNs.
Khedr et al. [71]	2023	○	○	●	●	Extracting IoT features from data plane with 9 state tables and send them to 6 ML classifiers. Interconnected controllers ensure consistent and real-time detection
Wang et al. [70]	2023	●	●	●	○	Adapting an entropy threshold based on network state per switch to detect DDoS flooding attacks, data plane reports feed a NN to detect non-DDoS attacks.
Jain et al. [93]	2023	●	○	○	●	Using a threshold based on the capacity of a table to detect flow table overflow attacks. If there are no matches, a flow request is sent to the controller.

a normal exchange of DNS requests. For instance, when compromised servers (c) send traffic that exceeds a rate threshold, the system detects the anomaly and indicates the possible occurrence of a DDoS attack. After the attack is detected by the switch closest to the victim, the system engages in the mitigation process by using the malicious packet for recirculation. The attack packet is customized and used to create a control tag to encode the IP address of the known attacker in the source IP field. This packet is sent (d) toward the switches at the edge of the network to block packets coming from the provided suspicious IP addresses. Hence, the dropping is carried out by reading a *blacklist* that matches the IP address or addresses of the attacker and the entry network port of the cyberattack. Moreover, DIDA also implements a *whitelist* to store information for known trusted devices.

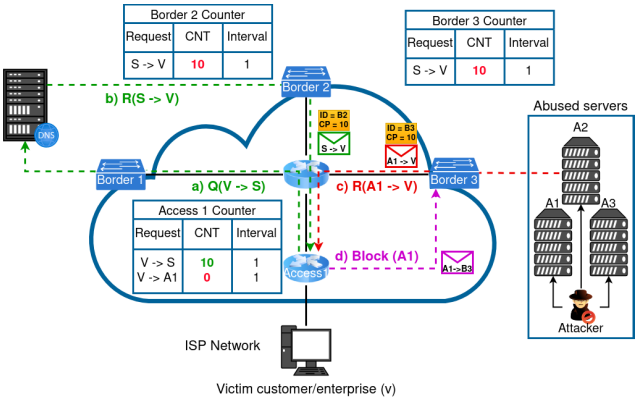

FIGURE 6. Detection and mitigation mechanisms implemented in DIDA (adapted from [83]).

In [43], Silveira Ilha et al. proposed EUCLID, a system that implements a defense-readiness state machine to define defense actions. The system sets an alarm flag and starts in the safe state where detection is active and mitigation is dormant. When an attack is detected, the system transitions to a defense-active state. In this state, EUCLID enforces packet dropping as a mitigation action. There is also a cooldown state where the dropping action is still enforced until a predefined number of time windows without attack occurs.

In Laraba et al. [38], authors proposed a two-module defense. The first module encompasses detection, while the second corresponds to a decision module. Authors implement mitigation actions inside a decision module that is designed as a Petri net. The defense system aims to mitigate multi-step and sophisticated attacks. A multi-step attack defines a sequence of steps with their corresponding sub-goals in order to disrupt the network. The mitigating action might be taken according to the sub-goals achieved by multi-step attacks. The Petri net is implemented using Match/Action tables in P4. Moreover, the Petri net defines the possible reactions and alerts, based on the patterns from the multi-step attack identified during the detection process. The system defines

different alert levels following the achieved sub-goals and applies dropping as mitigating action when all the sub-goals of the attack are achieved.

Kuka et al. [79] present an FPGA-accelerated system that provides a defense line against DDoS attacks by filtering traffic before it reaches the intended victims. The system defense is organized into modules. After malicious traffic is detected, this traffic is blocked using a module called *Blocking Filter*. This module defines hash tables to mark packets and collect statistics associated with offending IP addresses. After marking the suspicious packets, another module called *Check and Drop* discards attack traffic while also providing checks to detect packet malformation. The mitigation action filters traffic coming from the top-N sources by selectively blocking the IP addresses responsible for the attacks.

DIDA and DroPPPP introduce *blacklists* to indicate malicious traffic for further dropping. Moreover, in [73], [77], and [80], authors also apply the dropping action after parsing the incoming packets and checking them against a *blacklist*. Similarly, in [72] and [75], authors introduce Firewalls that apply dropping to traffic that matches ban lists. Usually, *blacklists* are constructed based on IP addresses. Nevertheless, by leveraging P4 capabilities, these *blacklists* can be generated based on different packet fields that match predefined security policies.

2) FILTERING

The filtering action is implemented similarly to dropping. For this action, filtering involves the selective discarding of malicious traffic to prevent disruptions in the network. Filtering might discard a portion of malicious packets following probabilistic methods or by statistical measurements. Moreover, filtering can be implemented as a mitigation approach while there is further analysis of the malicious traffic [45].

In BUNGEE [45], González et al. proposed an adaptive pushback mechanism to detect and mitigate DDoS attacks. Figure 7 presents the defense steps carried out by this solution implemented with P4. The pushback action allows the distribution of input information for detection techniques and mitigation actions progressively at the upstream data plane devices in the path from the attacker toward the intended victim. After attack detection (1), mitigation actions start to filter packets in order to restrict attack packets from reaching the intended victim (2). Then, through P4, the defense strategy consists of recirculating a cloned attack packet to the upstream devices. The packet is transformed into an alarm packet by adding a custom header that includes the IP address of the identified suspect. This alarm is used by the pushback (3,5) action in order to push the defense towards the edge of the network (4,6). BUNGEE implements filtering as a mitigation action in order to limit malicious traffic from reaching the network under attack. The system determines the suspect IP address and includes it in a suspect list. Packets that match the suspect list must be subjected to filtering. This

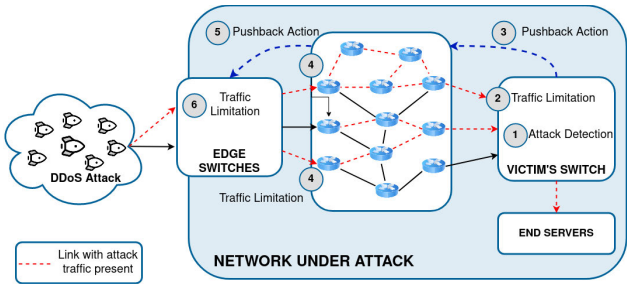

FIGURE 7. Overview of BUNGEE defense mechanism (adapted from [45]).

work adopts partial packet dropping, where there is a control ratio of suspect packets that will be allowed to follow their path toward the possible victim. The filtering mechanisms are applied at the edge of the network.

Hardegen et al. [58] proposed FLOWSEN, a security system to defend against heavy hitter attacks. These attacks aim to reveal private information by analyzing the frequency of interactions in computer networks. In FLOWSEN, authors incorporate dropping, filtering, and redirection as mitigation actions. The filtering strategy is applied to suspicious packets following a probabilistic method that leverages the P4 dropping primitive. The dropping probability depends on the amount of traffic rate exceeding a predefined threshold and the excess of a bit rate threshold. The higher the relative flow load, the higher the flow-specific drop ratio. Similarly, in [88], the NetHCF defense system deploys a filtering state to drop packets belonging to spoofing attacks that surpass a packet rate threshold.

In [36], upon the detection of a DoS attack against the SIP protocol, the defense system initiates the mitigation stage. In this stage, the controller installs rules in P4-programmable switches to filter traffic, and only drops malicious SIP INVITE packets from a device connected to a specific switch port. The drop action only discards malicious SIP INVITE packets while allowing other types of packets to pass through the specified port in the switch.

In BACKWARD [90], authors maintain a filtering list including the top-k connections to the intended victim. When an attack is detected, incoming packets are compared with the aforementioned list. Packets included in the top-k source list are discarded. The filtering list is shared with upstream devices by cloning a packet used to send a notification to upstream switches in order to pushback the defense and limit the attack near the edge of the network.

In JAQEN [73], Liu et al. proposed dropping, filtering, and redirection as mitigation actions. The filtering action is based on the implementation of *blacklists* and *whitelists* inside programmable devices. This action marks traffic aiming at allowing or discarding these packets from reaching the intended victim. The filtering action is implemented based on exact or approximated lists that contribute to stopping suspicious traffic from passing through the switch.

3) REDIRECTION

The redirection action consists of forwarding the malicious traffic to a specific device inside the network [96]. This action aims to further analyze the malicious traffic or to execute more sophisticated mitigating actions. Through this strategy, suspicious traffic can be forwarded to an IDS, a specific-purpose server (e.g. a honeypot), or an SDN controller, among other security-hardening devices, in order to further study the behavior of possibly malicious traffic. Furthermore, it might be possible to redirect or divert legitimate network traffic around sections that have been impacted by an attack. For instance, in the event of a link flooding attack affecting a network link, benign traffic could be redirected in order to circumvent the affected links until they are fully operational.

Ndonda et al. [76] designed a two-level defense system. The first level is implemented using BMv2 switches. The strategy leverages P4 Match/Action tables in order to construct a *whitelist*. According to this approach, in the first level, it is possible to define security policies and mitigation actions expressed through a *whitelist*. In case there is no match with the *whitelist*, the system redirects the traffic to an IDS in order to apply further analysis actions. The second defense level provides an IDS that uses a security engine to discard malicious packets while including their relevant information into a *blacklist* for further filtering of possible future threats.

