Flow Table Overflow Attacks in a Software-Defined Network (SDN): A Systematic Review

Aladesote Olomi Isaiah, *Member, IAENG,* Azizol Abdullah, Normalia Samian and Zurina Mohd. Hanapi

***ABSTRACT*—Software-defined networking (SDN) is a modern paradigm leveraging software programmability to enhance communication networks, garnering significant attention and undergoing substantial development due to its diverse applications. One key challenge in SDN lies in managing increasing traffic while avoiding flow table overflow, particularly due to the limited capacity of Ternary Content Addressable Memory (TCAM) in OpenFlow switches. This paper presents a Systematic Literature Review (SLR) that analyzes various approaches to defending against flow table overflow in SDN. Employing a structured approach, we sift through a substantial corpus of research, distilling it into 44 noteworthy articles published from 2015 to the present. We provide an overview of strategies to mitigate flow table overflow attacks, including eviction strategies, dynamic timeout mechanisms, flow rerouting, and aggregated flow entries. Additionally, we analyze mitigation approaches based on deployment strategies, testbed environments, and traffic generation methods. In conclusion, we identify research gaps and challenges, laying the groundwork for future investigations in this domain.**

***Index Terms*—data plane, flow table, flow table attacks, OpenFlow, software-defined network**

I INTRODUCTION

The digital age is steering in an era where the demands for Cloud Computing, Big Data, and the Internet of Things (IoT) are reshaping the landscape of network services. This transformation is driven by the increasing need for large-scale data centers and the exponential growth of big data processing, catalyzing a shift towards more efficient and intelligent networking architectures [1]. Among these, SDN emerges as a revolutionary concept that leverages software programmability to monitor, regulate, and enhance communication networks [2],[3].

Manuscript received September 11, 2023; revised May 07, 2024.

This work was supported and funded by Universiti Putra Malaysia (UPM).

Aladesote Olomi Isaiah is a Ph.D. student in the Department of Computer Communication and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia (e-mail: gs57427@student.upm.edu.my and isaaladesote@fedpolel.edu.ng)

Azizol Abdullah is an Associate Professor in the Department of Computer Communication and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia (e-mail: azizol@upm.edu.my)

Normalia Samian is a Senior Lecturer in the Department of Computer Communication and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia (e-mail: normalia@upm.edu.my)

Zurina Mohd. Hanapi is an Associate Professor in the Department of Computer Communication and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia (e-mail: zurinamh@upm.edu.my)

Furthermore, the flexibility of SDN simplifies the integration of new functionalities into the network, facilitating rapid technological advancements and significant growth [4]–[8]. DN achieves this by dividing networks into layers or planes, including the data plane (forwarding elements) and the control plane (controller), thereby introducing innovation, flexibility, centralization, virtualization, and programmability into networks [9], [10]. However, the implementation of SDN is not without challenges. For instance, OpenFlow-enabled switches, pivotal to SDN architecture, rely on Ternary Content Addressable Memory (TCAM) [11] and TCAM's high cost and exceptionally high power consumption inherently limit its capacity [12]. While commercial SDN switches have made strides in storing hundreds of thousands of flow entries, managing the escalating traffic presents a significant challenge. The necessity to install a substantial number of flow entries to accommodate this growth can lead to flow table overflow. Proactive mechanisms that install flow entries before the arrival of flows offer a potential solution to this issue. Nevertheless, implementing such mechanisms requires a deep understanding of traffic distribution and properties to ensure satisfactory network performance [13]. However, for dynamic applications where flow prediction is impractical, a reactive approach to flow entry installation proves more suitable. The hard timeout approach, tailored for short-lived flows which frequently populate networks, effectively manages such flows. Meanwhile, an idle timeout strategy accommodates other types of flow entries. Many researchers have embraced the flow entry eviction approach based on idle timeout, leading to enhanced flow table utilization and reduced controller involvement in entry removal. Nonetheless, this method may not effectively handle elephant flows [14], [15].

Moreover, flow delegation, a novel approach for addressing flow table capacity constraints, involves dynamically redistributing flow rules from a fully utilized switch to nearby switches with available capacity, a strategy explored by researchers [16]. Flow table overflow occurs when attackers consume the flow tables housing the controller's rules for managing packet flows, resulting in a Denial of Service (DoS) scenario that severely impacts network performance. Several surveys in the literature have addressed various flow table challenges and aspects, such as flow table management, challenges, and solutions [17], enhancing the limited flow table [18],[19].

To this end, earlier SDN review efforts have neglected to delve into approaches for mitigating flow table overflow, as no single survey has comprehensively tackled this issue. As a result, this survey presents a comprehensive examination of flow table overflow, detailing various mitigation approaches.

A. Contributions

Unlike traditional review methodologies, this study adopts a distinctly defined approach by employing a SLR methodology. This rigorous method ensures the inclusion of all high-quality publications and mitigates selection bias. To enhance the identification of relevant research studies, the SLR initiates with a meticulous search technique. Selected articles are scrutinized for solutions pertaining to the study's queries, while unsuitable papers are excluded based on abstract, title, full text, and publication year criteria. The main contributions of this paper are:

1. Identify high-quality research publications on flow table overflow in SDN.
2. Present detailed approaches to combat flow table overflow in SDN, including experimental details.
3. Present future perspective.

The rest of this paper is organized as follows: Section 2 outlines the stages of the SLR. Section 3 presents the overview of SDN, with a focus on the data plane and its forwarding components. Section 4 analyzes various approaches to prevent flow table overflow. Section 5 identifies research gaps. Finally, section 6 concludes the study.

II SURVEY PROTOCOL

A SLR serves to find and appropriately evaluate published articles related to a given research domain, utilizing a well-defined and structured approach. SLR helps in this work to reduce a large volume of papers into a manageable number for informed decision-making on flow table overflow, causes of flow table overflow, flow entry eviction approaches, and the results. Additionally, this review seeks to identify new and future research directions by pinpointing existing gaps in the literature.

B. Research Questions Formulation

A crucial aspect of this study is the design of research questions. Hence, the study addresses the following research questions through a comprehensive assessment and thorough critique of the selected articles:

RQ1: What is the overview of SDN? (Section 3).

RQ2: What constitutes the flow table, and what are the causes and effects of flow table overflow? (Section 4).

RQ3: What are the existing solutions to flow table overflow attacks? (Section 5).

RQ4: What issues have been addressed regarding overflow attacks in a flow table? (Section 6).

RQ5: What are the research gaps in the existing approaches, including their challenges and limitations? (Section 7).

This review focuses on developing and responding to the listed research questions and critically analyzing the various approaches to flow table overflow defense used in SDN. In the first question, RQ1, we briefly provide an overview of SDN. In RQ2, we categorize approaches to detecting, preventing, and mitigating attacks based on the proposed method, testing platform, and proposed technique. In RQ3, we present literature gaps identified in the existing articles and list issues related to flow table overflow, as addressed in RQ4. Additionally, we outline the main research issues that would motivate researchers to conduct this type of study in RQ5.

