MARS: Defending TCP Protocol Abuses in Programmable Data Plane

Dan Tang, Chenguang Zuo, Jiliang Zhang, *Senior Member, IEEE*, Keqin Li, *Fellow, IEEE*, Qiuwei Yang, and Zheng Qin, *Member, IEEE*

***Abstract*—The TCP protocol's inherent lack of built-in security mechanisms has rendered it susceptible to various network attacks. Conventional defense approaches face dual challenges: insufficient line-rate processing capacity and impractical online deployment requirements. The emergence of P4-based programmable data planes now enables line-speed traffic processing at the hardware level, creating new opportunities for protocol protection. In this context, we present MARS - a data plane-native TCP abuse detection and mitigation system that synergistically combines the Beaucoup traffic monitoring algorithm with artificial neural network (ANN) based anomaly detection, enhanced by adaptive heuristic mitigation rules. Through comprehensive benchmarking against existing TCP defense mechanisms, our solution demonstrates 12.95% higher throughput maintenance and 25.93% improved congestion window recovery ratio during attack scenarios. Furthermore, the proposed framework establishes several novel evaluation metrics specifically for TCP protocol protection systems.**

***Index Terms*—Attack mitigation, TCP protocol abuse, machine learning, heuristic rule, programmable data plane.**

I. Introduction

As INTERNET technology develops and has been widely applied, computer networks play increasingly significant roles in modern life. Emerging paradigms such as cloud-edge computing [1], IoT ecosystems [2], [3], and large language models (LLMs) impose unprecedented demands on networks, necessitating high-throughput architectures for big data processing and intelligent resource orchestration. However, traditional network infrastructures suffer systemic limitations: heterogeneous device interoperability, absence of global network-state awareness, insufficient real-time traffic visualization, and rigid bandwidth allocation mechanisms [4]. The resource needs and functional support required for emerging technologies are no longer met by these barriers, which also hinder the implementation of network services. In response to the evolution of Internet technology, the P4 (Programming Protocol Independent Packet Processors) language and PISA architecture have been proposed to facilitate data plane programmability. P4 language is a protocol-independent programming language for processing data packets. It can be used to customize the parsing process of data packets, as well as the process and conditions for forwarding and discarding data packets, allowing for independent programming of the data plane protocol.

Programmable data planes are a versatile technology that can address many challenges. Numerous network protocols are vulnerable to attacks from unscrupulous hosts who exploit their vulnerabilities. The flaws in TCP congestion control could potentially lead to manipulation attacks [5], which are meant to incite actions that either benefit malevolent users or injure normal users, instead of hijacking or crashing normal users. The optimistic ACK attack is an example of TCP protocol abuse, in which a malicious TCP receiver sends an acknowledgement before receiving a packet, allowing the sender to transmit the packet faster. When the quantity of malicious streams exceeds a particular threshold, the impact of optimistic ACK attacks may become apparent. Another example is when the TCP receiver fails to adhere to ECN specifications and disregards congestion notifications from the switch, which are intended to regulate the sender and maintain a normal data transfer rate during periods of congestion.

Currently, most research on defending against optimistic ACK attacks and ECN abuse uses traditional methods. However, there are few research projects addressing the defense against optimistic ACK attacks and ECN abuse in the programmable data plane. Additionally, almost all traditional research methods require modifying the TCP protocol or terminal hosts. This complicates the deployment of these methods for detecting and mitigating optimistic ACK attacks and ECN abuse, not to mention the significant expenses involved. It's pretty simple to modify the packet processing and forwarding logic, and deploy optimistic ACK attack and ECN abuse defense methods using the P4 language. This can be achieved without modifying the protocol or terminal host implementation, which can accelerate packet processing

Received 7 August 2024; revised 26 March 2025 and 13 June 2025; accepted 13 June 2025. Date of publication 17 June 2025; date of current version 7 October 2025. This work was supported in part by the National Natural Science Foundation of China (#62472153) and the Natural Science Foundation General Project of Chongqing under Grant No.CSTB2022NSCQ-MSX1378. The associate editor coordinating the review of this article and approving it for publication was Y. Cao. *(Corresponding author: Chenguang Zuo.)*

Dan Tang is with the College of Science and Electronic Engineering, Hunan University, Changsha 410082, China, and also with the Research Institute of Hunan University in Chongqing, Chongqing 401120, China (e-mail: Dtang@hnu.edu.cn).

Chenguang Zuo, Qiuwei Yang, and Zheng Qin are with the College of Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: chenguangzuo@hnu.edu.cn; yangqiuwei@hnu.edu.cn; zqin@hnu.edu.cn).

Jiliang Zhang is with the College of Semiconductors (College of Integrated Circuits), Hunan University, Changsha 410082, China (e-mail: zhangjiliang@hnu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TNSM.2025.3580467

and minimize CPU usage of the controller during the attack. Furthermore, detection and mitigation methods implemented using the P4 language can be easily modified and recompiled anytime, providing high flexibility. Inspired by [6], this article employs the P4 language in programmable data plane to address common TCP protocol abuse issues. We propose MARS to detect and mitigate TCP protocol abuse. Firstly, we utilize the Beaucoup system to identify the types of TCP protocol abuse in the network. Then, suspicious flows are identified based on the characteristics corresponding to the abuse type, and packet loss or modification operations are performed. Specifically, we have made the following contributions:

- We propose MARS, which can simultaneously deal with multiple TCP protocol abuse attacks.
- Compared to traditional TCP protocol abuse defense methods, defense methods in the programmable data plane can handle traffic and provide mitigation at line-speed.
- There are relatively few research projects on optimistic ACK attacks and ECN abuse in the programmable data plane, and the problem with present schemes is that the evaluation indicators are not precise enough. This article analyzed the impact of different experimental parameters on attack and mitigation effects, and quantified the evaluation indicators. When compared to the current techniques, the mitigation effect has been improved by about 10%.

This is how the remainder of the article will be structured. Chapter 2 primarily elucidates the research focus of the paper and discusses the research background and its significance. In the third chapter, an overview of TCP protocol abuse is provided, along with several examples of TCP protocol abuse. The MARS design is shown in Chapter 4. The usage of our approach in a number of TCP protocol abuses is covered in Chapter 5, and experimental findings are shown in Chapter 6. Finally, Chapter 7 of this article summarizes the research conducted in the paper and concludes the direction of future work.

II. Related Work

P4 has emerged as a novel packet processing technique that can be used with SDN technology [7], allowing for data plane programmability. P4 allows for custom packet headers, packet forwarding, and modification operations. New network functions, like traffic visualization and global network state monitoring, may be implemented thanks to programmability. The programmable technology of the data plane offers new ideas for network attack detection and mitigation. Numerous research on protecting programmable data planes from network attacks has been carried out recently.