Similarly, POSEIDON [85] implements redirection as one of the available mitigation actions, along with filtering and dropping. POSEIDON defines different primitive actions according to statistical results from the analysis of network traffic. Figure 8 presents a general overview of the implementation of POSEIDON. Actions can be executed completely on the switch, assisted by external general-purpose servers, or completely executed on specialized servers (e.g. honeypots and IDSs). "Switch-assisted" and "Server only" actions can implement redirection as mitigating actions. For switch-assisted actions, the mitigation is separated into actions executed in the switch while other actions are offloaded to servers. For instance, in the case of a SYN flood attack occurrence, it is possible to apply dropping and redirection as mitigating actions. If the traffic is benign, it is forwarded normally to the next hop, and the traffic might be dropped for suspicious packets. Otherwise, it is possible to redirect these packets to dedicated servers in order to further analyze them and take the required measures.

In [73], authors presented JAQEN, which aims to build a robust defense against volumetric DDoS attacks. JAQEN provides a P4 API to enforce switch-native mitigation actions such as redirection and filtering. The authors propose mitigation based on three steps: Filtering, analysis, and policy update. The filtering is based on lists that allow or block packets from reaching their destination. In the analysis step, packets are marked as either benign or as part of a DDoS attack. Finally, in the update process, the redirection action recirculates suspicious packets in order to further analyze them and update the filtering lists. The redirection action

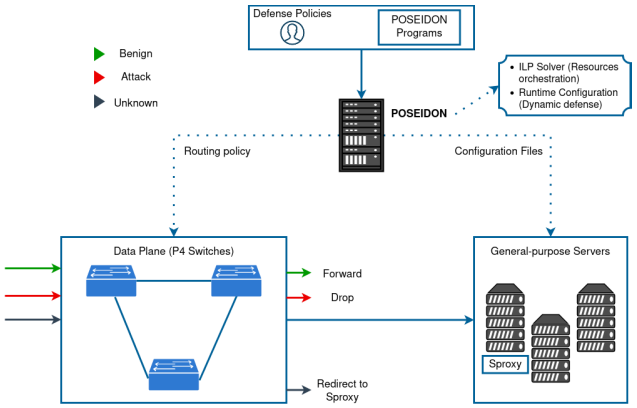

FIGURE 8. POSEIDON high-level architecture implementation including a defense instance (adapted from [85]).

clones packets and sends them to the control plane or tries to match them against a filtering list stored in the ingress pipeline of the switch. In that way, suspicious packets can be forwarded according to the decision represented with a *blacklist* or a *whitelist*.

In [37], Paolucci et al. proposed an optical offloading strategy to redirect traffic. The defense system leverages P4-based switches to divert traffic in order to decrease the congestion of the network in two scenarios. The first scenario consists of an edge switch that redirects malicious traffic to an external device for further analysis. The second scenario offloads traffic through an optical bypass when the packet rate limit surpasses a predefined threshold to avoid congestion attacks. This action permits benign traffic circulation through network channels with more available resources. On the other hand, Hardegen et al. [58] also proposed a redirection action in order to detour suspicious packets through non-optimal paths with the increased hop count in order to block them from reaching their destination.

4) ISOLATION

Traffic isolation is a mitigation strategy that confines malicious traffic to prevent attackers from exploiting network resources [96]. However, the implementation of isolation actions might be impractical due to the required amount of resources to store malicious traffic temporarily. This storing is required so that malicious packets can be further either analyzed or discarded. Moreover, the deployment of isolation strategies leveraging P4 is scarce, and there are just a few defense measures that incorporate isolation as a mitigation action. In fact, in our review of the state-of-the-art, we found only one work that proposed traffic isolation as part of its defense strategy.

In [37], authors proposed a defense strategy for detecting and mitigating DDoS attacks by implementing a NN. Figure 9 shows the pipeline of the proposed P4-programmable Neural Switch. Traffic ingress (1) into the switch where packets are parsed in order to extract useful information. The detection strategy involves traffic classification performed by the NN, where the inference output determines the

TABLE 4. P4 mitigation techniques overview.

P4 Solution	Year	Mitigation Actions				Core Idea
		Dropping	Filtering	Redirection	Isolation	
Vörös et al. [72]	2016	●	●	○	○	Security middleware that works as a software Firewall deploying *blacklist* and *whitelist* in a P4 switch. Packets that match with a *blacklist* are instantly dropped.
Datta et al. [75]	2018	●	○	○	○	P4Guard defines Firewall rules in the controller. When an attack is detected, these rules indicate to the P4-programmable switch to drop malicious packets.
Ndonda et al. [76]	2018	●	○	●	○	Two level defense, the first level drops malicious packets and redirects unknown packets to a second level functioning as an IDS that further analyses traffic.
Grigoryan et al. [77]	2018	●	○	○	○	LAMP defines a collaborative DP defense through a custom header embedding an alarm to drop malicious packets at the edge of the network.
Paolucci et al. [37]	2019	●	○	●	○	P4-programmable switch at the edge that applies dropping to malicious packets while implementing traffic offloading to dynamically reroute traffic when a threshold rate is surpassed.
Febro et al. [36]	2019	●	●	○	○	When a DoS attack is detected, the controller installs rules in the P4-programmable switch to filter packets and only drop malicious SIP invite packets coming from a specific port.
Kuka et al. [79]	2019	●	○	○	○	Implements a blocking filter to mark packets that surpasses a rate limit threshold, then a packet drop module discards malicious traffic.
Narayanan et al. [80]	2019	●	○	○	○	Defines different *whitelists* and *blacklists* where packets not matching with the *whitelists* are dropped by the P4 switch.
Friday et al. [81]	2020	●	○	○	○	Unified in-network detection and mitigation of DDoS attacks by defining custom headers to terminate the connection to block and drop malicious packets.
Simsek et al. [82]	2020	●	○	○	○	DroPPPP mitigates spoofing attacks by enforcing a trTCM to indicate whether a packet must be dropped, forwarded, or send to the controller for further analysis.
Khooi et al. [83]	2020	●	○	○	○	DIDA blocks unsolicited DNS responses by defining a custom packet with an alarm that it is sent to the edge of the network seeking to drop malicious traffic.
Gondaliya et al. [84]	2020	●	○	○	○	Defines an anti-spoofing mechanism drops packets that do not match with matches available in an access list.
Paolucci et al. [52]	2021	○	○	○	●	P4-programmable switch implementing an in-network Neural Network for traffic classification. After DDoS detection, the switch forwards these packets to an isolated network.
Silveira Ilha et al. [43]	2021	●	○	○	○	Euclid defines a defense-readiness state machine. In the defense-active state, the system activates mitigation actions by dropping DDoS malicious traffic.
Ding et al. [39]	2021	●	○	○	○	When a DDoS attack is detected, the controller of INDDoS instructs the P4-programmable switches to install rules that drop the packets belonging to possible malicious flows.
Liu et al. [73]	2021	●	●	●	○	JAQEN runs entirely in the DP. It deploys a mitigation API to enforce different mitigation actions such as dropping, filtering, redirection, or a combination.
González et al. [45]	2021	○	●	○	○	BUNGEE implements a pushback mechanism to move the defense to the edge of the network by creating a custom header specifying attack packets must be filtered.
Li et al. [85]	2021	●	●	●	○	POSEIDON is a responsive defense system where mitigation actions can be applied entirely in the DP, through general-purpose servers, or a combination.
Shen et al. [86]	2021	●	○	○	○	Mitigation of SYN and UDP flooding attacks by dropping the malicious packets entirely in the DP through a trTCM and *blacklists* for the respective attacks.
Hardegen et al. [58]	2022	●	●	●	○	FLOWSEN provides in-network classification leveraging ML techniques. It filters suspicious packets following a probabilistic method to discard them.
Ganesan et al. [59]	2022	●	○	○	○	Deploys ML techniques to classify traffic and detect attacks. The mitigation action involves the dropping of suspicious traffic.
Zang et al. [61]	2022	●	○	○	○	P4-based security solution that applies packet dropping in an IOT network in order to defend against DDoS attacks.
Roshani et al. [64]	2022	●	○	○	○	HybridDAD is a framework for the mitigation of DDoS attacks. After an attack is detected, the mitigation action drops packets according to Match-action tables.
Laraba et al. [38]	2022	●	●	○	○	Define a Petri net that determines the mitigation actions to be taken according to reached sub-goals in multi-step attacks. Defines alarms and dropping actions.
Febro et al. [87]	2022	●	○	○	○	Micro VNF is deployed in a P4-programmable network switch that defines an inspection and verification stage that drops malicious traffic that surpasses a threshold rate.
Zhang et al. [88]	2022	●	○	○	○	NetHCF is a line-rate in-network defense that drops packets belonging to IP spoofing attacks surpassing a threshold rate.
Li et al. [89]	2022	●	○	○	○	P4-NSAF deploys anti-spoofing and anti-flooding modules that drop malicious packets in IPv6 networks. Implements a trTCM, *blacklists*, and *whitelists*.
Oh et al. [90]	2023	●	●	○	○	BACKWARD implements a pushback action and filtering using a probabilistic method for the top-k source trying to connect to an identified victim.
Gaikar et al. [65]	2023	●	○	○	○	Implements ML techniques to detect malware attacks in IoT networks. For mitigation, the P4-programmable switch drops malicious packets.
Smyth et al. [91]	2023	●	○	○	○	SECAP implements source address verification and in-network anomaly detection. It applies dropping if packets fail the aforementioned checkpoints.
Al Sadi et al. [66]	2023	●	○	○	○	When the AI module detects malicious traffic, the control plane install a new P4 traffic dropping rule.
Zang et al. [67]	2023	●	○	○	○	It performs a simple dropping actions after the ML labels a packet as malicious.
González et al. [68]	2023	●	●	○	○	BUNGEE-ML initially filters suspect packets, then, after ML classification, the defense implements a pushback action to drop malicious packets.
Khedr et al. [71]	2023	●	○	●	○	Mitigation is distributed among controllers where the system drops all packets originating from the attack source. It also redirects benign traffic to avoid infected network nodes.
Wang et al. [70]	2023	●	○	○	○	A P4 switch discards attack packets independently. A safe back mechanism allows the controller to install rules to drop packets that were miss-classified as benign.
Jain et al. [93]	2023	●	○	○	○	The system is designed to mitigate flow table overflow attacks by blocking a port in the switch and dropping incoming packets. The port is added to a suspicious list.

packet forwarding rule. For benign traffic, packets are forwarded through a specific interface (2). If an attack is detected, the switch forwards the traffic through a different interface (3) that is connected to an isolated network. This network serves as a quarantine area for malicious traffic. Therefore, it is possible to avoid the disruption of the network while DDoS traffic is temporarily confined. Subsequently, according to the authors, this P4-based solution was the first approach to an Artificial Intelligence (AI)-driven switch that effectively applies in-network AI analysis by leveraging P4 programmability.