C. The Search Strategy

The search strategy includes a set of databases to ensure the inclusion of all relevant articles. The search strategy initially consults four digital libraries: ACM, IEEE Xplore, ScienceDirect, and Springer, and concludes with the academic search engine Google Scholar. Including Google Scholar ensures coverage all relevant scientific studies. The keywords used for the search are related to SDN only: “Flow Table Overflow” OR “Flow entries eviction” OR “Prevention of Flow Table Overflow” OR “Mechanism to Prevent Flow Table Overflow” OR “Flow Table Overload”.

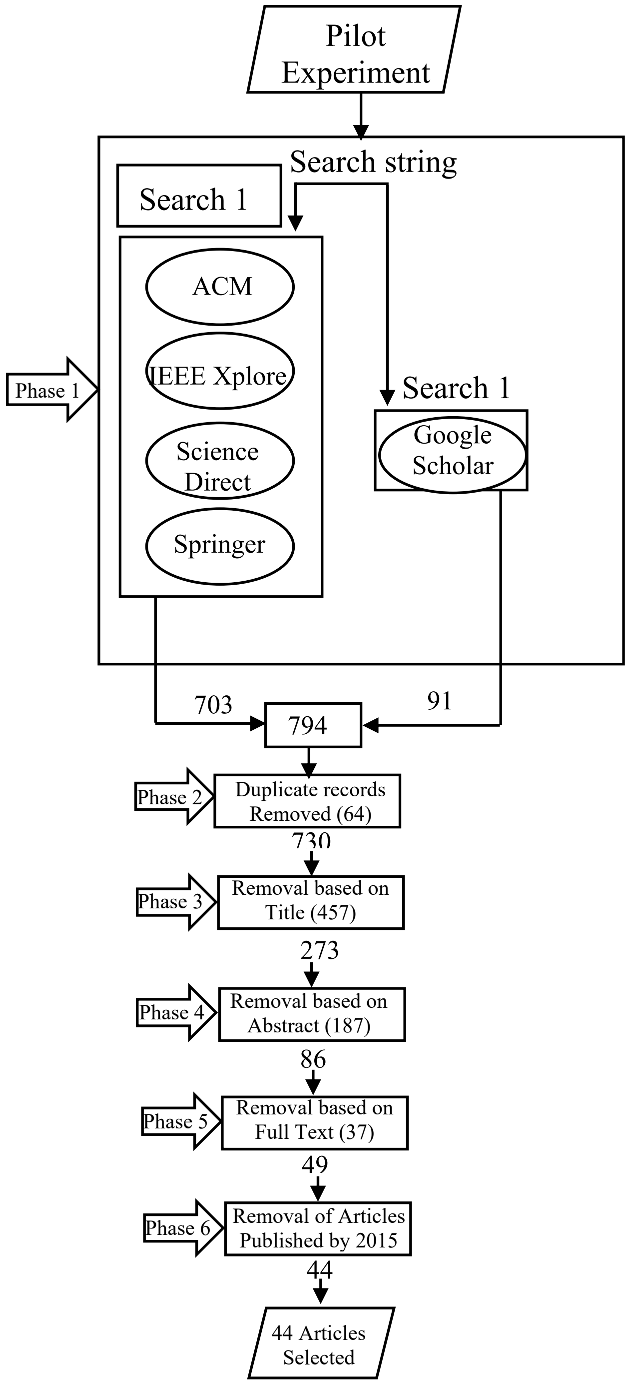

Fig 1. The overall process of the systematic review

D. *The Quality Assessment*

The purpose of the quality evaluation is to identify high-quality research articles. The evaluation process entailed thorough examination of each article, resulting in the selection of 44 research papers. Evaluation criteria encompassed the following: Is the problem statement sufficiently specific? Does the study offer guidance on implementing the research? Is the methodology clearly articulated? Are the results presented in a lucid manner? Can the research effectively address the research questions?

E. *Extraction of Data*

Each academic paper underwent meticulous scrutiny to extract essential information. The research extracted and utilized details including the title, authors, problem addressed, proposed solutions, simulation platform, utilized topology (if applicable), metric employed, motivation, and benefits. Table 1 illustrates the number of research publications at each stage of the evaluation process.

TABLE I
NUMBER OF RESEARCH PAPERS AT EACH PHASE OF THE REVIEW ACTIVITY

Search String	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6
ACM	114	113	53	13	6	6
Google Scholar	91	89	60	36	16	14
IEEE Xplore	389	350	107	31	22	20
Science Direct	123	111	30	3	2	2
Springer	77	67	23	3	3	2
Total	794	730	273	86	49	44

III AN OVERVIEW OF SDN

This section presents the SDN architecture and its layers.

A. SDN Architecture

SDN represents a network technology that facilitates the efficient and effective management of heterogeneous networks. It addresses the limitations of traditional network design, which struggles to accommodate the growing demand for deploying diverse applications with real-time communication requirements. This innovative networking paradigm involves relocating control modules from switches and routers to a centralized entity known as the controller [20],allowing for better resource utilization. This separation allows network administrators and operators to make better use of network resources and deploy resources more easily. Key characteristics of SDN include centralized control management, network automation, virtualization, ease of programmability, openness, and simplified devices [21]. Designers created SDN architecture (Figure 2) to enable the rapid development and deployment of network services and applications. SDN developers write computer network programs or code at the controller to manage the network in an OpenFlow-based SDN deployment. The controller is the brain of the network, which communicates with the Switches' OpenFlow agents to direct how to set up the data plane. Achieving this involves issuing flow modification instructions to insert rules in the forwarding tables [22]. In SDN, there are northbound and southbound application program interfaces (APIs) in addition to the layers [23],[24].

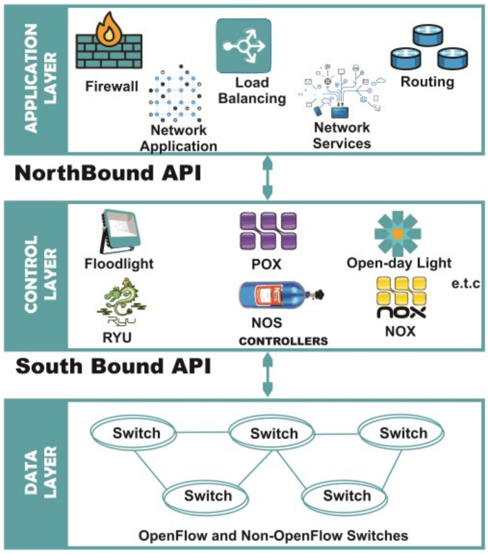

Fig 2. The Architecture of SDN (Adapted from [25] with copyright permission).

1). Data Layer: The data layer, also called the infrastructural layer, consists of network nodes that forward traffic and data. It consists of OpenFlow-enabled switches that manage traffic in line with the controller's instructions.

2). Control Layer: The control plane, also known as the controller, acts as a bridge between the applications and the data plane. Prominent SDN controllers include Ryu [26], POX [27], OpenDayLight [28], Floodlight [29], NOX [30], etc. The northbound interface in SDN links the controller to the application. It also communicates with the switches through the southbound interface [31]. One of the controllers' tasks is to produce flow rules, and the switches route traffic based on the flow rules [32].

3). Application Layer: The application layer contains network applications that help the control plane configure the network to meet these application needs, including network control, quality of network service, monitoring, etc. The layer utilizes the global view provided by the control layer to make recommendations [28] in designing various application-based rules and policies.