A. Network Attack Detection and Defense Methods Based on Traditional SDN

Compared with traditional networks, SDN offers a new approach for detecting and defending against TCP protocol issues through online deployment [4]. Peng et al. [8] used the ADVICE mechanism to detect DDoS attacks and reduced the workload of SDN controller. Humayun et al. [9] concluded that early expulsion has a positive impact on preventing flow table overflow based on the analysis of three schemes. Yue et al. [10] used the Bayesian voting mechanism for deep detection of LDoS attacks. Soylu et al. [11] described a way to protect against table overflow attacks using Network Function Virtualization (NFV). Ahuja et al. [12] utilized a new hybrid machine learning model to categorize traffic as benign or malicious. The fairness of resource allocation in cross domain collaboration scenarios is equally crucial. The fairness optimization mechanism in federated learning, such as FedUP [13], provides new ideas for distributed resource scheduling in defense systems by balancing efficiency and fairness. Novaes et al. [14] developed a system to defend against adversarial DoS attacks by employing an adversarial training method. Zhao et al. [15] suggested using stochastic differential equations to counteract overflow attacks.

Tang et al. [16] determined LDoS attacks by analyzing traffic performance and identified the sources and victims of the attacks based on flow features. [17] developed an intelligent attack targeting the flow table, which can degrade the performance of honest users. Liu et al. [18] mentioned a DDoS attack defense approach based on fast analysis of all packets in a sequence. Tang et al. [19] upgraded the symbolic aggregate approximation technique to identify LDoS attacks. Nallusamy et al. [20] created a sketch-based method for reducing entries, which involves periodic analysis, accelerating the aggregation of recurring entries, and creating a compact flow table using the Huffman coding compression technique. Tang et al. [21] brought Learning-To-Rank, an information retrieval technique, to SDN security and threat identification. Although traditional SDN separates control from data forwarding, it cannot achieve protocol-independent packet processing and high-speed packet forwarding. The emergence of programmable data planes has compensated for this.

B. Network Attack Detection and Defense Methods Based on Programmable Data Plane

Da Silveira Ilha et al. [22] developed a fine-grained, low-latency network traffic analysis method called EUCLID based on P4 language to mitigate DDoS attacks. Ding et al. [23] concluded a DDoS identification implementation based on P4 programming that utilizes normalized network traffic entropy changes, overcoming the limitations of P4 on many arithmetic operations. To optimize DDoS defense methods, González et al. [24] explained an in-network, collaborative pushback mechanism. Febro et al. [25] implemented a virtualized network function to secure the network edge. Jain et al. [26] mentioned a technique for identifying LOFT attacks in the data plane using integrated machine learning.

In [27], Kim et al. proposed an optimized sketch method that achieves perfect third-order feature extraction and can be applied to multiple security use cases. Carvalho et al. [28] achieved DataPlane-ML, which includes a credit algorithm to limit the activity of malicious nodes. González et al. [29] designed a mixed approach that pushes back malicious

traffic from the victim to protect against DDoS attacks. Zhou and Gu [30] implemented Cerberus, an efficient and effective in-network security monitoring system to protect in-network monitoring system. Kfoury et al. [31] introduced P4Tune, which uses passive PDPs to process packets and applies intelligent algorithms to legacy devices. Zheng et al. proposed a series of methods for deploying general machine learning models in programmable data planes, providing powerful tools for attack identification in programmable data planes [32], [33]. The functions of these methods are all implemented within the programmable data plane, effectively achieving the separation of anomaly detection and mitigation mechanisms from the controller. However, there are still significant challenges in the actual implementation process due to the restricted instruction set of the data plane programming language.

C. TCP Protocol Abuse Detection and Defense Methods

Xie et al. [34] achieved quick and accurate anomaly detection based on bilateral principal component analysis. Chen et al. [35] applied an effective security strategy to defend against TCP session hijacking by incorporating cryptography, signature technology, etc. Laraba et al. [6] used P4 programming language to program the data plane and established an EFSM model to resist improper TCP hosts in switches, in order to reduce TCP protocol abuse and defend against network attacks. Şahin and Demirci [36] implemented updatable bloom filters and connection pooling functions on programmable switches, effectively protecting against SYN flood.

Zhou et al. [37] devised an optimal thresholding mechanism to filter anomalous flow through the LSTM model. In [38], the author explored a new IoT dataset and detected DDoS attacks within it. Bhale et al. [39] mentioned OPTIMIST, which decreases the superfluity of IDS nodes and energy consumption by the K-coverage approach. Sharifian et al. [40] applied gravitational fixed radius nearest neighbor to choose a feature subset of DDoS attacks. Yungaicela-Naula et al. [41] provided a framework incorporating MTD, DL, RL, and NFV to defend against LDoS attacks. Dai et al. [42] designed Scotch to achieve high control plane capacity, improving the extensibility and elasticity of SDN networks. Tang et al. [43] utilized a port detection strategy to identify LDoS attacks.

Currently, most defense methods focus on DoS attacks, while limited research specifically targets optimistic ACK attacks and ECN abuse. And there are few evaluating metrics for detecting and mitigating method [6] in the programmable data plane. More effective methods for detecting and mitigating optimistic ACK attacks and ECN abuse need to be proposed. Moreover, the detection and mitigation methods for these two attacks in traditional networks typically involve modifying the protocol or terminal host implementation. This can result in time delays and may even lead to a loss of normal throughput when using these methods. Optimistic ACK attack and ECN abuse defense based on programmable data planes can reduce costs and improve efficiency to a certain extent.

III. Overview of TCP Protocol Abuse

TCP protocol can be abused in various types of attacks, including SYN Flood attack [36], TCP connection breach attack, DOS attack, etc. These attacks exploit specific characteristics of the TCP protocol, causing attacks or abuse on the network. Due to the fact that TCP protocol abuse can cause network performance degradation and result in bandwidth loss for normal flows, this article focuses on several examples of TCP protocol abuse, to demonstrate the efficiency of our approach.

A. Protocol Manipulation Attacks

TCP protocol abuse attacks strategically exploit inherent protocol vulnerabilities and information asymmetry between individual participants, coercing compliant nodes into deviating from normal congestion control mechanisms.

Optimistic ACK attack: Optimistic ACK attack leverages the characteristics of congestion control algorithms, which allow the receiver to control the rate. The attacker sends ACKs for segments that have not yet been received, causing the sender to mistakenly assume that the network conditions are better than they are, thereby increasing the sending rate. Malicious TCP connections can deplete the bandwidth of intermediate chains, leading to degradation of network performance. In this manner, parallel attack botnets can lead to widespread and sustained congestion crashes. Furthermore, optimistic ACK attacks have the potential to unjustly distribute a greater portion of the bandwidth by causing bad connections to share bottleneck links with legal TCP connections. Although the arrival time of the optimistic ACK has been expedited, the ACK number remains the same as under normal circumstances, thus increasing the concealment of the attack.

ECN abuse: ECN abuse exploits vulnerabilities in TCP congestion control. Under normal circumstances, when congestion occurs, the receiver should update the value of the ECE flag. Hence, an error occurs when the receiving host fails to change the ECE flag after the switch notifies the terminal host of congestion. As a result, the sender will not decrease the transmission rate, leading to a decline in network performance.