C. PLATFORM

The P4 programming language can be utilized across various platforms. Since P4 is nowadays the *de-facto* standard programming language for PDPs, according to our findings, the academic community and industry commonly opt for specific hardware and software-based platforms for the practical and experimental realization of their implementations. The following platforms emerge as the prevailing choices for facilitating the deployment of the contributions identified in the literature. Three of the most important platforms are BMv2 [7], which is software-based, and Intel Tofino [9] and NetFPGA [8], which are hardware-based. BMv2 can be cheaper than Tofino or NetFPGA because users just need an efficient computer to run the P4 programs. On the other hand, a Tofino switch or a NetFPGA device can provide more accurate experimental results, especially when distinguishing between emulated and real testbeds, as corroborated by existing literature.

1) BMV2

The Behavioral Model version 2 (BMv2) is an implementation in C++ of a software switch that can be programmed with P4. Its primary purpose is to emulate the entire packet processing behavior of the data plane using the P4 language. BMv2 operates by taking a JSON file as input, which is generated through the successful compilation of a P4 Program using the P4 reference compiler. It is important to emphasize that BMv2 is not intended for production use. Instead, it serves as a valuable tool for debugging, development, and research experimentation, particularly in the context of PDP environments [7].

It is worth noting that experiments with BMv2 are commonly integrated with Mininet [97] to create customized network topologies based on a PDP environment. Furthermore, BMv2 can be connected to a controller, thus implementing a complete realization of the SDN paradigm, considering the programmability of both the control and data planes [98].

Most of the reviewed works implement their P4-based security solutions using BMv2 switches. Authors can take advantage of software switches in order to expedite the deployment and testing of the proposed P4-based security solutions. For instance, in [72], authors proposed the first security middleware using P4 by leveraging a BMv2 switch. In [52], authors presented the demonstration of a P4 Neural Network switch deployed in a BMv2 switch. This work performs in-network traffic classification to detect and mitigate DDoS attacks. For works such as [49], [51], [55], and [63], authors used BMv2 switches to extract relevant traffic features in order to apply detection mechanisms through ML algorithms. Moreover, in P4-based security solutions such as [37], [38], [42], [45], [58], [77], [81], and [83], authors took advantage of BMv2 switches aiming to design a testing environment to evaluate the performance of the corresponding P4 detection and mitigation strategies.

2) BARETOOT-INTEL TOFINO

The Barefoot-Intel Tofino (BIT) programmable switches are specifically designed to implement PDPs through the use of the P4 programming language, enabling users to define and

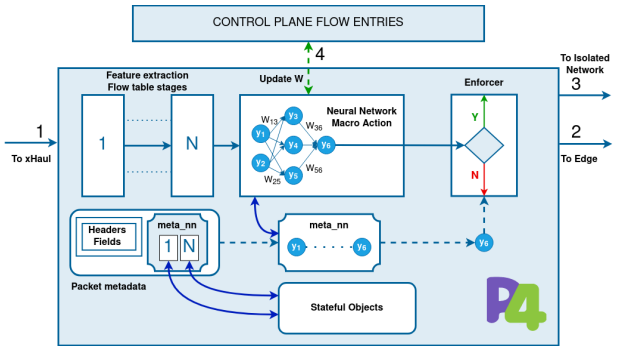

FIGURE 9. Pipeline of the proposed P4-programmable neural network switch (adapted from [52]).

customize packet processing behavior within the data plane. This programmability empowers the implementation of application-specific network functions, providing flexibility and adaptability to administrators. Notably, these switches exhibit notorious packet processing capabilities, operating at line-rate, with speeds of up to 24 terabits per second (Tbps), while maintaining custom programmed forwarding table capacities.

In the realm of research, BIT switches offer valuable network visibility and telemetry features, useful for the implementation of monitoring and analysis of network traffic patterns while preserving very high performance. Such capabilities hold significant promises for the research community, facilitating in-depth investigations and explorations within network-related domains, including cybersecurity [9].

TNA is the BIT solution based on PSA, both define a predetermined set of pipelines, each assigned to specific ports on the programmable switch. These pipelines include functions like parsing, deparsing, and control for both ingress and egress traffic [21]. Additionally, TNA offers conventional traffic management capabilities also found in PSA, such as mirroring sessions, packet replication using multicast, digests, resubmissions, and recirculation, among others. Notably, TNA incorporates dedicated packet generators at the end of each pipeline, capable of generating eight independent streams of packets, often referred to as *applications*. The triggering mechanisms for these applications are typically defined by on-time timers, periodic timers, port status changes, and packet recirculation events [22].

For instance, in [78], authors deployed CoFilter in a BIT switch and evaluated latency for traffic offloading by comparing their proposed solution against a software-based solution. Their findings indicated that traffic offloading latency using a BIT switch was 40 times lower than the latency in a software-based switch. In [50], authors deployed their P4-based solution in a BIT switch, aiming at exploring the capabilities of modern switches. In [56], authors implemented pHeavy in a BIT switch in order to demonstrate that the proposed P4-based security solutions can recollect and predict heavy flows at line-rate. Similarly, in [88], authors evaluated the performance of HetHCF when filtering spoofed IP attacks at line-rate. In P4-based security solutions such as POSEIDON [85] and JAQEN [73], authors assessed the implementation of detection and mitigation strategies to defend against DDoS attacks by comparing the performance of these solutions on simulated switches such as BMv2 and real switches based on BIT.

3) NETFPGA

The Portable Network Architecture (PNA) is equivalent to PSA, with the key distinction that it focuses on Network Interface Controllers (NIC) rather than switches. PNA is designed to manage data flows transiting between one or multiple interfaces at end host systems [20]. To the best of our knowledge, the most used PNA platform is NetFPGA SUME, which provides researchers with a practical tool for network experimentation, setting it apart from the software-focused approach of BMV2. This platform showcases the potential of integrating software and hardware capabilities in networking devices specifically designed for research and experimentation purposes. By utilizing Field-Programmable Gate Arrays (FPGAs), NetFPGA SUME offers a cost-effective solution for exploring and testing network designs. While its speed may be slightly lower compared to high-performance Intel-Tofino-based solutions, it still achieves data transmission rates of up to 100 Gbps. This combination of physicality and programmability makes NetFPGA SUME an appealing choice for researchers who want hands-on experience with network experimentation, even though it might not reach the same level of performance as BIT-based solutions [8].

Different authors deployed their defense solutions using an FPGA. In [37], [80], and [84], authors used NetFPGA for the deployment of their mitigation techniques, taking advantage of properties for accelerated packet processing and data filtering. Additionally, authors from [37] implemented their defense in NetFPGA and BMv2. They aimed to evaluate the proposed defense's impact over real programmable hardware devices. The P4-based defense solutions were validated in both platforms, and the authors concluded there was no significant performance degradation due to the NetFPGA parallel architecture. In [79], authors proposed an FPGA-accelerated device in order to mitigate attacks in an ISP backbone infrastructure before the attack reaches the intended victim.

V. ENHANCING CYBERSECURITY THROUGH P4-BASED SECURITY PROVISIONING AND ABSTRACTED FUNCTIONS

The P4 programming language has demonstrated its value in various networking domains, encompassing a wide range of solutions such as monitoring, traffic management, congestion control, routing and forwarding, 4G/5G cellular network (core networks), IoT, industrial networks, and time-sensitive networks [29]. In the context of cybersecurity, researchers have explored different P4-based approaches within two primary categories: **detection** and **mitigation**. These investigations include diverse platforms and have paved the way for the development of P4 solutions in their respective architectures,

Within the cybersecurity landscape, there is a trending category known as **Security Provisioning**, which focuses on proactive measures rather than reactive responses. This tendency involves the implementation of attack detection techniques and mitigation actions within a single system, leveraging P4 for security provisioning strategies. These strategies aim to mitigate potential vulnerabilities by implementing specific actions based on the taxonomy proposed in Figure 2. However, such conventional cybersecurity extra devices (firewalls, IDS, IPS, among others) often suffer from high costs, potential network performance degradation, and limited scalability in complex and critical systems [4], [99].

In this context, by harnessing the capabilities of PDP through P4 language, comparable cybersecurity functionalities provided by different middleboxes, such as Firewalls, IDSs, or IPSs, can be achieved through abstracted functions, while preserving the novel characteristics of the PDP architectures, such as line-rate efficiency. The following subsections illustrate the behavior related to the aforementioned middleboxes to determine which abstracted functions could be derived. Additionally, in this section, we present a summary of all the solutions addressed in this paper, along with the classification according to our proposed taxonomy (see Table 5).