4). Southbound API: The Southbound API refers to the interface that allows the control plane and the data plane to communicate with each other. Most SDN implementations use OpenFlow and Network Configuration Protocol (NetConf), of which OpenFlow is the most popular [33].

5). Northbound API: The Northbound API refers to the interface that enables communication between the control plane and the application layer. It facilitates information exchange between the control layer and the applications, with features depending largely on individual network applications.

B. Use Cases of SDN Systems

This section discusses the application and virtualization of SDN, as well as Named Data Network (NDN).

1). Virtualization of SDN: Network Virtualization (NV) enables multiple virtual networks to operate on a single physical network substrate, with each virtual network designed to meet the requirements of specific network services or end-user applications. The goal of network virtualization, an SDN use case, is to address several networking issues, including flexibility, resource usage, and on-demand deployments [34]. Network hypervisors (NH) for SDN provide the necessary features for virtualizing SDNs. These hypervisors logically segregate various virtual SDN networks and their associated tenant controllers [35]. Solutions to SDN-based virtualization are categorized into control plane virtualization, data plane virtualization, and heterogeneous virtualization.

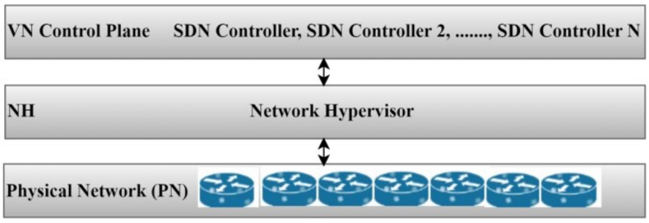

Fig. 3. SDN-NV architecture (Adapted from [36] with copyright permission).

Network virtualization development addresses ossification issues and resolves constraints in communication networks. Its key drivers include rapid service deployment, cost reduction, and quicker network operations. Numerous researchers have introduced Network Virtualization solutions using SDN, such as hypervisor architecture in VeRTIGO [37], Carrier-grade Virtualization Scheme [38], ADVisor [39], FlowVisor [40], AutoSlice [41], and OpenVirteX [42]. However, these approaches encounter challenges with dynamic network changes. HyperFlex, however, enhances resource utilization by virtualizing the hypervisor into separate functions. It implements control plane virtualization using SDN network element software on commodity hardware or software, facilitating variable function virtualization allocation. Hyperflex regulates the receiving rate by discarding the control channel packets [43]. Researchers have implemented NV in cloud settings, representing an advanced application for SDN. They observed that NH contributes to SDN-NV overhead by adding more processing to the control plane. The study measured computational overhead and found that, despite the increasing number of switches, VN, and flows, the overhead from network hypervisors (NH) remains constant [44]. The study in [45] addresses fairness in control channels in SDN-NV scenarios using throughput and setup time as performance metrics. Comparative results show that Sincon reduces interference across control channels in throughput cases and achieves greater improvement in control channels measured by setup time.

2). NDN: NDN emerged to develop an effective Internet alternative, enabling content-centric communication to adapt to the rapidly changing content distribution paradigm [46]. It enhances network communication through data security, in-network caching, and multipath forwarding. CCFS, a controller-based forwarding and caching strategy proposed in [47], addresses inefficiencies in NDN's modules. This architecture focuses on how controllers maintain cache cooperation and how forwarding mechanisms function, outperforming existing algorithms. NDN uses routable content names instead of IP addresses, increasing complexity for applications requiring advanced content delivery. The authors in [48] introduced an Enhanced NDN (ENDN) architecture, which provides content delivery services encoded in the data plane using customized P4 applications.

IV OVERVIEW OF OPENFLOW

This section deals with OpenFlow, flow table, background to flow table attacks, and their causes. Additionally, it introduces a Programmable SBI.

A. Introduction to OpenFlow

OpenFlow is the most widely used Application Programming Interface (API) in SDN technology, owing to its low implementation costs and potential for novel solutions [49]–[51]. It is also the first SDN-specific standard interface, allowing high-performance, granular traffic management across various networking devices [52]. It aims to standardize the communication between a controller (control plane) and the switches (data layer). Moreover, its specification describes how to move control logic from a switch to a controller. The OpenFlow architecture, as depicted in Figure 4, includes features that enable researchers (both in academia and industry) to explore new ideas and test new applications, such as traffic analysis, flow abstraction, and real-world network experiments. These applications were proposed to ease the network in areas like configuration, management, security, virtualization, etc. An OpenFlow switch, also known as a forwarding device, comprises (i) at least a flow table and a group table, which handle packet lookups and forwarding; (ii) at least an OpenFlow channel to an external controller, ensuring secure communication through the OpenFlow protocol with the controller. There are two ways in which the controller can add, update, and delete flow entries in the flow table in an OpenFlow-enabled switch: reactive and proactive [53]. The flow table comprises flow entries, with each entry dictating how packets in a flow are processed and routed. Match fields (rules for matching), counters, and actions make up the flow entries. The match fields' role is to match incoming packets, counters help collect the flow's statistics, and actions reveal how to process a matching packet. Packet header fields are collected and matched against the matching fields section of the flow table entries when the packet arrives at the OpenFlow switch, undergoing a match test. If a matching entry occurs, the switch executes the instructions associated with it (or actions). If not, a table-miss flow occurs. The table-miss entry handles this by discarding the packet, continuing the matching process to the next flow table, or forwarding packet to the controller for further action(s). Pantou/OpenWRT [54], OpenvSwitch (OVS), BOFUSS [55], Indigo, and ofsoftswitch13 are examples of OpenFlow switches. The most popular among them is OVS.

B. Flow Table

A flow table is a part of the SDN switch that store flow rules. An OpenFlow-enabled flow table can be divided into three components: Datapath (hardware layer), Control path (software path), and the OpenFlow protocol. The Datapath, responsible for packet forwarding and lookups, includes at least one flow table or a group table.

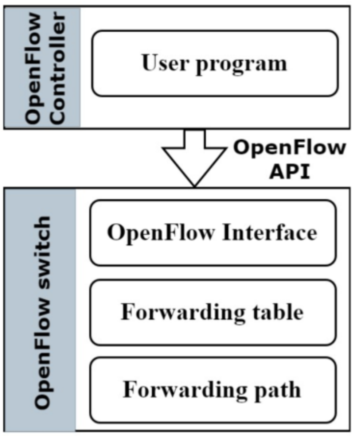

Fig 4. OpenFlow (Adapted from [56] with copyright permission).

The flow table contains flow entries, while the group table holds a collection of group entries. The control path acts as a channel, enabling the switch and the controller to exchange packets and commands via the OpenFlow protocol. An OpenFlow Switch (OF-Switch) stores flow entries in its flow table, which has a limited capacity ranging from a few hundred to thousands of entries, insufficient to handle the millions of flows typical in data center networks. Consequently, the required rules significantly exceed the flow table's capacity. TCAM, a highly efficient associative memory, hosts the flow table. Each flow table within the OpenFlow switch contains flow entries [57], with three fields: Packet Header, Action, and Statistic. Flow entries, used for matching and processing packets, are limited in size [58],[59]. This limitation often results in flow table overflow, making the reinstallation of flow entries challenging and degrading network performance.