ACK division attack: While TCP's congestion management algorithm uses segments to regulate the transmission rate, its error control method uses bytes. ACK segmentation attacks exploit this by dividing the acknowledgement of a data segment into multiple smaller ACKs, thereby inducing the sender to increase the congestion window faster. Specifically, when an attacker receives a data segment containing N bytes, they do not send an ACK that acknowledges all N bytes, but instead split it into M ACKs, with each ACK only acknowledging a portion of the data $(M < N)$. In this way, the sender will mistakenly believe that more data has been received, thereby increasing the congestion window faster.

B. DoS Attack

A sort of network attack known as denial of service (DoS) aims to prevent legitimate users from accessing or using the target system by making it incapable of providing services.

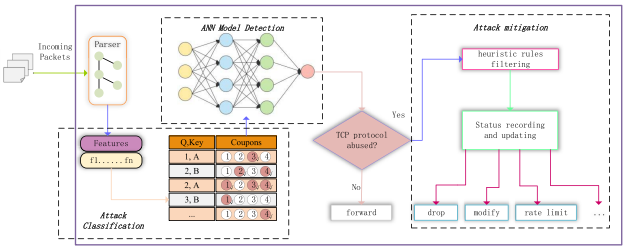


Fig. 1. The Overview of proposed system.

TABLE I
PARAMETERS DEFINED FOR DIFFERENT QUERIES

Attack Type	Key	Attribute	Threshold
Optimistic ACK	{srcIP, dstIP}	sequence step size	4200
ECN abuse	{srcIP, dstIP}	queue length	10
ACK division	{srcIP, dstIP}	ack number if load==0	400
SYN flood	{dstIP, dstPort}	{srcIP, srcPort} if TCP SYN	1500
Low-rate Dos	{srcIP, dstIP}	packet length	200

DoS attacks achieve this goal by consuming the resources of the target system or by disrupting its normal operation.

SYN flood attack: SYN flood attack is a DoS method, mainly used to attack network services based on TCP protocol, such as Web servers, mail servers, etc. This attack exploits a vulnerability in the TCP protocol's three-way handshake mechanism by sending many forged SYN requests to consume the resources of the target server, resulting in legitimate user connection requests being unable to be processed properly.

LDoS attack: Low-Rate Denial of Service (LDoS) is a type of attack against network services that consumes network resources by sending intermittent burst traffic, resulting in a decrease in performance or unavailability of the target service. Unlike traditional DoS attacks, LDoS attacks do not require sending large quantity of data packets, but instead utilize adaptive mechanisms in network protocols such as TCP's congestion control mechanism or HTTP's request-response mechanism to occupy network resources maliciously. This attack method is more covert, difficult to detect, and consumes network resources more efficiently.

IV. PROPOSED SYSTEM DESIGN

This section describes our proposed architecture (see Fig. 1), which consists of three modules: (i) attack classification, (ii) ANN deployment, and (iii) heuristic rule mitigation strategy. The workflow of the entire system is shown in the Fig. 1.

A. Attack Classification

There are various types of TCP protocol abuse attacks. To determine which attacks exist in the current network traffic, we used the system in [44] to simultaneously detect multiple attacks. The BeauCoup system efficiently detects various network attacks on the data plane through its unique coupon collector mechanism. The core principle is to map key information in network traffic (such as IP address, port number, etc.) into "coupons", and randomly allocate these attributes to a limited number of coupons through a random hash function. When all coupons for a query are collected, the system considers that the corresponding network behavior (such as DDoS attacks, heavy hitters, etc.) has met the preset abnormal conditions and triggers an alert. In the data plane, BeauCoup utilizes the hardware resources of programmable switches to store coupon states through a compact bit vector structure and strictly limits the number of memory accesses per packet processing, ensuring efficient operation in high-throughput and low-latency network environments. This design enables BeauCoup to handle multiple heterogeneous queries simultaneously with limited hardware resources, efficiently detecting various attack behaviors in network traffic. The keys and attributes used in each attack are shown in Table I.

B. Feature Extraction and ANN Deployment

1) Custom Protocol: The P4 data plane programming language has complete protocol independent properties. It not only supports the TCP/IP protocol stack, but also allows users to customize new communication protocols, and designs the specific meanings and bit widths of several fields in the packet header. The protocol number assigned to the custom communication protocol in this study is 0x0651, which contains multiple fields. This protocol has two uses. Firstly, when collecting training data for the detection model, it utilizes the feature field in the header of the protocol to convey the calculated feature values from each attack detection window for model training. In order to adjust the neural network's parameters during training, the output results are stored in the other field, $predict$. Next, in the attack mitigation mechanism, the field $alarm$ of the protocol will convey information about whether the attack has been detected. This information is utilized to assess the efficiency of optimistic ACK attack identification in the experimental part. Finally, to enable direct use of this protocol as a lower-level protocol for Ethernet frames for easy identification at the programmable switch entry parsing point, the last field $ether\_type$ stores the protocol number, making it convenient for programmable switches to directly retrieve specific values and assign them to the corresponding fields of Ethernet frames when reconstructing the header set of data packets.

2) ANN Deployment: Due to the approximately 10% error in the query results, we use an ANN model to further reduce false positives after determining the attack type. In switches equipped with detection modules, ANN is implemented in the ingress pipeline of the data plane. There are two issues to solve when deploying ANN on the data plane: normalizing features and activating functions. Currently, programmable data planes do not support floating-point and division operations. So the feature normalization in the programmable data plane was achieved approximately using the longest prefix matching flow table term.

The table structure in the programmable data plane can store flow rules in the form of binary < match, action >. Based on the native supported longest prefix matching flow table,

```
1  Action normalization1 (int<32> normalized_value)
   {meta.normalized_value1 = normalized_value;}
2  table sequence_normalized {key = {meta.sequence: lpm;}
   actions = {normalization1;}}
3  apply {sequence_normalized.apply();}
```

Fig. 2. P4 Pseudocode for normalization.

each flow table entry stores a prefix and corresponding mask. Convert the mapping relationship between the input and output of the mix-max normalization operation within the min-max constraint range into a series of flow rules that match based on the longest prefix. Each prefix used for matching represents a specific range of feature input values. When the input value matches the prefix, the true min-max normalization result corresponding to the average of the left and right boundaries of this range is used as the output. The premise of doing so is to tolerate the occurrence of minor errors appropriately, and the size of the error is artificially set. In this article, the error size is set to 2^{-7}, which means that a series of continuous and prefix-aggregatable input values with a difference of no more than 2^{-7} between the true min-max normalization results will be used as an independent matching term. Because the P4 data plane programming language does not support the same table structure being instantiated multiple times, a longest prefix matching table was established for each feature normalization operation. The keywords used for matching are the feature values, and the actions performed are to obtain the normalized result values of the features. In programmable switches, the specific code implementation is shown in Fig. 2.