A. FIREWALL

A Firewall acts as an enforceable security system that uses pre-defined rules to monitor and control network traffic, enabling the inspection of incoming and outgoing data. By establishing a virtual barrier between internal infrastructure and external networks, like the Internet, Firewalls play an important role in safeguarding network security. Two primary types of Firewalls exist: Host-based and Network-based. Host-based Firewalls operate on individual host computers and govern the flow of network traffic to and from these specific machines. On the other hand, Network-based Firewalls filter traffic among multiple networks and are typically implemented within network hardware [100]. In this paper, we focused on Network-based Firewalls.

Several P4-based cybersecurity proposals can be classified within this category, as their functionality relies on the implementation of Firewall operational principles and functionalities. For instance, Vörös et al. [72] devised the pioneering P4 Firewall featuring *whitelist* and *blacklist* capabilities for diverse Protocol Data Units (PDU). Similarly, González et al. [45] incorporated a suspects list, while Narayanan et al. [80] employed *whitelists* and *blacklists*, whereby incoming packets are evaluated against these lists. These *whitelist* and *blacklist* functionalities are governed by policy-based rules that dictate the appropriate actions to be enforced when a key is activated in a Match/Action table. Consequently, the handling of a flow relies on the utilization of table entries expressing network policies. These policies are subsequently enforced upon matches of packets against the rules defined within the Match/Action tables. The fine-grained processing capability of PDPs facilitates adaptable filtering of incoming traffic, enabling the management of flows in diverse manners.

B. INTRUSION DETECTION SYSTEM (IDS)

Hardware or software-based IDS can perform network monitoring in order to detect malicious activity or policy violations. If a violation occurs, the network administrator or a data collector is notified by alarms from different sources [100]. Considering this scenario, new approaches in PDP involving ML might be relevant since their classification result can be considered analogous to an alarm being triggered.

Several examples within the academic literature highlight the application of intrusion detection and classification techniques. For instance, Lapolli et al. [42] proposed an entropy-based anomaly detection approach, which triggers alarm notifications. In a similar way, Zhang et al. [56] introduced pHeavy, a binary classification framework trained on imbalanced data, enabling network operators to address the most significant and volumetric flows (or heavy flows) upon receiving alarms. Meanwhile, Zheng et al. [57] embedded ML algorithms into programmable switches to classify diverse traffic, and used the outcomes of this classification as alerts for a Security Operations Center (SOC), looking for appropriate response actions. Finally, Ding et al. [44] presented a DDoS detection strategy, which, by means of an analysis based on entropy variations obtained from normalized network traffic, generates alarms to the monitoring collector. These examples illustrate the PDP development and implementation tools for various intrusion detection mechanisms, using P4 language, showcasing the utilization of anomaly detection, classification, and entropy-based approaches for timely response to security threats detection in-band.

C. INTRUSION PREVENTION SYSTEM (IPS)

An IPS serves the purpose of not only monitoring network or system activities for potentially malicious behavior but also actively engaging in the identification, recording, reporting, and mitigation of such activities. Thus, IPSs are considered extensions of IDSs because they engage in the detection and mitigation of malicious activities [100].

Some PDP-based cybersecurity approaches can be placed in this category. Paolucci et al. [101] incorporate detection and mitigation in their solution. After detecting a pre-defined number of connection attempts in a port scan, the P4 switch discards the forthcoming attempts. Hence, it can be categorized as an IPS. Through its collaborative APIs, Liu et al. [73] proposed a P4-based solution that quickly adapts to cyber-attacks and their changes to detect and mitigate different threats and their alterations. Finally, microVNF [87] workflows treated packets with deep packet inspection, verification processes, and specific counters to determine whether a flow is malicious.

The following mechanisms are examples of collaboration between IPS and Firewall. González et al. [45] built a *blacklist* based on the entropy threshold. Then this *blacklist* feeds a Firewall rule. Once detection occurs, the system communicates with other P4-enabled switches to keep malicious traffic as close to the source as possible. When traffic becomes normal, the malicious source exits the *blacklists*. On the other hand, based on the constructed Match/Action tables, Narayanan et al. [80] created *whitelists* and *blacklists* that can develop rules and implement in the switch a behavior similar to a Firewall. Thus, the combination of different tables helps to harden the system against spoofing attacks.

The summary of the latter sections can be found in Table 5, which includes the use of our proposed taxonomy applied to various research papers, including categorization concerning detection techniques, mitigation actions, security provisioning, and platforms.

VI. RESEARCH CHALLENGES AND FUTURE TRENDS

This section presents a summary of trends and challenges identified in the field of cybersecurity solutions leveraging PDPs. It is important to note that this is an evolving area, and the specific trends and problems discussed might evolve with each new contribution.

A. DEVELOPMENT LIMITATIONS

The first challenge that PDP developers might face is transitioning from emulated environments to physical platforms. Emulated environments are designed and implemented with a complete alignment to the programming language specification, which might not be the case in actual chip implementations found in physical devices. Moreover, programming languages for PDPs (e.g. P4) lack different primitives and artifacts, such as floating point arithmetic and loop control structures, to guarantee line-rate performance and operation stability.

On the other hand, the use and management of a considerable amount of data and artificial intelligence algorithms can bring big advantages for network management [102], especially when it comes to cybersecurity. However, network devices, programmable switches included, face challenges due to their inherent memory limitations, which arise from restrictions in processing and storage capabilities; this poses a significant challenge in the implementation of attack detection approaches based on AI-techniques that are deployed in the data plane. Nevertheless, PDP researchers have made noteworthy progress in addressing these limitations by leveraging the potential of stateful elements within P4 and supporting elements within programmable architectures, such as meters, counters, and registers, among others. By employing efficient memory management techniques, including optimized data structures, hash operations, and memory allocation strategies, researchers have sought to maximize the utilization of available resources. However, aspects such as the effective processing of individual flows remain challenging.

It is crucial to note that cybersecurity solutions using PDP architecture might face development restrictions based on the system's design. However, earlier paragraphs presented instances where line-rate performance premise is mandatory and significant improvements in managing memory resources were achieved. This progress aims to make in-band cybersecurity solutions viable, including detection techniques and mitigation actions, all supported by elements (meters, registers, counters, among others) specified in the language [25] and the respective architecture [7], [20], [21], [22]. The above-mentioned fact requires continuous improvements and abstracted methods for their practical application, either just using the data plane [55], [61] or even applying a collaboration with the control plane [49], [52], [69]. The ongoing efforts to overcome memory constraints and processing delays are driven by the acknowledgment that effective cybersecurity measures depend on efficient resource utilization, intelligent data processing techniques, and robust memory management strategies. By further exploring and enhancing the capabilities of programmable switches, researchers aim to unlock the full potential of PDP in cybersecurity solutions, ensuring a harmonious balance between performance, resource constraints, and the evolving demands of network environments.

B. NEXT STEPS IN DATA PLANE BASED MACHINE LEARNING DEPLOYMENT

In recent years, there has been a significant surge in the popularity of AI solutions, particularly in the field of ML, driven by remarkable advancements in processors, GPUs, and computational capabilities. As a result, the landscape of AI-driven applications has witnessed substantial enhancements, introducing many practical achievements that were previously inconceivable. Even in the face of memory and processing limitations, these advancements have demonstrated that noteworthy results can still be achieved in the domains of networking and cybersecurity. In the context of networking, researchers have made notable efforts to adapt several ML algorithms in order to be implemented with P4 leveraging different PDP architectures (e.g. PSA, TNA) for cybersecurity classification purposes. These adaptations have primarily involved collaborations between control and data planes [49], [52], [69] or an offline training followed by the deployment of an abstraction on the data plane [55], [61].

However, the utilization of in-network ML detection techniques mandates the exploration of various target architectures, which can exhibit divergences in their definitions and components, including metadata instances [7], [20], [21], [22]. Fortunately, a common thread among these techniques is the incorporation of Match/Action tables featuring activation keys. Within the context of these tables and keys, establishing abstracted entities for features, classes, and even clusters are viable, depending on the specific ML classifier employed. According to the nature of the ML model and the aforementioned tables and keys, the executed actions manifest through feasible architectural operations

like voting, computed vectors, probabilities, distances, and others [48]. Finally, the programmable switch undertakes the classification of a flow by leveraging mechanisms such as decoded terms, vote tallies, highest probabilities, or distances [47].

Table 2 shows that ML-based P4 cybersecurity related implementations are supervised algorithms. This bias towards supervised learning techniques is a direct consequence of P4's inherent limitations in processing and memory, more so with unsupervised learning and deep learning that often tries to manage a considerable amount of information. However, it is crucial to acknowledge the existing gap when it comes to incorporating unsupervised learning and deep learning algorithms in the data plane. These algorithms hold great potential for effectively analyzing and processing heterogeneous data present in network traffic. On the one hand, unsupervised algorithms can leverage their ability to identify patterns and anomalies without labeled training data. Unsupervised algorithms are proficient at detecting patterns and anomalies in data even when no labeled training data is available. On the other hand, deep learning algorithms process data through multiple layers of neural networks to understand complex patterns. Both approaches offer valuable insights and can enhance cybersecurity detection in complex network settings, though they do require substantial amounts of data. Bridging the last mentioned gaps and exploring the utilization of algorithms, which require a large amount of data and memory (e.g. deep learning and unsupervised learning), in the data plane is an important avenue for future research and holds the promise of unlocking new capabilities and advancements in network security. Considering what was explained in the previous paragraph, one way to obtain satisfactory results is through collaboration between the control and data planes, seeking a complete process involving data collection, data processing, model validation, model deployment, and analysis of the results [103]. Thus, harmonious collaboration can avoid control plane exhaustion by delegating some tasks to the data plane, such as collecting better features based on the premise of DPI and PDPs.