1). Background of Flow Table Overflow Attacks: The first flow table attack appeared after the launch of the internet in 1969. In a conventional network, flow table overflow attacks occur in the Media Access Control (MAC) address and routing tables. The former attack happens when an attacker bombards the switch with many MAC addresses from spoofing sources. In contrast, the latter attack happens when a malicious router alerts trustworthy routers to routes to fictitious (imaginary) destinations. Researchers in [60],[61] have raised security concerns about flow table overflow threats interfering with SDN. In [62], the authors grouped the flow table attacks into Brute force, Slow, and Sophisticated. Updating or removing flow table entries involves flow mod messages with additional parameters, hard timeouts, and idle timeouts, the latter being an automatic yet inefficient method for flow table management. Security challenges, such as rule insertion & manipulation and overflow, characterize the flow table. Rule insertion, for example, results in violations of the three security triads. The former violates the three triads (CIA) of security, while the latter compromises the network's availability. In this scenario, the switches become unable to hold additional flow entries, and the controller becomes overwhelmed due to an influx of illegitimate requests from the attackers.

2). The Causes of Flow Table Overflow: Switches equipped with OpenFlow-based technology leverage TCAM for swift flow entry lookup and mask matching. However, the constraints of TCAM, both in terms of capacity and cost, restrict OpenFlow switches to accommodating only a limited number of flow entries, typically in the range of tens of thousands. Consequently, this limitation presents a significant challenge, leading to potential flow table overflows, particularly under scenarios of burst traffic or deliberate overflow attacks, thereby severely impacting network performance. In practice, this means that most flow tables tend to reach their capacity threshold, exacerbating the issue and posing a considerable risk to network functionality. To manage packet handling, the controller, as a pivotal component in the SDN architecture, issues instructions to switches. However, this architecture becomes susceptible to exploitation by malicious actors who leverage its inherent functionality. These bad actors inundate switches with numerous packets, characterized by altered header fields that do not conform to existing flow rules. Consequently, these packets necessitate forwarding to the controller for processing. The controller, being the central decision-maker in the SDN, handles these packets and instructs the switch accordingly, thereby adding instructions to the switch's flow table. However, this influx of illegitimate traffic not only overwhelms the switch's flow table but also hampers its ability to process genuine packets effectively. As a result, the network experiences degradation in performance and efficiency, highlighting the critical need for robust solutions to mitigate the impact of flow table overflows on network operations. [63]. Consequently, legitimate users cannot access the flow table [64]. The limited size of flow entries in a flow table results in flow table overflow and reinstallation of flow entry challenges, degrading network performance. Earlier versions of the OpenFlow specification (1.0–1.3.2) prevented adding new flow entries when a table reached capacity and sent an error message to the controller [65]. Versions 1.4 and higher introduce two solutions for this issue: eviction and vacancy events. Eviction enables the switch to automatically discard less important entries, making space for new ones. It employs techniques like Least Recently Used (LRU), First In First Out (FIFO), or Random selection for this purpose [66]. The choice of which flow entry to remove depends on either the switch's decision or factors like flow entry parameters, the switch's resource allocations, and internal limitations. Vacancy events allow the controller to get an early warning based on a set capacity threshold, enabling proactive measures to prevent table overload.

3). A Programmable SBI: P4 is a domain-specific programming language designed for defining the handling and forwarding of data plane network traffic in P4-enabled forwarding devices, such as network appliances, switches, routers, and network interface cards [67]. The workflow of the P4 programming model includes three main components: P4 architecture, P4 program, and target [68]. The architecture delineates the blocks and interfaces within the data plane. Developers craft the program, typically utilizing the P4 language, tailored for either a target software device or a hardware design. This target could be a software-based switch or a hardware component [69]. This target demonstrates the advantages of programmable data plane devices over traditional networks and SDN. These advantages include allowing user code to control message forwarding, ensuring the same P4 program runs on multiple targets without changing runtime applications, using protocol-independent primitives for packet processing, and employing a robust computing model where match-action stages can function both serially and in parallel. However, despite these benefits, the programmable data plane is limited by its finite memory capacity and inability to perform complex computations like division, exponential, or logarithmic calculations [70].

V REDUCING TCAM ENTRIES IN SDN

SDN uses a unique memory type called TCAM to maximize its programmability benefits. The capacity of TCAM in available SDN switches ranges from 1 to 2 Mbits. Each 1 Mbit chip costs about 350 USD and consumes an average of 15 Watts [71]. Due to TCAM's limitations, SDN switches can only hold a few rules. To manage the flow table in SDN switches more efficiently and reduce TCAM entries, experts have proposed several approaches. One approach is a compression method for the flow table, known as a two-level tagging strategy. This strategy replaces flow entries with two simpler, smaller tags – a path tag (PT) and a flow tag (FT). These tags help reduce the bits needed for TCAM entries to represent flow rules. For example, tagged flows require only 24 bits, significantly less than the 356 bits needed for standard flow entries, thus increasing TCAM's storage capacity [72]. Another proposal is bit weaving, a compression algorithm applied to TCAM. This method lowers the number of rules needed to implement policies on a single switch. Bit weaving involves finding bit swaps that allow related rules to be written as an LPM table, followed by LPM table compression and merging compatible rules into a ternary string [73].

The iSTAMP approach, as proposed in [74] , introduces a method for measuring incoming flows at either fine-grained or coarse levels. This technique dynamically divides flow inputs and utilizes optimization algorithms to enhance the accuracy of network flow predictions. It dynamically splits flow inputs and uses optimization algorithms for accurate network flow predictions. Additionally, to reduce the number of flow rules in network devices and address the rule placement issue, the approach uses wildcard expressions and logic reduction, resulting in minimal compression time [75]. Furthermore, the MINNIE compression technique has two phases: routing and compression. In the routing phase, flows are distributed across the network using a shortest-path method to prevent link overloading. The compression phase employs an effective table compression heuristic to generate three compressed routing tables, selecting the smallest one for use [76].

Researchers in [77] investigated two types of slow DdoS attacks that exploit the limited capacity of switches to store forwarding rules. They recommended combining SIFT with other mitigation techniques and Moving Target Defense-based strategies to counter these attacks. They also proposed the TCAM Razor, which uses multi-dimensional topological transformation and decision trees to minimize TCAM rules [78]. To address NV's scalability issues, which consume significant switch memory, control channel, and CPU cycles, the Flow Virt approach was introduced for flow merging with low overhead [79].

VI FLOW TABLE OVERFLOW PROPOSED SOLUTIONS IN SDN

In this section, we delve into proposed solutions for addressing flow table overflow in SDN. Our analysis encompassed a thorough review of 44 selected articles, as illustrated in Figure 1. Figure 5 provides a taxonomy of flow table attacks, categorizing solutions based on the methods proposed, testing platforms employed, techniques utilized, and specific issues targeted in mitigating these attacks.
Among the 44 articles surveyed, 19 identified optimal strategies for eliminating flows when the flow table reaches saturation. Additionally, seven articles outlined strategies for establishing suitable values for flow entry timeouts, thereby reducing overall flow table space. Moreover, eight articles proposed rerouting flows from switches that consume excessive flow table space to nearby switches with available capacity, effectively optimizing resource utilization. Furthermore, six articles concentrated on the aggregation of flow entries as a means of conserving flow table space. Lastly, four articles introduced various methods aimed at preventing flow table overflow attacks, bolstering the security and resilience of SDN infrastructures against such threats.