The formula for the activation function can be expressed as Eq. (1), which provides a nonlinear modelling capability for neural networks. For activation functions, because the input value allows negative values, the longest prefix matching cannot be used. Therefore, we adopted a segmentation approach for the activation function. Therefore, the proposed approach divides the activation function into several sections. We then calculated the mean of the dependent variables mapped by the left and right endpoint values of each section to determine the output value of the activation function within the interval. The flow was then divided into two categories: normal and suspicious. If the alarm field value in the received clone data packet is 1, the corresponding stream will be placed in the list of suspicious flows.

$$y = \frac{1}{1 + e^{-x}}, \tag{1}$$

C. Attack Mitigation

When the ANN model identifies flows with abnormal behavior in the network, we use the arithmetic and logical operations provided by P4 language to instantiate heuristic rules in the data plane. As shown in the Fig. 1, the mitigation module includes three steps: heuristic rule filtering, state recording and updating, and hierarchical action execution. In the detection module, some detections focus on streams, while others focus on the overall network situation. For the former scenario, we only need to take direct control measures on the abnormal flow. In the latter case, it is necessary to further confirm the abnormal flows that need to be processed.

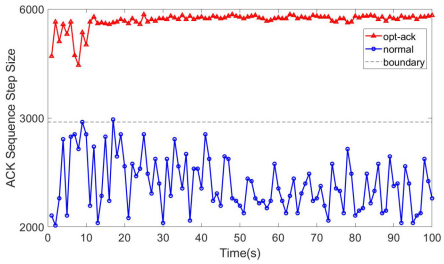


Fig. 3. ACK sequence step size.

When the detection module focuses on the overall network situation, the mitigation module will activate upon receiving alert information, filter abnormal flows by executing rules for each flow, and then record the detection results in the attack state storage (packet loss list or state machine state) and trigger hierarchical actions (such as modifying packets, discarding flows, rate limiting, etc.) to alleviate TCP protocol abuse. In Section IV, we will introduce examples of optimistic ACK attacks, ECN abuse, and LDoS attacks.

V. Application Examples

This section explains how our suggested approach is used in LDoS attacks, ECN misuse, and optimistic ACK attacks.

A. Application to Optimistic ACK

In this chapter, we introduced detection and mitigation methods for optimistic ACK attacks, demonstrating the efficacy of our proposed approach.

1) Selection of Specific Features: During data transmission, the ACK sequence number represents the serial number of the first data byte in the next packet segment that the receiving end expects to receive from the sending end. For the sender receiving ACKs, the difference between the current and last ACK numbers represents the amount of confirmed data each time. A significant difference in the step size of ACK sequence numbers is identified by sampling and analyzing the normal flow and optimistic ACK attack flow data groups. As illustrated in Fig. 3, the optimistic ACK flow sends ACKs faster than the normal flow, resulting in a greater amount of data confirmed each time compared with the normal flow. Therefore, the ACK sequence step size can be used as a feature to determine whether an optimistic ACK attack has occurred.

2) Optimistic ACK Mitigation: To alleviate optimistic ACK attacks, this article establishes a packet loss list, where the index of each flow is the hash of the 4-tuple of the flow. For all suspicious traffic, make a second judgment and exclude normal traffic with high traffic. Owing to the presence of optimistic ACK attacks in the flow, the attacker sends the ACK in advance, resulting in a larger ACK number than normal. The expected ACK number can be obtained by calculating the sequence number and payload values. The expected ACK number is compared to the actual ACK number. If the actual ACK number is greater than the expected ACK number, the flow is determined to be optimistic. Next, this flow is placed

Algorithm 1 Optimistic ACK Attack Defense

Input: *tcp packet*
Output: *outgoing packet*

flow ← hash(srcIP,dstIP,srcPort,dstPort)
suspicious_list.read(attacker,meta.flow)
payload ← IPv4.length - IPv4.header
if $attacker == TRUE$ **then**
  if $meta.payload == 0$ **then**
    $expected\_ack.read(expected\_ackNo, flow)$
    if $meta.expected\_ackNo < ackNo$ **then**
      $expect\_payload.write(0, expected\_ackNo)$
      $expect\_payload.write(1, ackNo)$
      $drop\_list.write(flow, 1)$
    else
      $expected\_ackNo \leftarrow seqNo + payload$
      $expected\_ack.write(flow, expected\_ackNo)$
    end if
  end if
else
  $drop\_list.write(flow, 0)$
end if
$drop\_list.read(attacker1, flow)$
if $attacker1 == TRUE$ **then**
  $drop()$
end if
if $attacker1 == FALSE$ **then**
  Return *outgoing packet*
end if

in the packet loss list for discarding. The entire algorithm is shown in Algorithm 1.

B. Application to ECN Abuse

In response to the problem of limited detection and mitigation strategies for ECN abuse in the field of programmable data plane, this chapter proposes a detection and mitigation mechanism for ECN abuse that is completely deployed in the data plane.

1) Selection of Specific Features: The role of the ECN mechanism is to reduce congestion in the network by lowering packet loss rates, and the occurrence of ECN abuse can lead to network re-congestion. Therefore, the degree of congestion can distinguish between normal situations and those in which ECN abuse occurs. As presented in Fig. 4, the length of the switch egress queue is positively correlated with the degree of network congestion, so the switch egress queue length can be selected as a feature.

2) ECN Abuse Mitigation: In order to alleviate ECN abuse, this article adopts the method of controlling the state transition to control the behavior of switches. First of all, when there is no congestion on the link, its condition is NORMAL. When the link is congested and there is improper behavior by the receiver, the switch takes a mitigating action, and the corresponding state is CONGESTION. In the NORMAL state, the ECE flag of the receiver was set to 0, and in the CONGESTION state, the ECE flag of the receiver was set to 1. Algorithm 2 displays the entire procedure.

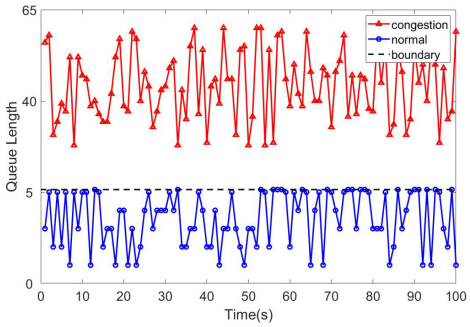


Fig. 4. Switch egress queue length.

Algorithm 2 ECN Abuse Defense

Input: *tcp packet*
Output: *outgoing packet*

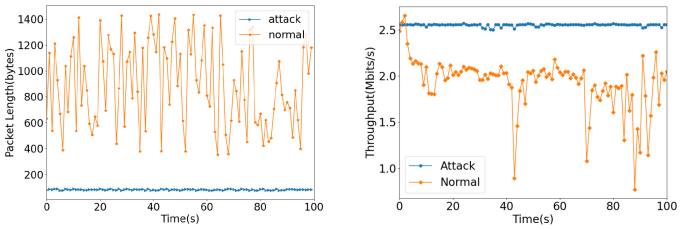
flow ← hash(srcIP,dstIP,srcPort,dstPort)
if $IPv4.ecn \neq 0$ **then**
  $ce.write(flow, 1)$
end if
$ecn\_state.read(cur\_ecn\_state, 0)$
$ce.read(ECT, flow)$
if $cur\_ecn\_state = CONGESTION$ *and* $ACK \neq 0$ *and* $ECT = 1$ **then**
  $tcp.ece \leftarrow 1$
else if $cur\_ecn\_state = NORMAL$ *and* $ACK \neq 0$ *and* $ECT = 1$ **then**
  $tcp.ece \leftarrow 0$
end if
Forward *packet*

C. Application to LDoS Attack

In this section, we apply our method to identify and neutralize LDoS attacks.