C. TOWARDS IN-NETWORK COMPUTING AND SELF-DRIVING NETWORKS

The thriving of computer networks and the associated devices used by individuals and companies to carry out their daily activities makes managing networks more crucial than ever. There has been a continuous increase in the deployment of network hardware and software tools. Hence, with the expansion of computer networks, their management has become so complex that it seems impossible to have an effective control [13]. Consequently, new approaches are required for network management to improve the operation of computer networks. It is paramount to develop new network technologies that ease the burden of network operations while at the same time, the network administration capabilities are enhanced [104]. In that matter, the self-driving network concept aims to deploy autonomous networks capable of predicting changes and adapting to the behavior of users and enterprises without the need for human intervention [105].

Subsequently, it is necessary to introduce technologies that change the network in order to make it easier to manage. SDN is a paradigm that enhances network management by including substantial programmability functions. Likewise, PDPs allow the deployment of network management tasks, making it feasible to perform self-driving tasks. This way, it is possible to improve the performance of different network activities. Furthermore, with the development of PDP devices such as switches (e.g. Barefoot Tofino [9]) SmartNICSs (e.g Netronome Agilio cards [6]), among others, it is possible to perform in-network functions leveraging ML and DL.

Through ML and DL, network functionalities might be dynamically adapted according to the dynamics of the network. Moreover, both can be used for optimization and decision-making activities [47]. There are successful cases of ML and DL deployment algorithms. For instance, in Iisy [48], authors present different ML classification models in order to classify traffic and evaluate the feasibility, performance, and resource consumption of ML techniques.

In that matter, it is important to highlight the close relationship between ML and networking. Even more, ML is a fundamental tool to achieve the notion of self-driving networks. Additionally, ML can greatly benefit from new paradigms such as PDP in order to offload network functions to the data plane [46]. Nevertheless, despite the advantages of PDP, the implementation of ML or DL capabilities might have some limitations in terms of computational resources, memory storage, and limited operations. Consequently, it is desirable to aim at light ML/DL models with minor consumption of computing and memory resources while using simple arithmetic or logical operations. However, lightweight ML models might not achieve satisfactory performance for different network activities, as in the case of traffic classification [106]. Hence, alternatives such as Binary Neural Networks (BNN) have emerged as an alternative aiming to bring intelligence to the data plane [107]. Although BNNs are a good strategy, deploying large models in PDPs with such limited resources is still a challenge. Therefore, the networking community is constantly developing alternatives to deploy ML and DL in such a way that the vision of in-network computing can be achieved. For instance, it is possible to establish communication between control and data planes in order to deploy trained ML models on the control plane by capturing relevant traffic characteristics from the data plane devices.

Furthermore, in-network computing can significantly benefit the cybersecurity area. SDN and PDP play a vital role in enhancing network security. Self-driving networks provide the required autonomy in order to implement detection and mitigation strategies that do not require human intervention and can be adapted to different security needs. Through the implementation of ML and DL techniques, it is possible to automate the detection process. Moreover, for mitigation

TABLE 5. P4 solutions summary.

P4 Solution	Year	Detection				Mitigation				Security Provisioning			Platform		
		Threshold	Entropy	Machine Learning	Matching List	Dropping	Filtering	Redirection	Isolation	Firewall	IDS	IPS	BMv2	Barefoot Tofino	NetFPGA
Vörös et al. [72]	2016	●	○	○	●	●	○	○	○	●	○	○	●	○	○
Datta et al. [75]	2018	●	○	○	●	●	○	○	○	●	○	○	●	○	○
Ndonda et al. [76]	2018	●	○	○	●	●	○	●	○	●	●	○	●	○	○
Grigoryan et al. [77]	2018	●	○	○	●	●	○	○	○	○	○	●	●	○	○
Cao et al. [78]	2018	○	○	○	●	○	●	○	○	●	○	○	○	●	○
Paolucci et al. [37]	2019	●	○	○	○	●	○	●	○	○	●	○	●	○	●
Febro et al. [36]	2019	●	○	○	○	●	●	○	○	●	●	○	●	○	○
Lapolli et al. [42]	2019	●	●	○	○	○	○	○	○	○	●	○	●	○	○
Kuka et al. [79]	2019	●	○	○	●	●	○	○	○	●	○	○	○	○	●
Narayanan et al. [80]	2019	○	○	○	●	●	○	○	○	●	○	●	●	○	●
Friday et al. [81]	2020	●	○	○	○	○	○	○	○	○	○	●	●	○	○
Simsek et al. [82]	2020	●	○	○	●	●	○	○	○	○	○	●	●	○	○
Khooi et al. [83]	2020	●	○	○	●	●	○	○	○	○	○	●	●	○	○
Gondaliya et al. [84]	2020	○	○	○	●	●	○	○	○	○	○	●	○	○	●
Musumeci et al. [51]	2020	○	○	●	○	○	○	○	○	○	●	○	●	○	○
Lee et al. [55]	2020	○	○	●	○	○	○	○	○	○	●	○	●	○	○
Macías et al. [49]	2021	○	○	●	○	○	○	○	○	○	●	○	●	○	○
Zhang et al. [56]	2021	●	○	●	○	○	○	○	○	○	●	○	●	●	○
Carvalho et al. [62]	2021	○	●	●	○	○	○	○	○	○	●	○	●	○	○
Barradas et al. [50]	2021	○	○	●	○	○	○	○	○	○	●	○	○	●	○
Paolucci et al. [52]	2021	○	○	●	○	○	○	○	●	○	●	○	●	○	○
Silveira Ilha et al. [43]	2021	●	●	○	○	●	○	○	○	○	○	●	●	○	○
Ding et al. [39]	2021	●	●	○	○	○	○	○	○	○	●	○	●	○	○
Liu et al. [73]	2021	●	●	○	●	●	●	●	○	○	○	●	○	●	○
González et al. [45]	2021	●	●	○	●	○	●	○	○	●	○	●	●	○	○
Li et al. [85]	2021	●	○	○	●	●	●	●	○	○	○	●	○	●	○
Shen et al. [86]	2021	●	○	○	●	●	○	○	○	●	○	●	●	○	○
Zheng et al. [57]	2022	○	○	●	○	○	○	○	○	○	●	○	○	●	○
Hardegen et al. [58]	2022	●	○	●	○	●	●	●	○	○	○	●	●	○	○
Ganesan et al. [59]	2022	○	○	●	○	●	○	○	○	○	●	○	●	○	○
Coelho et al. [63]	2022	○	○	●	○	○	○	○	○	○	●	○	●	○	○
Heggi et al. [60]	2022	○	○	●	○	○	○	○	○	○	●	○	●	○	○
Zang et al. [61]	2022	○	○	●	○	●	○	○	○	○	●	○	●	○	○
Roshani et al. [64]	2022	○	○	●	○	●	○	○	○	○	●	○	○	●	○
Laraba et al. [38]	2022	●	○	○	○	●	●	○	○	○	○	●	●	○	○
Ding et al. [44]	2022	●	○	○	○	○	○	○	○	○	●	○	○	●	○
Febro et al. [87]	2022	●	○	○	●	●	○	○	○	○	○	●	●	○	○
Zhang et al. [88]	2022	●	○	○	●	●	○	○	○	○	●	○	○	●	○
Li et al. [89]	2022	●	○	○	●	●	○	○	○	○	○	○	●	○	○
Oh et al. [90]	2023	●	○	○	○	○	●	○	○	○	●	○	●	○	○
Gaikar al. [65]	2023	○	○	●	○	●	○	○	○	○	●	○	●	○	○
Smyth et al. [91]	2023	○	○	○	●	●	○	○	○	○	○	●	●	○	○
Al Sadi et al. [66]	2023	○	○	●	○	●	○	○	○	○	●	○	●	○	○
Zang et al. [67]	2023	○	○	●	○	●	○	○	○	○	●	○	●	○	○
Clemens et al. [92]	2023	●	○	○	○	○	○	○	○	○	●	○	●	○	○
González et al. [68]	2023	○	○	●	○	●	●	○	○	○	○	●	●	○	○
Doriguzzi et al. [69]	2023	○	○	●	○	○	○	○	○	○	●	○	●	○	○
Khedr et al. [71]	2023	○	○	●	●	●	○	●	○	○	○	●	●	○	○
Wang et al. [70]	2023	●	●	●	○	●	○	○	○	○	●	○	●	○	○
Jain et al. [93]	2023	●	○	○	●	●	○	○	○	●	○	○	●	○	○

actions, autonomous networks will allow the fast deployment of countermeasures against cyberattacks targeting the network. Consequently, by implementing in-network operations, it will be possible to deploy self-driving secure networks. In this way, enhancing and deploying proactive actions in the monitoring, prevention, detection, and mitigation strategies will be possible. For instance, monitoring actions might deploy network functions and manage network resources according to the current state of the network. Prevention strategies might leverage the analysis of traffic patterns to enforce decisions in the presence of suspicious traffic. For detection strategies, it is possible to implement ML and DL strategies to detect cyberattack patterns. Moreover, mitigation strategies can leverage the detection output in order to define and deploy different mitigation actions.

For instance, authors from [50], [51], and [55] implement their proposed ML algorithms in the data plane. Therefore, their defense strategies avoid bottleneck communication with centralized entities such as the network controller. This approach deploys classification decisions as match-action rules in the P4 switch. Moreover, in [52], authors present the first NN implemented directly on a P4 switch. In contrast, in articles such as [49], [59], [63], and [68], authors propose ML solutions that involve the collaboration of control and data plane. The data plane collects relevant traffic data required by the ML algorithm in order to perform the

classification. The ML can be placed in the control plane as an API application that activates an alarm upon the presence of an attack. Finally, implementing cybersecurity P4 solutions with ML faces the challenges of optimizing communication time among control and data plane collaboration, deploying classification rules in the P4 switches, and hardware resources to implement ML algorithms directly in data plane switches.