A). Mitigating Flow Table Overflow Attacks Using Eviction Strategy

In addressing flow table overflow, eviction emerges as a pivotal strategy, facilitating the removal of existing flow entries to accommodate new rules. This process is particularly vital for popular switch systems like OVS, Pica8, and Cisco Nexus, which commonly rely on the LRU eviction technique. Table 2 offers a comprehensive overview of various eviction mechanisms employed to combat flow table overflow. Noteworthy among these strategies is FTGuard, proposed by the authors of [80]. FTGuard introduces a defense mechanism grounded in prioritization to safeguard switches against saturation and overflow attacks. This innovative approach underscores the proactive measures necessary to mitigate the risks associated with flow table overflow in SDN environments. This mechanism analyzes and categorizes network traffic into high, medium, and low priority. It starts the flow eviction process with lower-priority entries, making room for incoming flows. When the switch's flow table fills up, it uses values stored in the Flow-Mod message's field to remove entries. This strategy employs a statistical approach to assign values to flows. Similarly, authors in [81] introduced the Short Flow First (SFF) replacement algorithm. This algorithm classifies flows into short and long survival periods based on each flow entry's matching period. Deleting short flows first increases matching entries and reduces controller overhead. The SFF al'orithm outperforms FIFO, LFU, and LRU, especially with varying flow table sizes. In another related work, the authors of [82] proposed a two-stage timeout cache management scheme to preserve significant flow entries. The primary table stores entries based on timeout durations set by the controller. When a flow becomes inactive, it moves to the Inactive Flow Queue instead of being removed. This scheme prioritizes evicting short-lived flows to conserve resources while keeping active flows.

The study in [83] suggested the WLRU algorithm for flow table management. This algorithm assigns initial weights to each flow, increasing the weight for existing flows or saving new flow information before forwarding. Enhancing their previous work, CAB, the authors implemented CAB-ACME [84], a reactive caching approach. This approach improves CAB's flow table usage by dynamically adjusting bucket shapes and sizes to fit traffic patterns and preloads large rules for quick response to traffic and policy changes.

To address scalability issues caused by limited flow table capacity, the authors of [85]. Proposed a strategy based on transmission layer disconnection. This approach employs TCP and SCTP control signals to determine flow completion, adding an Active Connection Counter to each entry. The flow entry is automatically ejected when the counter hits zero. The authors in [86] presented the SRL framework with two modules: flow aggregator and hashing. The controller computes hash values for every packet, using the source IP address and maximum segment size. During overflow, the controller replaces the entry with the lower hash value with a new entry. In [87], the authors developed a technique using a D-ITG traffic generator to initiate flow rule eviction before overflow. This approach identifies table capacities and the appropriate eviction threshold. Once the threshold is known, the switch starts the eviction process using Random, LRU, or FIFO techniques.

The study in [88] found that existing works didn't address eviction techniques for UDP flows. They proposed a dynamic monitoring solution using RL and Q-Learning. This model, comprising states, actions, and rewards, uses adaptive sampling of UDP flow statistics to determine when to evict a flow entry. Reference [89] introduced the concept of multiple bloom filters (MBF) to reduce controller-switch communication due to table misses. MBF encodes flows based on locality and recentness, automatically removing less relevant rules during overflow. This increased the overall hit ratio by about 63.2% compared to LRU. The authors of [90] proposed setting a threshold for early eviction of flow entries, inversely related to the number of hosts and packet arrival speed, to reduce packet loss and latency. In [91], a unique flow rule eviction algorithm, Dynamic In/Out Balancing, was proposed. Instead of a fixed threshold, it dynamically modifies flow timeout based on time.

The study in [92] developed a dynamic in/out balancing method with the least frequently used (DIOB/LFU) criteria. This method evicts rules with a zero idle timeout and counter value when the flow table is full, significantly reducing table overflow. FireGuard [93] designed to prevent complex crossfire attacks, consists of three elements: a traffic locator, an attack detector, and a traffic monitor. The strategy uses switch information to identify attacks and their paths. However, its effectiveness in physical environments remains untested. The work in [94] proposed using the hidden Markov model (HMM) for a proactive approach to overcome TCAM memory size limitations. This technique uses a utilization table for eviction and categorizes traffic based on setup flow rules, showing superior performance in various environments. The study in [95], introduced a method to mitigate flow table overflow by replacing forwarding flows from attackers with drop flows, monitored by their timeouts. This method restricts the controller's mitigation technique when traffic increases.

The work in [96] presented a model to counter SDN-based table-overflow attacks using a mathematical technique based on SDN topology. This model includes a token bucket algorithm to ensure consistent transmission for legitimate clients while limiting attacker data rates. The authors in [97] introduced a rate-limiting approach, incorporating a flow-checking module into the controller to regulate traffic and blacklist flows exceeding thresholds. Researchers in [98] introduced a machine learning-based system to select the appropriate flow for removal, using historical data to predict flow entry durations. The study in [99], presented the STAR adaptive routing approach, using limited flow-table resources for efficient network operation. STAR intelligently deletes expired flow inputs and determines routes for new entries based on real-time switch usage. The study in [100] introduced the TF-IdleTimeout technique, dynamically modifying flow entry idle timeout based on real network activity to optimize TCAM capacity usage.

In [101] researchers proposed STEREOS, a machine learning-based intelligent eviction technique, classifies flow inputs into active and inactive categories, significantly reducing control overhead and improving network speed and packet loss rates. Another mechanism named DTER has been proposed in [102], uses a decision tree to select the best flow entries, temporarily storing others using the CBF until their idle timeout expires. Entries, temporarily storing others using the CBF until their idle timeout expires. To detect and prevent low-rate DoS (LdoS) attacks, in [103] authors proposed a mechanism using statistical analysis and LRU replacement for mitigation. This approach includes data collection, overflow prediction, attack detection, and mitigation modules.