1) Selection of Specific Features: LDoS attacks periodically send brief high-intensity pulse streams to cause critical bottleneck links in the network to suddenly enter a congested state, forcing upstream TCP traffic senders to change parameters such as congestion window and congestion avoidance threshold according to congestion control mechanisms. This leads to a sharp drop in the proportion of bandwidth occupied by normal traffic in the link and a decrease in effective link utilization. As shown in Fig. 5a, the size of data packets generated by LDoS attacks is much smaller than normal data packets, so we choose packet length as the attribute. Due to the use of high-frequency UDP packets in LDoS attacks, it can cause significant fluctuations in TCP traffic, as shown in Fig. 5b. So we use TCP traffic distribution entropy as the detection feature. We used the method provided in [45] to calculate entropy on a P4 switch.

2) LDoS Attack Mitigation: In order to mitigate LDoS attacks, this article simultaneously uses packet loss lists and state transition mechanisms. When the mitigation module receives two consecutive alarm messages, it enters the DEFENSE state. If the current number of window flows is greater than THRESHOLD1 and the previous number of window flows is less than THRESHOLD2, then it is



(a) Packet length (b) TCP traffic distribution entropy

Fig. 5. LDoS Attack Features.

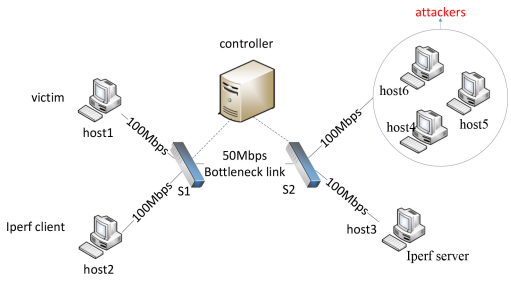


Fig. 6. Experimental topology used in mininet.

Algorithm 3 LDoS Attack Defense

Input: *tcp packet*
Output: *outgoing packet*

flow ← hash(srcIP,dstIP,srcPort,dstPort)
suspicious_list.read(attacker,meta.flow)
if $timestamp > window\_time$ **then**
  $CW \leftarrow CW + 1$
end if
$cur\_flow\_count = cur\_flow\_count + 1$;
if $meta.state == DEFENSE$ **then**
  if $cur\_flow\_count > THRESHOLD1$ and $last\_flow\_count < THRESHOLD2$ **then**
    $drop\_list.write(flow, 1)$
  end if
else if $meta.state == EXIT$ **then**
  if $cur\_flow\_count < THRESHOLD1$ **then**
    $drop\_list.write(flow, 0)$
    $meta.state == SAFE$
  else
    $drop\_list.write(flow, 1)$
  end if
end if
$drop\_list.read(attacker1, flow)$
if $attacker1 = TRUE$ **then**
  $drop()$
end if

determined that the flow is an attack flow. Next, put this flow into the packet loss list for discarding. When the mitigation module receives normal information twice in a row, it enters the EXIT state. If the current number of window flows is less than THRESHOLD1, the flow will no longer be considered malicious. The entire algorithm is shown in Algorithm 3.

VI. Evaluation

A. Setup

1) Settings: **Testbed.** We accomplished the data plane project for defense using the P4 language and deployed it in a bmv2 switch. We conducted experiments using Mininet and p4runtime_switch. Mininet and bmv2 cannot achieve the level of a normal network; therefore, benchmark scenarios were used for comparison in different experiments. However, this provided us with a method to verify the functionality of the solution.

Topology. Fig. 6 shows the experimental topology of the validation experiment. As shown in the figure, hosts h1, h2, and h3 represent legitimate groups of hosts that send or receive legitimate TCP or UDP traffic through bottleneck links. UDP traffic is used as the background traffic. The other hosts represent parties with improper behavior in the TCP protocol. The number of hosts with normal or inappropriate behavior can be adjusted. We deploy the detection and defense modules in switch s2. With a 50Mbps bandwidth, the link connecting switches s1 and s2 was configured as a bottleneck link. The bandwidth of other links is 100Mbps.

Traffic generation. We establish a TCP connection between h2 and h3 as normal traffic. We replay the WIDE dataset as background traffic. In terms of generating attack traffic, optimistic ACK attacks, ECN abuse, and ACK division attacks simulate malicious behavior by running C++ scripts; SYN flood attacks are launched through the hping3 command; LDoS attacks are initiated through Python Scapy.

2) Metrics: To improve the effectiveness, precision, f1-score, and accuracy were used to assess our detection method, which have been widely utilized in previous work [37]. Traffic that contains TCP protocol abuses is treated as a positive sample; traffic without such attacks is treated as a negative sample. The quantity of attack flows that are correctly anticipated is known as the true positive, the quantity of benign flows is known as the true negative, the number of attack flows that are incorrectly predicted is known as the false positive, and the quantity of normal flows is known as the false negative. When TCP protocol abuses occur, the bottleneck link becomes congested; therefore, we use the throughput and congestion window of bottleneck links as evaluation indicators for the mitigation effect.

B. MARS Performance Evaluation

In the evaluation, we tested the effectiveness of detection and mitigation separately.

1) Attack Detection: Tables II and III show a graphical representation of the final test results. In the detection experiment, the programmable switch downstream of the link was loaded with an attack detection program. Three sets of training data were collected in real-time, with 1, 2, and 3 attackers in each set. We detected a total of five TCP protocol abuses and compared them with IIsy [32] and Planter [33]. It can be observed that the ANN detection model deployed under programmable switches has high detection performance for different types of TCP protocol abuse. Through preliminary

TABLE II
ATTACK DETECTION PERFORMANCE (ACCURACY, PRECISION AND F1-SCORE)

Accuracy	Optimistic ACK	ECN Abuse	ACK Division	SYN Flood	LDoS
MARS(ANN)	97.20	95.84	97.33	95.91	96.57
IIsy(SVM)	92.85	93.46	92.22	93.15	94.26
Planter(DT)	95.39	93.21	96.56	94.29	95.54
Planter(RF)	93.54	94.77	94.36	94.15	92.15

(a) Accuracy

Precision	Optimistic ACK	ECN Abuse	ACK Division	SYN Flood	LDoS
MARS(ANN)	98.39	98.65	97.76	98.48	98.57
IIsy(SVM)	96.18	96.93	97.36	97.04	96.83
Planter(DT)	95.37	98.52	93.41	94.69	97.88
Planter(RF)	97.11	94.34	96.52	96.68	97.93

(b) Precision

F1-score	Optimistic ACK	ECN Abuse	ACK Division	SYN Flood	LDoS
MARS(ANN)	98.47	98.24	98.13	98.48	98.07
IIsy(SVM)	96.85	97.14	96.87	97.64	96.54
Planter(DT)	95.91	97.84	95.95	96.02	97.71
Planter(RF)	97.84	95.82	96.87	97.54	97.03

(c) F1-score

screening using the Beaucoup algorithm, a large amount of normal traffic was excluded, reducing the false positive rate.