D. IMPROVING MITIGATION ACTIONS

Both academia and industry have increasingly focused on the development of P4-based cybersecurity solutions. Through a literature review it is possible to notice that most works center around the implementation of detection strategies, leaving out mitigation in second place. Mitigation is a paramount topic that requires more attention in the cybersecurity research domain. In this area, the focus is to develop robust systems to remediate the network post-attack detection. The development of appropriate techniques might enable a better response in terms of effectiveness and time response in the presence of cyberattacks. Therefore, and as mentioned before, there are few mitigation strategies leveraging programmable data planes. In this paper we showcase a large area of opportunities for implementing diverse mitigation actions leveraging Programmable Data Planes.

In order to achieve effective and fast alleviation of the network, mitigation solutions can take advantage of P4 to deploy more robust systems. Solutions can apply different mitigation actions as presented in the proposed taxonomy in Figure 2. Furthermore, by leveraging P4 and custom headers, it is feasible to implement collaborative mitigation actions among data plane devices without the intervention of the control plane. This way, the working load of the controller can be reduced while the mitigation measures are distributed across the whole network. Likewise, through packet customization, mitigation measures might include relevant information in the headers in order to execute different security tasks.

Subsequently, P4-based mitigation solutions can take advantage of the P4 language properties to combine different approaches that will enhance the response of the network. By leveraging network programmability, it is feasible to replace fixed middlebox devices for P4 solutions that include a mitigation module that enforces actions according to the network state and the characteristics of the attack. For instance, after an attack is detected, it is possible to indicate different mitigation rules to data plane devices without the need for controller intervention, which could improve response time during the mitigation process. Furthermore, more sophisticated mitigation actions can be implemented, such as deploying honeypots or applying techniques designed to alleviate path congestion through traffic redirection. Finally, from Table 5, it is noticeable an emphasis on the development of P4-detection solutions. Therefore, the development of P4-mitigation solutions is a topic that requires greater focus in order to take advantage of the capabilities offered by the P4 programming language.

E. INTRINSIC PDP SECURITY CHALLENGES

Through Programmable Data Planes (PDP), there is potential for significant advancement within the domain of network security aimed at protecting network devices against the more sophisticated network attacks occurring every day. In that matter, considerable efforts undertaken by academia and industry have been dedicated to deploying programmable switches seeking to execute diverse security tasks in the data plane. Moreover, incorporating security mechanisms within PDPs is a fundamental component in discussing research challenges to develop robust infrastructures in the data plane to prevent potential threats. It is crucial to engage in discussions concerning the security challenges intrinsic to PDPs, seeking to establish more robust strategies for detecting and mitigating attacks.

For instance, malicious actors have the potential to exploit vulnerabilities inherent to the programmability of the data plane, aiming to manipulate the network traffic behavior, thereby introducing novel attack vectors. In that regard, in [108], The authors present the term ‘‘sensitive attacks’’ to describe a category of attacks that specifically exploit the programmable nature of switches, aiming to manipulate network behavior by adapting network traffic. Furthermore, there is a critical need for robust security verification mechanisms, primarily because network administrators determine the definition of forwarding behavior. The user-defined configuration of the data plane could inadvertently expose vulnerabilities or bugs in the programming used to configure the programmable switches. In the context of P4, an expected requirement involves the development of mechanisms that facilitate the analysis and validation of P4 programs to ensure their accurate and secure operation.

Consequently, the successful implementation of detection and mitigation strategies within Programmable Data Planes (PDPs) relies on the effectiveness of data plane technologies, providing the essential reliability and security mechanisms necessary for PDPs to enhance their security.

VII. CONCLUSION

This research conducted a comprehensive exploration of P4 cybersecurity solutions within the timeframe of 2016 to 2023. The primary objective of this study was to contribute to the field of cybersecurity in PDP, specifically by providing a refined categorization of detection techniques, mitigation actions, and deployment platforms by the research community.

This paper started by providing a concise overview of the key components of the P4 data plane language in order to prepare the reader for a deeper understanding of the subsequent discussions. A collaborative taxonomy was then introduced, aiming to establish a framework that facilitates the integration of detection, mitigation, and deployment platforms, all using inter and intra-categories.

This taxonomy was designed to foster synergistic approaches and encourage the inclusion of complementary elements rather than promoting exclusivity. We presented the importance of a proactive cybersecurity, arguing why adopting functions abstracted from classical security devices can improve the performance of PDP cybersecurity solutions through the P4 language and including a Security Provisioning element of the diverse solutions.

Finally, this work concludes with a reflection on the identified research challenges and describes topics for future exploration. The identification and analysis of these challenges are crucial to provide new research branches for the research community engaged in PDP cybersecurity.

In summary, this work contributes to evolving the field of P4 cybersecurity solutions by offering valuable insights represented with the novel taxonomy. Also, highlighting the benefits of Security Provisioning and discussing our perception respecting the key research challenges and future directions.

REFERENCES

[1] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[2] J. C. Correa Chica, J. C. Imbachi, and J. F. B. Vega, "Security in SDN: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 159, Jun. 2020, Art. no. 102595.

[3] I. A. Valdovinos, J. A. Pérez-Díaz, K.-K.-R. Choo, and J. F. Botero, "Emerging DDoS attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions," *J. Netw. Comput. Appl.*, vol. 187, Aug. 2021, Art. no. 103093.

[4] M. Cheminod, L. Durante, L. Seno, and A. Valenzano, "Performance evaluation and modeling of an industrial application-layer firewall," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2159–2170, May 2018.

[5] Intel. (2023). *Data Plane Development Kit (DPDK)*. [Online]. Available: https://www.dpdk.org/

[6] NETRONOME. (2023). *NETRONOME Agilio SmartNICs*. [Online]. Available: https://www.netronome.com/products/smartnic/overview/

[7] P4 Language Consortium. (2015). *Behavioral Model Version 2 (BMv2)*. [Online]. Available: https://github.com/p4lang/behavioral-model

[8] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "NetFPGA SUME: Toward 100 Gbps as research commodity," *IEEE Micro*, vol. 34, no. 5, pp. 32–41, Sep. 2014.

[9] Intel. (2023). *Barefoot Tofino 2*. [Online]. Available: https://www.barefootnetworks.com/products/brief-tofino-2/

[10] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[11] P4 Language Consortium. *P4 Open Source Programming Language*. Accessed: Mar. 8, 2023. [Online]. Available: https://p4.org/

[12] S. Kaur, K. Kumar, and N. Aggarwal, "A review on P4-programmable data planes: Architecture, research efforts, and future directions," *Comput. Commun.*, vol. 170, pp. 109–129, Mar. 2021.

[13] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014.

[14] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on P4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, vol. 9, pp. 87094–87155, 2021.

[15] O. Michel, R. Bifulco, G. Retvari, and S. Schmid, "The programmable data plane: Abstractions, architectures, algorithms, and applications," *ACM Comput. Surv. (CSUR)*, vol. 54, no. 4, pp. 1–36, 2021.

[16] M. J. S. Smith, *Application Specific Integrated Circuits*. Reading, MA, USA: Addison-Wesley, 1997, doi: 10.1109/ACCESS.2019.2907793.

[17] *Application-Specific Integrated Circuit (ASIC)*. Accessed: Mar. 10, 2023. [Online]. Available: https://technav.ieee.org/topic/application-specific-integrated-circuit-asic

[18] T. Høiland-Jørgensen, J. D. Brouer, D. Borkmann, J. Fastabend, T. Herbert, D. Ahern, and D. Miller, "The express data path: Fast programmable packet processing in the operating system kernel," in *Proc. 14th Int. Conf. Emerg. Netw. Experiments Technol.*, Dec. 2018, pp. 54–66.

[19] A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing SDN from OpenFlow to P4: A survey," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–37, Sep. 2023.

[20] P4 Language Consortium. *P4 Portable NIC Architecture (PNA)*. Accessed: Mar. 10, 2023. [Online]. Available: https://p4.org/p4-spec/docs/PNA.html

[21] P4 Language Consortium. *P416 Portable Switch Architecture (PSA)*. Accessed: Mar. 10, 2023. [Online]. Available: https://p4.org/p4-spec/docs/PSA-v1.2.html

[22] Barefoot Networks-Intel. *P416 Intel TofinoT Native Architecture—Public Version*. Accessed: Mar. 10, 2023. [Online]. Available: https://github.com/barefootnetworks/Open-Tofino/blob/master/PUBLIC_Tofino-Native-Arch.pdf

[23] Y. Gao and Z. Wang, "A review of P4 programmable data planes for network security," *Mobile Inf. Syst.*, vol. 2021, pp. 1–24, Nov. 2021.

[24] P4 Language Consortium. (2023). *P4 Language and Related Specifications*. [Online]. Available: https://p4.org/specs/

[25] P4 Language Consortium. (2023). *$P4_{16}$ Language Specification*. [Online]. Available: https://staging.p4.org/p4-spec/docs/P4-16-v1.2.4.html

[26] M. Budiu and C. Dodd, "The $P4_{16}$ programming language," *ACM SIGOPS Operating Syst. Rev.*, vol. 51, no. 1, pp. 5–14, Sep. 2017.

[27] A. AlSabeh, J. Khoury, E. Kfoury, J. Crichigno, and E. Bou-Harb, "A survey on security applications of P4 programmable switches and a STRIDE-based vulnerability assessment," *Comput. Netw.*, vol. 207, Apr. 2022, Art. no. 108800.

[28] S. Kianpisheh and T. Taleb, "A survey on in-network computing: Programmable data plane and technology specific applications," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 701–761, 1st Quart., 2023.