F. *Mitigating Flow Table Overflow Attacks Using Flow Entry Timeout or Dynamic Timeout*

This section delves into the mitigation of flow table overflow attacks through the implementation of flow entry timeout or dynamic timeout mechanisms. Table 3 summarizes the various flow entry timeout or dynamic timeout mechanisms against flow table overflow. In [104], the authors combined a dynamic hybrid timeout strategy with a peer support strategy to prevent flow table overload, which can lead to DDoS attacks and assist in acquiring necessary flow data for attack detection. When the flow table usage nears its maximum, the strategy allocates longer durations with larger idle timeout numbers, while flows with shorter durations receive smaller timeout values. The results demonstrate its effectiveness in preventing flow table overflow. The authors in [105] established a hard timeout for long-lived flows based on short inter-arrival periods and set a specific value for short-lived flows. This method removes a flow entry from the table if no packet matches it within a certain time. It has successfully reduced controller overhead and experimental results show a 64.8% decrease in the number of packets in messages. However, it struggles to erase invalid and completed flows from the table, which is crucial for active network operations. Isyaku et al. expanded on IHTA and introduced AH-IHTA (AH-IHTA) [106]. In this approach, flows receive timeouts based on their characteristics. The controller frequently collects data on all active entries from the switch and stores them in a module. When a table miss-entry occurs or a new flow arrives, the module's active flow entries are examined and compared with the flow table usage. If the table shows high usage, the scheme removes the data flow with the fewest packets. Otherwise, new flow entries are installed. In [107], the authors implemented an Adaptive Flow Table Management (AFTM) scheme. AFTM employs dynamic timeout assignment based on flow characteristics and proactive eviction to monitor flow table utilization at set intervals. The cache within this scheme holds flow information and related entries, identifying entries with a long lifespan in the flow table to be removed when the usage ratio exceeds the predetermined threshold. In addition, the authors in [108] presented HQTimer to address the impact on the data plane performance arising from attackers' exploitation of the flow table and apply Q-learning to set values for flow expiry timeout to enhance the flow table performance. The approach is efficient in a small-scale network and does not require switch modifications. The authors in [109] presented a dynamic timeout approach utilizing idle and hard timeouts, which depends on the per-flow packet count. The three components that make up the suggested scheme are the statistics module, the timeout calculation module, and the 2D counting Bloom Filter. The first module updates the bloom filter by extracting specific data (features) from flow entries. The second modules determine the values for the hard and soft timeouts, while the third assigns timeout to every flow. The authors [110] proposed an approach where all extraneous entries that cause bloat are recognized using HyperLogLog, aggregated, and organized into clusters using Hierarchical Agglomerative Clustering in this entry reduction approach. Furthermore, the redundant entries in each cluster are optimized using a Pareto optimizer and a multi-objective optimization technique.

C. Mitigating Flow Table Overflow Attacks by Rerouting Flows

In this section, we explore strategies aimed at mitigating flow table overflow attacks. Table 4 provides a comprehensive summary of studies focusing on rerouting flows from switches with excessive flow table usage to neighboring switches with available capacity. The study in [111] proposes NFV-Guard, a method to mitigate table overflow attacks in SDNs using Network Function Virtualization to filter attackers dynamically. This approach filters traffic through an NFVI, enabling precise management of table overflow attacks. The method operates in three phases: virtual honeypot, NFV-GUARD Controller, and Dynamic Traffic Filtering and Distribution. The virtual honeypot dynamically resizes devices and assigns flow entries. The NFV-GUARD Controller oversees networking tasks. The final phase involves computing the THD, merging IP, and processing TLS packet-in. This method excels when handling a massive influx of new traffic. However, for attacks with complex, covert patterns, existing approaches that prevent overload in a single switch prove ineffective; they only prevent flow table overflow attacks. Authors in [112] introduces a QoS-aware mitigation technique that identifies non-overloaded switches to defend against flow table overflow attacks. This technique involves a traffic monitoring module to observe switch status and a traffic guiding module to check for available flow table space. Standard forwarding rules are inserted if space permits; otherwise, directional rules reroute packets to nearby switches, preventing buildup on the victim switch. A stochastic differential equation-based defense [113] addresses the shortcomings of centralized detection methods in SDN networks. This method consolidates unused space in the network's flow tables, redistributing new entries during attacks. In [114], the authors propose a flow table mitigation technique for managing the flow table and preventing overflow by collecting the state of the switches (data collection) regularly using a sampling approach and applying flow-table space usage strategies. The technique brings about a reduction in table miss rate. However, it is ineffective with dynamic traffic. In [115], the authors introduced the discrete-time finite-state Markov chain (DTMC) model and unsupervised hashing to defend against flow table overload and link spoofing attacks, respectively. More specifically, DMTC determines the status of every switch and forwards the same to the controller. To mitigate the flow table overflow, it uses the switch information to redirect the flows from busy and overflow switches to idle switches. The authors proposed DIFF [116], a dynamic routing technique, to classify traffic based on its impact on resources on the network and adjust the routing pattern. The scheme makes distinctions based on how they affect the resources on the network and modifies routing patterns to lessen flow-table overflow issues and wasteful bandwidth distribution. It creates a set of paths for each pair on the source-destination link edge switches. New flow paths are dynamically selected from pre-generated path sets to balance flow-table usage. The scheme adaptively reroutes elephant flows, utilizing the law of providing max-min equal bandwidth to achieve maximum throughput. The experimental results show that DIFF can simultaneously manage connection utilization and flow tables. It also reduces the controller's workload, and packet latency, thus enhancing throughput compared to other methods (OSPF, Hedera, and FE). Authors in [117] proposed a flow table sharing method that allows a switch in the network unable to process a flow transfer to another switch with a spare flow table. The approach reroutes traffic from overloaded switches to idle or neighbor switches with free flow table resources. It yields a reduction in the number of control messages and RTT time. Nevertheless, the victim switch may flood the nearby switches in the event of a significant attack, leading to a DoS attack.

F. Mitigating Flow Table Overflow Attacks through Aggregated Flow Entries Mechanisms

This section delves into solutions aimed at resolving flow table overflow attacks by aggregating flow entries. Table 5 presents the summary of the various studies through aggregate flow entries to resolving flow table overflow attacks. In [118], the authors proposed IDFA to prevent flow table overflow by creating duplicate entries, which are then combined to form a single entry. The processing logic resides in the switches instead of the controller. Three modules make up IDFA. The first module is responsible for adding and verifying flow entries. The second effectively aggregates flows using a dynamic threshold, and the third handles flow aggregation using degradation and repermutation techniques. It brings about a reduction in flow entry size. However, it uses up the limited TCAM memory. In terms of compression ratio, average flow convergence time, and the likelihood that a flow table overflow will occur, the method outperforms FTRS. The authors of [119] presented a flow rule aggregation method for reducing the number of flow rules in an SDN switch while limiting the impact on individual traffic flow QoS (such as packet loss, delay, etc.). It determines the optimum network path using a heuristic called Best fit. The proposed method performs better than previous benchmark methods (Greedy, Random, Exact-Match, and Agg-Delay) with some metrics. To resolve the flow table overflow occasioned by the management of each flow in SDN and enable effective cluster-based flow management, a similarity-based hierarchical clustering framework [120] is proposed, which uses both similarity-based initial clustering and hierarchical cluster merging. The framework allows flows to be grouped into cluster for routing and processing. The experimental results show that the approach can reduce forwarding rules to 32% and 27% for data center networks and campus networks, respectively, compared to per-flow management. In [121], the authors offer a quick and efficient bit and subset weaving-based flow aggregation technique to reduce the flow table size, offer realistically quick updates and mitigate the problem of flow table overflow such that it takes a short time to update the table. The flow rules are split into different partitions based on their instructions. This results in a reduction in the flow table capacity and suitable update time. It performs better than the FFTA scheme in terms of the average compression ratio. The authors in [122] presented a flow table overbooking isolation guarantees problem (FOLA) approach to route flows through multiple paths. It routes a flow via a path to prevent overflow and enhances the network throughput of the system. To address the diverse behavioral patterns, they were displayed by flows with different properties, and the use of timeouts (idle and hard) results in inefficient management of flows when set higher than the flow durations.

Authors in [123] developed an approach employing a hidden Markov model (HMM) in which entries that are often accessed are placed in the Agg-ExTable to alleviate the issue of a bloated single table and improve flow table management. The method lessens flow processing time. However, it consumes a lot of memory and solely addresses TCAM constraints.