2) Attack Mitigation: In Fig. 7, we demonstrated the changes in throughput and congestion window under different conditions. When the switch does not respond to improper behavior, the throughput and congestion window of the normal flow suffer a certain degree of loss. In the presence of mitigation measures, throughput and congestion windows were partially restored.

We compared our method with the EFSM [6] method and FSWT [46] method. EFSM is a method that runs on a programmable data plane to monitor and mitigate TCP protocol abuse by implementing complex, stateful functions in the data plane. FSWT is a method running in traditional SDN that describes network traffic by extracting features such as entropy, energy ratio, contrast, and correlation from time-frequency distributions, and uses decision tree models for LDoS attack detection. As shown in Fig. 7, the heuristic approach has a greater throughput recovery ratio compared to the EFSM approach. When the quantity of attackers is twice that of normal hosts, the recovery rate of this method is 12.95 percentage points higher than that of the EFSM method. When the number of attackers is three times that of normal hosts, the recovery rate of this method is 10.92 percentage points higher than that of the EFSM method. Meanwhile, this method also has a higher congestion window recovery rate. In cases where the number of attackers varied, the recovery rates remained close.

We also discussed the impact of different mitigation strategies on the effectiveness of mitigation. As shown in Fig 8, for malicious flows, when using packet loss strategy, the throughput recovery is the highest, and the effect of using rate limiting or redirection is similar, but both are lower than when directly droping packets. Therefore, we choose discard packets as a mitigation method.

3) System Performance: Fig. 9a demonstrates the CPU consumption of switches under MARS architecture

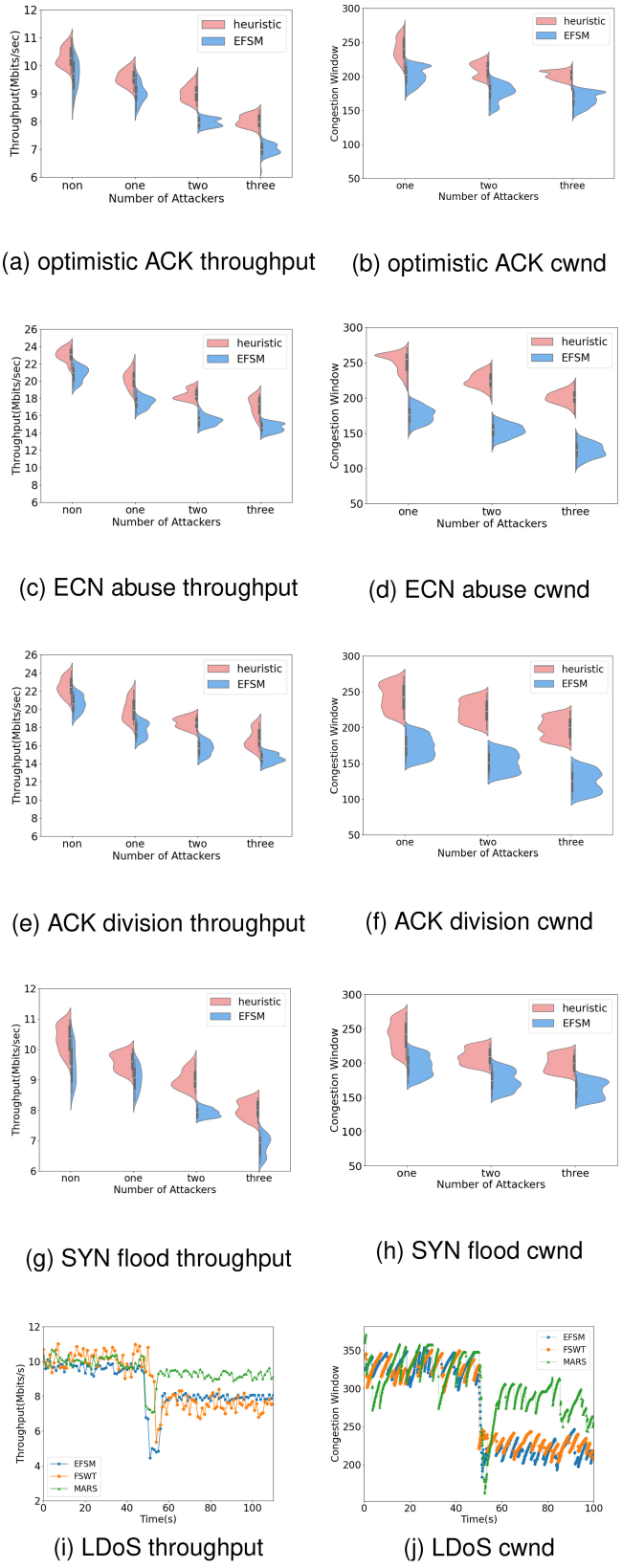


Fig. 7. Attack mitigation effect.

deployment. Experimental results show that the CPU load growth rate is only 1.81% after deploying MARS, which thoroughly verifies the low- cost nature of this solution. Fig. 9b reveals the latency overhead introduced

TABLE III
ATTACK DETECTION PERFORMANCE(FPR AND FNR)

Method	Model	Optimistic ACK		ECN Abuse		ACK Division		SYN Flood		LDoS	
		FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR	FPR	FNR
MARS	ANN	**1.16**	1.44	**1.21**	**2.15**	**1.18**	1.49	**1.33**	**1.52**	**1.75**	**2.43**
IIsy[32]	SVM	2.32	2.48	1.83	2.64	2.75	3.62	1.64	1.76	2.15	3.74
Planter[33]	DT	3.45	3.55	2.96	2.84	3.11	**1.36**	2.47	2.61	2.49	2.46
Planter[33]	RF	2.21	**1.42**	1.95	2.66	2.07	2.78	2.31	1.59	2.18	3.85

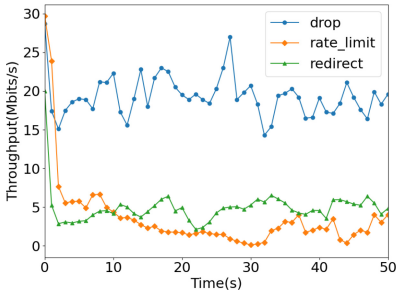


Fig. 8. Throughput under different mitigation strategies.

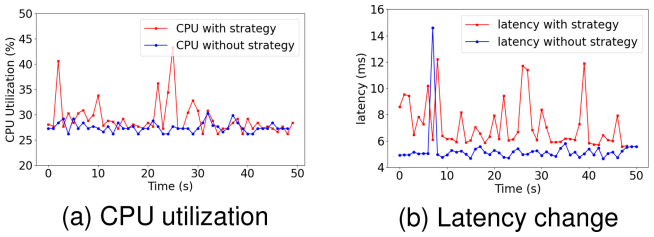


Fig. 9. CPU Utilization and Latency.