[29] F. Hauser, M. Häberle, D. Merling, S. Lindner, V. Gurevich, F. Zeiger, R. Frank, and M. Menth, "A survey on data plane programming with P4: Fundamentals, advances, and applied research," *J. Netw. Comput. Appl.*, vol. 212, Mar. 2023, Art. no. 103561.

[30] *National Institute of Standards and Technology*. (2023). *Confidentiality, Integrity, Availability*. [Online]. Available: https://csrc.nist.gov/glossary/term/confidentiality_integrity_availability

[31] Cybersecurity & Network Security Solutions & Services. (2023). *Fundamentals*. [Online]. Available: https://www.rapid7.com/fundamentals/

[32] J. P. Anderson, "Computer security threat monitoring and surveillance," James P. Anderson Company, USA, Tech. Rep., 1980.

[33] 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.

[34] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, Dec. 2019.

[35] National Institute of Standards and Technology. (2023). *Glossary*. [Online]. Available: https://csrc.nist.gov/glossary/

[36] A. Febro, H. Xiao, and J. Spring, "Distributed SIP DDoS defense with P4," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–8.

[37] F. Paolucci, F. Civerchia, A. Sgambelluri, A. Giorgetti, F. Cugini, and P. Castoldi, "P4 edge node enabling stateful traffic engineering and cyber security," *J. Opt. Commun. Netw.*, vol. 11, no. 1, pp. 84–95, Jan. 2019.

[38] A. Laraba, J. François, I. Chrisment, S. R. Chowdhury, and R. Boutaba, "Detecting multi-step attacks: A modular approach for programmable data plane," in *Proc. IEEE/IFIP Netw. Operations Manag. Symp.*, Apr. 2022, pp. 1–9.

[39] D. Ding, M. Savi, F. Pederzolli, M. Campanella, and D. Siracusa, "In-network volumetric DDoS victim identification using programmable commodity switches," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1191–1202, Jun. 2021.

[40] C. Estan, G. Varghese, and M. Fisk, "Bitmap algorithms for counting active flows on high speed links," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, 2003, pp. 153–166.

[41] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58–75, Apr. 2005.

[42] A. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Offloading real-time ddos attack detection to programmable data planes," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Apr. 2019, pp. 19–27.
[43] A. d. S. Ilha, A. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Euclid: A fully in-network, P4-based approach for real-time ddos attack detection and mitigation," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 3121–3139, Sep. 2021, doi: 10.1109/TNSM.2020.3048265.
[44] D. Ding, M. Savi, and D. Siracusa, "Tracking normalized network traffic entropy to detect DDoS attacks in P4," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 4019–4031, Nov. 2022.
[45] L. A. Q. González, L. Castanheira, J. A. Marques, A. Schaeffer-Filho, and L. P. Gaspary, "BUNGEE: An adaptive pushback mechanism for DDoS detection and mitigation in P4 data planes," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, May 2021, pp. 393–401.
[46] W.-x. Liu, C. Liang, Y. Cui, J. Cai, and J.-m. Luo, "Programmable data plane intelligence: Advances, opportunities, and challenges," *IEEE Netw.*, early access, pp. 1–8, Dec. 2022, doi: 10.1109/MNET.124.2200113.
[47] Z. Xiong and N. Zilberman, "Do switches dream of machine learning: Toward in-network classification," in *Proc. 18th ACM Workshop Hot Topics Netw.*, Nov. 2019, pp. 25–33.
[48] C. Zheng, Z. Xiong, T. T. Bui, S. Kaupmees, R. Bensoussane, A. Bernabeu, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, "Iisy: Practical in-network classification," 2022, *arXiv:2205.08243*.
[49] S. G. Macías, L. P. Gaspary, and J. F. Botero, "ORACLE: An architecture for collaboration of data and control planes to detect DDoS attacks," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, May 2021, pp. 962–967.
[50] D. Barradas, N. Santos, L. Rodrigues, S. Signorello, F. M. V. Ramos, and A. Madeira, "FlowLens: Enabling efficient flow classification for ML-based network security applications," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.
[51] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-assisted DDoS attack detection with P4 language," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.
[52] F. Paolucci, L. De Marinis, P. Castoldi, and F. Cugini, "Demonstration of P4 neural network switch," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, Jun. 2021, pp. 1–3.
[53] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[54] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J., A Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, Apr. 2016.
[55] J.-H. Lee and K. Singh, "SwitchTree: In-network computing and traffic analyses with random forests," *Neural Comput. Appl.*, vol. 32, pp. 1–12, Nov. 2020, doi: 10.1007/s00521-020-05440-2.
[56] X. Zhang, L. Cui, F. P. Tso, and W. Jia, "PHeavy: Predicting heavy flows in the programmable data plane," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4353–4364, Dec. 2021.
[57] C. Zheng, M. Zang, X. Hong, R. Bensoussane, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, "Automating in-network machine learning," 2022, *arXiv:2205.08824*.
[58] C. Hardegen, S. Rieger, and T. Geier, "Multi-step attack detection and mitigation enhancing in-network flow classification," in *Proc. 5th Int. Conf. Adv. Commun. Technol. Netw. (CommNet)*, Dec. 2022, pp. 1–10.
[59] A. Ganesan and K. Sarac, "Attack detection and mitigation using intelligent data planes in SDNs," in *Proc. IEEE Global Commun. Conf.*, Dec. 2022, pp. 1–6.
[60] S. R. Heggi, P. Sukarno, and S. A. Mugitama, "LSTM-NB: DoS attack detection on SDN with P4 programmable dataplane," in *Proc. Int. Conf. Adv. Creative Netw. Intell. Syst. (ICACNIS)*, Nov. 2022, pp. 1–6.
[61] M. Zang, E. O. Zaballa, and L. Dittmann, "SDN-based in-band DDoS detection using ensemble learning algorithm on IoT edge," in *Proc. 25th Conf. Innov. Clouds, Internet Netw. (ICIN)*, Mar. 2022, pp. 111–115.
[62] R. N. Carvalho, L. R. Costa, J. L. Bordim, and E. A. Alchieri, "Detecting DDoS attacks on SDN data plane with machine learning," in *Proc. 9th Int. Symp. Comput. Netw. Workshops (CANDARW)*, 2021, pp. 138–144.
[63] B. Coelho and A. Schaeffer-Filho, "BACKORDERS: Using random forests to detect DDoS attacks in programmable data planes," in *Proc. 5th Int. Workshop P4 Eur.*, Dec. 2022, pp. 1–7.
[64] M. Roshani and M. Nobakht, "HybridDAD: Detecting DDoS flooding attack using machine learning with programmable switches," in *Proc. 17th Int. Conf. Availability, Rel. Secur.*, Aug. 2022, pp. 1–11.
[65] M. H. Gaikar and K. Haribabu, "A data-plane approach for detecting malware in IoT networks," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2023, pp. 578–583.
[66] A. A. Sadi, M. Savi, D. Berardi, A. Melis, M. Prandini, and F. Callegati, "Real-time pipeline reconfiguration of P4 programmable switches to efficiently detect and mitigate DDoS attacks," in *Proc. 26th Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Mar. 2023, pp. 21–23.
[67] M. Zang, C. Zheng, T. Koziak, N. Zilberman, and L. Dittmann, "Federated learning-based in-network traffic analysis on IoT edge," in *Proc. IFIP Netw. Conf. (IFIP Networking)*, Jun. 2023, pp. 1–6.
[68] L. A. Quintero González, L. Castanheira, J. A. Marques, A. E. Schaeffer-Filho, and L. Paschoal Gaspary, "Bungee-ML: A cross-plane approach for a collaborative defense against DDoS attacks," *J. Netw. Syst. Manage.*, vol. 31, no. 4, p. 77, Oct. 2023.
[69] R. Doriguzzi-Corin, L. A. D. Knob, L. Mendozzi, D. Siracusa, and M. Savi, "Introducing packet-level analysis in programmable data planes to advance network intrusion detection," *Comput. Netw.*, vol. 239, Aug. 2023, Art. no. 110162.
[70] Y.-C. Wang and P.-Y. Su, "Collaborative defense against hybrid network attacks by SDN controllers and P4 switches," *IEEE Trans. Netw. Sci. Eng.*, early access, pp. 1–16, Oct. 2023, doi: 10.1109/TNSE.2023.3324329.
[71] W. I. Khedr, A. E. Gouda, and E. R. Mohamed, "P4-HLDMC: A novel framework for DDoS and ARP attack detection and mitigation in SD-IoT networks using machine learning, stateful P4, and distributed multi-controller architecture," *Mathematics*, vol. 11, no. 16, p. 3552, Aug. 2023.
[72] P. Voros and A. Kiss, "Security middleware programming using P4," in *Human Aspects of Information Security, Privacy, and Trust*. Toronto, ON, Canada: Springer, 2016, pp. 277–287.
[73] Z. Liu, H. Namkung, G. Nikolaidis, J. Lee, C. Kim, X. Jin, V. Braverman, M. Yu, and V. Sekar, "Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric DDoS attacks with programmable switches," in *Proc. USENIX Secur. Symp.*, 2021, pp. 3829–3846.
[74] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman, "One sketch to rule them all: Rethinking network flow monitoring with UnivMon," in *Proc. ACM SIGCOMM Conf.*, NY, NY, USA, Aug. 2016, pp. 101–114.
[75] R. Datta, S. Choi, A. Chowdhary, and Y. Park, "P4Guard: Designing P4 based firewall," in *Proc. MILCOM - IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2018, pp. 1–6.
[76] G. K. Ndonda and R. Sadre, "A two-level intrusion detection system for industrial control system networks using P4," in *Proc. Electron. Workshops Comput.*, 2018, pp. 31–40.
[77] G. Grigoryan and Y. Liu, "LAMP: Prompt layer 7 attack mitigation with programmable data planes," in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2018, pp. 1–4.
[78] J. Cao, J. Bi, Y. Zhou, and C. Zhang, "Cofilter: A high-performance switch-assisted stateful packet filter," in *Proc. ACM SIGCOMM 2018 Conf. Posters Demos*, 2018, pp. 9–11.
[79] M. Kuka, K. Vojanec, J. Kucera, and P. Benácek, "Accelerated DDoS attacks mitigation using programmable data plane," in *Proc. ACM/IEEE Symp. Architectures Netw. Commun. Syst. (ANCS)*, Sep. 2019, pp. 1–3.
[80] N. Narayanan, G. C. Sankaran, and K. M. Sivalingam, "Mitigation of security attacks in the SDN data plane using P4-enabled switches," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Dec. 2019, pp. 1–6.
[81] K. Friday, E. Kfoury, E. Bou-Harb, and J. Crichigno, "Towards a unified in-network DDoS detection and mitigation strategy," in *Proc. 6th IEEE Conf. Netw. Softwarization (NetSoft)*, Jun. 2020, pp. 218–226.
[82] G. Simsek, H. Bostan, A. K. Sarica, E. Sarikaya, A. Keles, P. Angin, H. Alemdar, and E. Onur, "DroPPPP: A P4 approach to mitigating DoS attacks in SDN," in *Information Security Applications*. Jeju Island, (South) Korea: Springer, 2020, pp. 55–66.
[83] X. Z. Khooi, L. Csikor, D. M. Divakaran, and M. S. Kang, "DIDA: Distributed in-network defense architecture against amplified reflection DDoS attacks," in *Proc. 6th IEEE Conf. Netw. Softwarization (NetSoft)*, Jun. 2020, pp. 277–281.
[84] H. Gondaliya, G. C. Sankaran, and K. M. Sivalingam, "Comparative evaluation of IP address anti-spoofing mechanisms using a P4/NetFPGA-based switch," in *Proc. 3rd P4 Workshop Eur.*, Dec. 2020, pp. 1–6.