VII DETAILED ANALYSIS OF THE LITERATURE AND RESEARCH GAPS

After conducting a comprehensive analysis of flow table overflow attacks, the authors have categorized the examined articles according to various attack approaches. The classification of reviewed publications is outlined in Tables 6–9, which categorize the studies based on eviction strategies, entry and dynamic timeouts, rerouting of flows, aggregated flows, and other methods. Based on the summarized findings from these tables, several research gaps have been identified. These include but are not limited to:

1). After a thorough analysis of flow table overflow attacks, it is evident that 54.54% of researchers primarily relied on the eviction strategy to mitigate these attacks. Conversely, a smaller percentage of researchers, comprising 15.91%, explored the entry and dynamic timeout approach, along with the rerouting flows technique, while 13.64% opted for an aggregated flows approach (Table 6). However, there's a notable gap in research concerning aggregate flows, rerouting flows, and dynamic timeout approaches for addressing flow table overflow attacks. Specifically, the rerouting flow strategy assumes that some flow tables remain unburdened, redirecting incoming flows to these neighboring tables. Nevertheless, none of the researchers have considered the possibility of overwhelming all flow tables simultaneously, which could be a significant vulnerability considering the number of switches. Given these observations, further investigation into dynamic timeout, rerouting traffic, and aggregated flow techniques is warranted to develop more comprehensive solutions for mitigating flow table overflow attacks effectively. These strategies hold promise but require deeper exploration and analysis to ensure their practical applicability and efficacy in real-world scenarios.

2). In total, 63.64% of the researchers applied their approaches to the controller module because of the unintelligent nature of the switch (Table 7). This method's deployment on the controller necessitates the controller's acquisition of information on each flow entry in the switch, necessitating interaction and communication between the controller and the switches. The controller's memory, processing power, overhead, and bandwidth are all used in their interaction. Except for [73], which requires a switch modification, 36.36% of authors deployed their solutions on the switch without altering it to address the issues caused by approaches on the controller module. Deploying flow table overflow attack solutions in SDN switches is a crucial area for research.

3). Table 8 provides clear insights into the methodologies employed by researchers in evaluating their studies. It indicates that a majority (68.19%) utilized simulation or emulation tools, while 15.91% developed self-made simulators using diverse programming languages. Furthermore, among the studies leveraging SDN controllers (63.64% of the total), only a fraction (9.09%) opted for logically distributed controllers. Notably, the use of a logically centralized controller, as highlighted by [124], introduces a single point of failure. Given these findings, a critical area of research involves implementing solutions tailored to address flow table overflow threats within the framework of a logically distributed controller architecture. This approach seeks to mitigate the risks associated with single points of failure, thereby enhancing the robustness and reliability of SDN infrastructures.

4). In the experimental evaluation, software switches were predominantly utilized in most studies (84.09%), while hardware switches were employed in only 6.81% of cases

(Table 9). It's well understood that the distinct processing capabilities of hardware and software switches can significantly impact switch performance across various parameters [125]. Hence, there arises a necessity to conduct evaluations using both types of switches with the same approaches to ascertain their effectiveness in providing solutions to table overflow attacks in SDN. Furthermore, it's noteworthy that all studies evaluated their work within a linear topology, except for [80, 91, 95, 99, 105, 115, 116]. Therefore, there exists a clear need for further investigation utilizing alternative topologies such as tree and fat networks. Such exploration can provide valuable insights into the performance and scalability of proposed solutions in diverse network configurations, thus enhancing the applicability and generalizability of research findings.

5). For their studies, 47.71% of researchers utilized a common benchmark dataset, while 27.27% employed traffic-generating tools to simulate traffic, including both normal and attack scenarios. However, relying solely on traffic-generating tools may not accurately replicate real-world traffic levels. Moreover, CAIDA stands out as the sole benchmark dataset used. Therefore, there is a crucial need to incorporate real datasets into the development of solutions aimed at detecting, mitigating, and preventing flow table overflow attacks. This area of research warrants significant attention to ensure that proposed solutions are effectively validated against real-world traffic patterns, thereby enhancing their reliability and applicability in practical scenarios.

6). In the realm of SDN, optimizing eviction strategies emerges as a pivotal area of research, particularly in mitigating flow table overflow attacks. Our study reveals that 54.54% of researchers have adopted these strategies, underscoring their crucial role in SDN security. This opens an exciting avenue for further innovation. We propose an exploration of advanced algorithms that enhance the efficiency of eviction processes, thereby striking a balance between network performance and security. Comparative analyses of various eviction strategies under diverse network loads and attack scenarios will provide invaluable insights. This research direction not only promises to fortify SDN against sophisticated threats but also paves the way for groundbreaking advancements in network management. By delving deeper into optimizing these strategies, we can redefine the boundaries of network security and efficiency, making a substantial contribution to the field of SDN.

7). In the quest to fortify SDN against flow table overflow attacks, enhancing the utilization of simulation tools stands as a crucial endeavor. Our findings highlight that a substantial 68.19% of researchers rely on simulation or emulation tools, signaling an urgent need for more refined and realistic models. We propose a bold initiative to develop state-of-the-art simulation tools that accurately mirror the complexities of real-world network environments and cyberattack patterns. Collaborating with industry experts to access real traffic data and configurations will inject a dose of practicality into these simulations. Additionally, integrating artificial intelligence into these tools could offer predictive insights and a deeper understanding of network behavior under varied conditions. This approach not only elevates the accuracy of our research outcomes but also serves as a beacon for future studies, guiding the way towards more resilient and intelligent SDN solutions.

8). In the dynamic landscape of SDN, the deployment efficiency of controller modules stands as a frontier for groundbreaking research. Our analysis indicates that a striking 63.64% of researchers target the controller module due to the switch's limited intelligence, pointing to a significant opportunity for enhancement. The need for the development of innovative algorithms and frameworks that streamline flow table management while minimizing resource consumption is key. These advancements could revolutionize the controller's functionality, potentially integrating predictive analytics or machine learning to achieve unprecedented efficiency. By shifting the focus to more resource-efficient controllers and potentially redistributing intelligence to the switches, we can dramatically enhance network resilience and performance. This proactive approach in redefining controller module deployment will not only address current challenges in SDN but also set a new standard for future network architectures, fostering a paradigm shift in how we conceptualize and implement network intelligence.

VIII CHALLENGES AND FUTURE DIRECTION

A. Deployment of Flow Table Overflow Detection, Mitigation, and Prevention Solutions

The deployment of solutions for detecting, mitigating, and preventing flow table overflow attacks presents a critical challenge in SDN. The majority of detection, mitigation, and prevention measures against flow table overflow attacks have been implemented within the controller [80, 84, 86–88, 92–103, 106–116, 122]. Consequently, communication between forwarding elements and the controller becomes essential for acquiring switch information and redirecting all traffic (normal and attack) to the controller for detection and mitigation. Moreover, the controller must continually gather flow statistics from forwarding devices to monitor network traffic, resulting in overhead and latency. Some authors [82, 84, 115] have tackled this challenge by deploying the detection and mitigation modules into both the controller and the switch. Therefore, there is a pressing need to distribute the deployment of solutions for flow table overflow attacks. This entails exploring methods to decentralize the implementation of these solutions, reducing the burden on controllers and enhancing overall network efficiency and resilience. Consequently, the distribution of deployment for solutions to flow table overflow attacks represents a significant area of concern and interest for future research endeavors.