TABLE IV
RESOURCE CONSUMPTION PER STAGE

TCAM(KB)	SRAM(MB)	Maximum Number of Parallel Operations	Maximum Number of Parallel Table Lookups
14.08	3.417	2	2

by the defense mechanism: compared with the unprotected scenario, MARS adds an average processing delay of approximately 5ms. In terms of resource utilization, Table IV systematically evaluates the hardware resource consumption characteristics of MARS through a quantitative indicator system including TCAM/SRAM memory usage, maximum concurrent operations, and per-stage table lookup times. Measurement results indicate that this architecture does not impose significant resource overhead on switches while achieving attack protection capabilities.

VII. CONCLUSION

In this article, we propose MARS, a TCP protocol abuse defense system deployed on P4 switches. MARS uses Beaucoup algorithm and ANN model to detect attacks, and uses heuristic rules for mitigation. We demonstrated the effectiveness of our method in defending against several TCP protocol abuse cases. It can effectively detect these abuses and alleviate the bandwidth and congestion window reduction caused by these TCP protocol abuses, which proves that our method can alleviate different types of TCP protocol abuses in programmable data planes without imposing too much burden on switches. Our future direction is to run this method on real commercial switches to consider network conditions in real-world scenarios.

REFERENCES

[1] H. Zhou, Z. Wang, H. Zheng, S. He, and M. Dong, "Cost minimization-oriented computation offloading and service caching in mobile cloud-edge computing: An A3C-based approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1326–1338, May/Jun. 2023.

[2] J. Lu, H. Liu, R. Jia, Z. Zhang, X. Wang, and J. Wang, "Incentivizing proportional fairness for multi-task allocation in crowdsensing," *IEEE Trans. Services Comput.*, vol. 17, no. 3, pp. 990–1000, May/Jun. 2024.

[3] W. Liang, S. Xie, D. Zhang, X. Li, and K.-c. Li, "A mutual security authentication method for RFID-PUF circuit based on deep learning," *ACM Trans. Internet Technol.*, vol. 22, no. 2, pp. 1–20, 2021.

[4] D. Tang, R. Dai, Y. Yan, K. Li, W. Liang, and Z. Qin, "When SDN meets low-rate threats: A survey of attacks and countermeasures in programmable networks," *ACM Comput. Surv.*, vol. 57, no. 4, pp. 1–32, 2024.

[5] N. Kothari, R. Mahajan, T. Millstein, R. Govindan, and M. Musuvathi, "Finding protocol manipulation attacks," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 26–37.

[6] A. Laraba, J. François, S. R. Chowdhury, I. Chrisment, and R. Boutaba, "Mitigating TCP protocol misuse with programmable data planes," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 760–774, Mar. 2021.

[7] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 3431–3446.

[8] J.-C. Peng, Y.-H. Cui, Q. Qian, C. Guo, C.-H. Jiang, and S.-F. Li, "ADVICE: Towards adaptive scheduling for data collection and DDoS detection in SDN," *J. Inf. Secur. Appl.*, vol. 63, Dec. 2021, Art. no. 103017.

[9] U. Humayun, M. Hamdan, and M. N. Marsono, "Early flow table eviction impact on delay and throughput in software-defined networks," in *Proc. 11th IEEE Int. Conf. Control Syst., Comput. Eng. (ICCSCE)*, 2021, pp. 7–12.

[10] M. Yue, Q. Yan, Z. Lu, and Z. Wu, "CCS: A cross-plane collaboration strategy to defend against LDoS attacks in SDN," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 3, pp. 3522–3536, Jun. 2024.

[11] M. Soylu, L. Guillen, S. Izumi, T. Abe, and T. Suganuma, "NFV-Guard: Mitigating flow table-overflow attacks in SDN using NFV," in *Proc. IEEE 7th Int. Conf. Netw. Softw. (NetSoft)*, 2021, pp. 263–267.

[12] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, "Automated DDOS attack detection in software defined networking," *J. Netw. Comput. Appl.*, vol. 187, Aug. 2021, Art. no. 103108.

[13] H. Liu, J. Lu, X. Wang, C. Wang, R. Jia, and M. Li, "FedUP: Bridging fairness and efficiency in cross-silo federated learning," *IEEE Trans. Services Comput.*, vol. 17, no. 6, pp. 3672–3684, Nov./Dec. 2024.

[14] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença Jr, "Adversarial deep learning approach detection and defense against DDoS attacks in SDN environments," *Future Gener. Comput. Syst.*, vol. 125, pp. 156–167, Dec. 2021.

[15] X. Zhao, Q. Wang, Z. Wu, and R. Guo, "Method for overflow attack defense of SDN network flow table based on stochastic differential equation," *Wireless Pers. Commun.*, vol. 117, pp. 3431–3447, Apr. 2021.

[16] D. Tang, Y. Yan, S. Zhang, J. Chen, and Z. Qin, "Performance and features: Mitigating the low-rate TCP-targeted DoS attack via SDN," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 428–444, Jan. 2022.

[17] M. Yu, T. Xie, T. He, P. McDaniel, and Q. K. Burke, "Flow table security in SDN: Adversarial reconnaissance and intelligent attacks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 6, pp. 2793–2806, Dec. 2021.

[18] X. Liu, J. Ren, H. He, B. Zhang, C. Song, and Y. Wang, "A fast all-packets-based DDoS attack detection approach based on network graph and graph kernel," *J. Netw. Comput. Appl.*, vol. 185, Jul. 2021, Art. no. 103079.

[19] D. Tang, Y. Yan, C. Gao, W. Liang, and W. Jin, "LtRFT: Mitigate the low-rate data plane DDoS attack with learning-to-rank enabled flow tables," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 3143–3157, 2023.

[20] P. Nallusamy, T. R. Reshmi, and M. Krishnan, "AGFT: Adaptive entries aggregation scheme to prevent overflow in multiple flow table environment," *Concurr. Comput., Pract. Exp.*, vol. 34, no. 1, 2022, Art. no. e6491

[21] D. Tang, S. Wang, B. Liu, W. Jin, and J. Zhang, "GASF-IPP: Detection and mitigation of LDoS attack in SDN," *IEEE Trans. Services Comput.*, vol. 16, no. 5, pp. 3373–3384, Sep./Oct. 2023.

[22] A. da Silveira Ilha, Â. 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.

[23] 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 Manag.*, vol. 18, no. 2, pp. 1191–1202, Jun. 2021.

[24] 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. Manage. (IM)*, 2021, pp. 393–401.

[25] 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.

[26] L. Jain, U. Venkanna, and S. Vollala, "FTSheild: An intelligent framework for LOFT attack detection and mitigation with programmable data plane," *Expert Syst. Appl.*, vol. 265, Mar. 2025, Art. no. 125865.

[27] S. Kim, S. M. M. Mirnajafizadeh, B. Kim, R. Jang, and D. Nyang, "SketchFeature: High-quality per-flow feature extractor towards security-aware data plane," in *Proc. ISOC NDSS*, 2025, pp. 1–17.

[28] R. N. Carvalho, L. R. Costa, J. L. Bordim, and E. A. Alchieri, "DataPlane-ML: An integrated attack detection and mitigation solution for software defined networks," *Concurr. Comput., Pract. Exp.*, vol. 35, no. 19, 2023, Art. no. e7434.