[85] G. Li, M. Zhang, S. Wang, C. Liu, M. Xu, A. Chen, H. Hu, G. Gu, Q. Li, and J. Wu, "Enabling performant, flexible and cost-efficient DDoS defense with programmable switches," *IEEE/ACM Trans. Netw.*, vol. 29, no. 4, pp. 1509–1526, Aug. 2021.
[86] Z.-Y. Shen, M.-W. Su, Y.-Z. Cai, and M.-H. Tasi, "Mitigating SYN flooding and UDP flooding in P4-based SDN," in *Proc. 22nd Asia–Pacific Netw. Operations Manag. Symp. (APNOMS)*, Sep. 2021, pp. 374–377.
[87] A. Febro, H. Xiao, J. Spring, and B. Christianson, "Edge security for SIP-enabled IoT devices with P4," *Comput. Netw.*, vol. 203, Feb. 2022, Art. no. 108698.
[88] M. Zhang, G. Li, X. Kong, C. Liu, M. Xu, G. Gu, and J. Wu, "NetHCF: Filtering spoofed IP traffic with programmable switches," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1641–1655, Mar. 2023.
[89] Y. Li, W. Yang, Z. Zhou, Q. Liu, Z. Li, and S. Li, "P4-NSAF: Defending IPv6 networks against ICMPv6 DoS and DDoS attacks with P4," in *Proc. IEEE Int. Conf. Commun.*, May 2022, pp. 5005–5010.
[90] S. Oh, S. Han, H. Lee, and S. Pack, "BACKWARD: A victim-centric DDoS detection and mitigation scheme in programmable data plane," in *Proc. IEEE 20th Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2023, pp. 989–990.
[91] D. Smyth, S. Scott-Hayward, V. Cionca, S. McSweeney, and D. O'Shea, "SECAP switch—Defeating topology poisoning attacks using P4 data planes," *J. Netw. Syst. Manag.*, vol. 31, no. 1, p. 28, Jan. 2023.
[92] V. Clemens, L.-C. Schulz, M. Gartner, and D. Hausheer, "DDoS detection in P4 using HYPERLOGLOG and COUNTMIN sketches," in *Proc. IEEE/IFIP Netw. Operations Manag. Symp.*, May 2023, pp. 1–6.
[93] L. Jain and V. U., "P4 based switch centric flow table overflow detection and mitigation in data plane devices," in *Proc. 5th Int. Conf. Recent Adv. Inf. Technol. (RAIT)*, Mar. 2023, pp. 1–6.
[94] NI Standards and Technology. *Computer Network Defense (CND)*. Accessed: Aug. 22, 2023. [Online]. Available: https://csrc.nist.gov/glossary/term/computer_network_defense
[95] National Institute of Standards and Technology. *Mitigation*. Accessed: Mar. 7, 2023. [Online]. Available: https://csrc.nist.gov/glossary/term/mitigation
[96] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: Methods, practices, and solutions," *Arabian J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, Feb. 2017.
[97] *Mininet: An Instant Virtual Network on Your Laptop (or Other PC)*. Accessed: Feb. 15, 2023. [Online]. Available: http://mininet.org/
[98] *ONOS+P4 Tutorial for Beginners*. Accessed: Feb. 16, 2023. [Online]. Available: https://wiki.onosproject.org/pages/viewpage.action?pageId=16122675
[99] S. S. Ottenburger and U. Ufer, "Smart cities at risk: Systemic risk drivers in the blind spot of long-term governance," Wiley, USA, Tech. Rep., 2023, doi: 10.1111/risa.14102.
[100] MITRE D3FEND. (2023). *D3fend Artifacts*. [Online]. Available: https://d3fend.mitre.org/
[101] F. Paolucci, F. Cugini, and P. Castoldi, "P4-based multi-layer traffic engineering encompassing cyber security," in *Proc. Opt. Fiber Commun. Conf. Expo. (OFC)*, Mar. 2018, pp. 1–3.
[102] M. Latah and L. Toker, "Artificial intelligence enabled software-defined networking: A comprehensive overview," *IET Netw.*, vol. 8, no. 2, pp. 79–99, Mar. 2019.
[103] S. A. Gutiérrez, J. W. Branch, L. P. Gaspary, and J. F. Botero, "Watching smartly from the bottom: Intrusion detection revamped through programmable networks and artificial intelligence," 2021, *arXiv:2106.00239*.
[104] N. Feamster and J. Rexford, "Why (and how) networks should run themselves," in *Proc. Appl. Netw. Res. Workshop*, NY, NY, USA, Jul. 2018, p. 20.
[105] A. S. Jacobs, R. J. Pfitscher, R. A. Ferreira, and L. Z. Granville, "Refining network intents for self-driving networks," in *Proc. Afternoon Workshop Self-Driving Netw.*, Aug. 2018, pp. 15–21.
[106] J. Luo, W. Liu, M. Tan, and H. Chen, "Binary neural network with P4 on programmable data plane," in *Proc. 18th Int. Conf. Mobility, Sens. Netw. (MSN)*, Dec. 2022, pp. 960–965.
[107] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognit.*, vol. 105, Sep. 2020, Art. no. 107281.
[108] Q. Kang, J. Xing, and A. Chen, "Automated attack discovery in data plane systems," in *Proc. 12th USENIX Workshop Cyber Secur. Experimentation Test*, 2019, pp. 1–5.

CHRISTIAN GARZÓN received the B.Eng. degree in telecommunications engineering from Universidad de Antioquia (UdeA), Medellín, Colombia, in 2021, where he is currently pursuing the M.Eng. degree in telecommunications engineering. He is also with the GITA Laboratory, a research group. His main research interests include software defined networks, programmable data planes, and cybersecurity.

SANTIAGO RÍOS-GUIRAL received the B.Eng. degree in electronic engineering from Universidad de Antioquia (UdeA), Medellín, Colombia, in 2022. He is currently pursuing the M.Eng. degree with the Telecommunications Program. He is also a current member of the GITA Laboratory, a research group, UdeA. His research interests include cybersecurity, software defined networks, and programmable data plane.

ERWIN LEAL received the B.Eng. degree in electronic engineering, the M.Eng. degree in engineering with a major in telecommunications, and the Ph.D. degree in electronic engineering from Universidad de Antioquia (UdeA), in 2001, 2009, and 2019, respectively. He is currently a Full Professor with the Electronic Engineering Department, UdeA, where he has been a Faculty Member, since 2009. Also, he was a full-time Faculty Member with Universidad de Santo Tomás (2001–2009). He is also with the GITA Laboratory, a research group, UdeA. His research interests include software defined networks, telematic services, NFV, and network security.

SERGIO A. GUTIÉRREZ received the Ph.D. degree in computer science from Universidad Nacional de Colombia, Medellín, in 2018. He is currently a full-time Professor and a Researcher with the GITA Laboratory, a research group, Universidad de Antioquia (UdeA). His research interests include computer networks, security in computer networks, data center networks, software defined networks, and programmable data planes and application of pattern recognition and machine learning to computer networks.

JUAN F. BOTERO (Member, IEEE) received the Ph.D. degree in telematics engineering from the Technical University of Catalonia, Spain, in 2013. He is currently an Associate Professor with the Electronics and Telecommunications Engineering Department, Universidad de Antioquia (UdeA), Colombia. In 2013, he joined the GITA Laboratory, a research group, UdeA. His main research interests include quality of service, software defined networks, NFV, cybersecurity, network management, and resource allocation.