[29] L. A. Q. González, L. Castanheira, J. A. Marques, A. E. Schaeffer-Filho, and L. P. Gaspary, "Bungee-ML: A cross-plane approach for a collaborative defense against DDoS attacks," *J. Netw. Syst. Manage.*, vol. 31, no. 4, p. 77, 2023.

[30] H. Zhou and G. Gu, "Cerberus: Enabling efficient and effective in-network monitoring on programmable switches," in *Proc. IEEE Symp. Security Privacy (SP)*, 2023, pp. 1–16.

[31] E. Kfoury, J. Crichigno, and E. Bou-Harb, "P4Tune: Enabling programmability in non-programmable networks," *IEEE Commun. Mag.*, vol. 61, no. 6, pp. 132–138, Jun. 2023.

[32] C. Zheng et al., "IIsy: Hybrid in-network classification using programmable switches," *IEEE/ACM Trans. Netw.*, vol. 32, no. 3, pp. 2555–2570, Jun. 2024.

[33] C. Zheng et al., "Planter: Rapid prototyping of in-network machine learning inference," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 54, no. 1, pp. 2–21, 2024.

[34] K. Xie et al., "On-line anomaly detection with high accuracy," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1222–1235, Jun. 2018.

[35] M. Chen, F. Dai, B. Yan, and J. Cheng, "Encryption algorithm for TCP session hijacking," in *Proc. 6th Int. Conf. Artif. Intell. Secur. (ICAIS)*, Hohhot, China, Jul. 2020, pp. 191–202.

[36] M. E. Şahin and M. Demirci, "ConPoolUBF: Connection pooling and updatable bloom filter based SYN flood defense in programmable data planes," *Comput. Netw.*, vol. 231, Jul. 2023, Art. no. 109802.

[37] H. Zhou, Y. Zheng, X. Jia, and J. Shu, "Collaborative prediction and detection of DDoS attacks in edge computing: A deep learning-based approach with distributed SDN," *Comput. Netw.*, vol. 225, Apr. 2023, Art. no. 109642.

[38] Kamaldeep, M. Malik, and M. Dutta, "Feature engineering and machine learning framework for DDoS attack detection in the standardized Internet of Things," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 8658–8669, May 2023.

[39] P. Bhale, D. R. Chowdhury, S. Biswas, and S. Nandi, "OPTIMIST: Lightweight and transparent IDS with optimum placement strategy to mitigate mixed-rate DDoS attacks in IoT networks," *IEEE Internet Things J.*, vol. 10, no. 10, pp. 8357–8370, May 2023.

[40] Z. Sharifian, B. Barekatain, A. A. Quintana, Z. Beheshti, and F. Safi-Esfahani, "Sin-Cos-bIAVOA: A new feature selection method based on improved African vulture optimization algorithm and a novel transfer function to DDoS attack detection," *Expert Syst. Appl.*, vol. 228, Oct. 2023, Art. no. 120404.

[41] N. M. Yungaicela-Naula, C. Vargas-Rosales, and J. A. Pérez-Díaz, "SDN/NFV-based framework for autonomous defense against slow-rate ddos attacks by using reinforcement learning," *Future Gener. Comput. Syst.*, vol. 149, pp. 637–649, Dec. 2023.

[42] Y. Dai, A. Wang, Y. Guo, and S. Chen, "Elastically augmenting the control-path throughput in SDN to deal with Internet DDoS attacks," *ACM Trans. Internet Technol.*, vol. 23, no. 1, pp. 1–25, 2023.

[43] D. Tang, R. Dai, C. Zuo, J. Chen, K. Li, and Z. Qin, "A low-rate DoS attack mitigation scheme based on port and traffic state in SDN," *IEEE Trans. Comput.*, vol. 74, no. 5, pp. 1758–1770, May 2025.

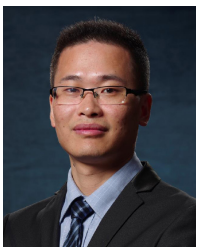
[44] X. Chen, S. Landau-Feibish, M. Braverman, and J. Rexford, "Beaucoup: Answering many network traffic queries, one memory update at a time," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, 2020, pp. 226–239.

[45] D. Tang, B. Liu, K. Li, S. Xiao, W. Liang, and J. Zhang, "PLUTO: A robust LDoS attack defense system executing at line speed," *IEEE Trans. Depend. Secure Comput.*, vol. 22, no. 3, pp. 2855–2872, May/Jun. 2025.

[46] X. Wang, D. Tang, Y. Feng, Z. Qin, B. Xiong, and Y. Liu, "An LDoS attack detection method based on FSWT time–frequency distribution," *Expert Syst. Appl.*, vol. 256, Dec. 2024, Art. no. 125006.

Dan Tang received the Ph.D. degree from the Huazhong University of Science and Technology in 2014. He is an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, and also with the Research Institute of Hunan University in Chongqing, Chongqing, China. His research interests include the areas of computer network security, computer information security, and architecture of future Internet.

Chenguang Zuo received the B.E. degree in computer science and technology from Hunan University, Changsha, China, in 2023, where he is currently pursuing the postgraduate degree with the College of Computer Science and Electronic Engineering. His research interests include programmable data plane and network security.



Jiliang Zhang (Senior Member, IEEE) received the Ph.D. degree in computer science and technology from Hunan University, Changsha, China, in 2015. From 2013 to 2014, he worked as a Research Scholar with the Maryland Embedded Systems and Hardware Security Lab, University of Maryland at College Park, College Park. From 2015 to 2017, he was an Associate Professor with Northeastern University, China. He is currently a Full Professor with Hunan University. He is the Director of the Chip Security Institute of Hunan University, and the Secretary-General of CCF Fault-Tolerant Computing Professional Committee. He has authored more than 60 technical papers in leading journals and conferences. His current research interests include hardware security, integrated circuit design, and intelligent system. He was the recipient of the CCF Integrated Circuit Early Career Award. He is serving as a Steering Member for the Hardware Security Forum of China and a Guest Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS.

Keqin Li (Fellow, IEEE) is a SUNY Distinguished Professor of Computer Science with the State University of New York. He is also a National Distinguished Professor with Hunan University, China. He has authored or coauthored over 850 journal articles, book chapters, and refereed conference papers. He holds over 70 patents announced or authorized by the Chinese National Intellectual Property Administration. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, heterogeneous computing systems, computer networking, and machine learning. He has received several best paper awards. He is among the world's top five most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an Associate Editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He is an AAIA Fellow. He is also a member of the Academia Europaea (Academician of the Academy of Europe).

Qiuwei Yang received the Ph.D. degree in 2008. He is an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His research interests include cloud computing security, mobile security, distributed system security, and network security.

Zheng Qin (Member, IEEE) received the Ph.D. degree in computer software and theory from Chongqing University, China, in 2001. He is currently a Professor of Computer Science and Technology with Hunan University, China. He has accumulated rich experience in products development and application services, such as in the area of financial, medical, military, and education sectors. His main interests include computer network and information security, cloud computing, big data processing, and software engineering. He is a member of the China Computer Federation and ACM.