B. Providing Solutions to Flow Table Overflow Attacks in Various Scenarios

Addressing flow table overflow attacks requires solutions tailored to both typical network settings and scenarios where OpenFlow switches are under threat. Researchers have offered solutions to these attacks in both typical network settings [98, 99, 104, 108, 117, 123] and when the OpenFlow switch is under threat. It's imperative to apply the same techniques to detect and mitigate attacks under these two scenarios (normal network setting and OpenFlow Switch) to determine the most effective technique for each scenario.

C. The Security in SDN

Security in SDN is paramount due to the separation of the control plane from the data plane, which introduces vulnerabilities such as flow table overflow attacks leading to DoS incidents. Attackers exploit the limited size of TCAM to flood the flow table, compromising the confidentiality, integrity, and availability (CIA) of the network. Thus, implementing robust measures is essential to safeguard SDN networks and their resources against security threats. The separation of the control plane from the data plane ushers in the security threats, such as flow table overflow attacks resulting in DoS attacks in SDN when not adequately prevented. The attackers take advantage of the limited size of TCAM to overflow the flow table. It violates the three triads (CIA) of security such that the switches could not hold additional flow entries, and the controller would not be unavailable due to an influx of illegitimate requests from the attackers. Therefore, it is required to put appropriate measures in place to guarantee the confidentiality, integrity, and availability of the networks and their resources to prevent security threats in SDN.

D. Deployment of Solutions in a Multi-Controller Architecture

In the realm of SDN, distributed controllers outperform centralized ones in scalability, consistency, load balancing, and response time necessitating the adoption of a multi-controller architecture to mitigate single points of failure and ensure network availability.

Scalability, consistency, load balancing, and response time are all areas where distributed controllers outperform centralized controllers [126],[127]. In the SDN environment, it is necessary to consider multi-controller architecture to address the single point of failure in a centralized architecture of SDN, as this will also cater to handling large traffic volume and ensures network availability.

E. Empirical validation in a range of network configurations

Empirical validation across diverse network configurations is crucial to assess the practical effectiveness, adaptability, and scalability of methods proposed for addressing flow table overflow attacks, particularly in resource-constrained environments.

Methods proposed to address flow table overflow attacks consume TCAM memory heavily, creating scalability issues for larger networks and further limiting scalability and applicability, especially in resource-constrained environments. It is worth noting that many proposed techniques struggle to adapt to dynamic traffic scenarios, causing inefficiencies in flow management. Empirical validation across diverse network setups is imperative to ascertain the practical effectiveness, adaptability, and scalability of these methods. Moreover, addressing these challenges is pivotal for advancing the scalability, adaptability, and efficiency of systems in real-world deployments, as improvements in real-time adaptation and efficiency are crucial, particularly in managing flow entries exhibiting diverse behavioral patterns.

F. The need for more optimization strategies

Certain approaches like NFV-Guard and specific rerouting strategies are promising in preventing flow table overflow attacks, but their efficacy against complex and covert attack patterns remains uncertain. Furthermore, a notable research gap lies in the lack of validation of proposed strategies within real-world network environments, raising concerns about their genuineness in practical scenarios. Additionally, some optimization techniques aim to identify and improve redundant entries within flow tables, there remains a need for more efficient optimization approaches to address the bloat and inefficiencies within SDN flow tables comprehensively. These gaps represent significant avenues for further academic research in SDN, pivotal for enhancing security, scalability, and practical applicability in real-world network environments.

G. Resource-efficient strategies and unified evaluation techniques

There is a need for a comprehensive analysis of approaches' robustness against sophisticated attacks, exploration of dynamically adaptive eviction strategies, establishment of unified evaluation metrics, and further research on resource-efficient strategies for scalability in large-scale SDN networks. These areas represent key directions for enhancing the effectiveness, security, and scalability of systems in practical deployments.

H. Empirical evaluation in real-world settings

Ensuring the scalability and adaptability of techniques in dynamic networks is challenging; proposed solutions may struggle against complex attacks, requiring enhancements to tackle covert patterns effectively. Empirical validation in real-world scenarios is crucial to confirm practical effectiveness and real-time adaptation to evolving threats is vital for bolstering security measures. Additionally, while some approaches mitigate primary attacks, they might inadvertently expose vulnerabilities, risking secondary attacks or network disruptions.

I. Diverse Network Topologies

In the evolving domain of SDN, exploring diverse network topologies represents a pivotal step toward comprehensive research. Our analysis reveals a predominant focus on linear topologies, a scenario that scarcely reflects the multifaceted nature of real-world networks. There is a need for a bold expansion into studying SDN's behavior across a spectrum of complex topologies, including tree, star, and fat-tree configurations. This exploration is not just an academic exercise; it is a vital undertaking to understand how SDN solutions perform under varied structural complexities, especially in the face of flow table overflow attacks. By broadening our investigative scope to encompass these diverse topologies, as this will uncover critical insights into the resilience and adaptability of SDN architectures. This foray into uncharted territory promises to elevate our understanding of SDN, ensuring that our solutions are robust, versatile, and aligned with the intricate realities of modern network infrastructures.

J. Employing Real World dataset.

In the quest to enhance the robustness of SDN against flow table overflow attacks, the employment of real-world datasets emerges as a crucial and transformative research strategy. Our study underscores the limitations of relying solely on common benchmark datasets and traffic-generating tools, which currently dominate the research landscape. There is a need to champion the pioneering move to utilize real-world traffic datasets, bringing an unprecedented level of authenticity and relevance to our research. This approach not only promises a more accurate representation of network behaviors under attack scenarios but also offers invaluable insights into the effectiveness of proposed solutions in genuine settings. By embracing real-world data, we propel our research beyond theoretical models, grounding it in the tangible complexities of existing network environments. This shift marks a significant stride towards developing SDN solutions that are not just theoretically sound, but practically invincible in the face of evolving cyber threats.

IX CONCLUSIONS

The development of SDN has been primarily driven by its advantages over traditional networks, simplifying and streamlining flow table management through its centralized architecture. However, despite these advantages, ensuring the security of SDN remains a significant challenge. This paper presents a systematic review of flow table overflow attacks in SDN, examining 44 high-quality research articles out of a pool of 794. These articles are categorized based on suggested solutions, with 54.54% employing eviction methods, 15.91% utilizing entry and dynamic timeouts along with flow rerouting, and 13.61% adopting aggregated flows. Our findings highlight the need to explore alternative approaches such as entry and dynamic timeouts, flow rerouting, and flow aggregation. Additionally, articles are categorized based on where solutions are deployed: 63.64% at the control plane, 36.36% at the data plane, and 6.82% addressing overhead and latency issues by deploying solutions in both the switch and the controller. Consequently, the distribution of solution deployment for flow table overflow attacks emerges as a crucial area of interest and concern.

Furthermore, 68.19% reviewed papers validated their approaches with simulation or emulation tools, while 15.91% used self-developed simulators using various programming languages. In addition, only 9.09% of the 63.64% of researchers that deployed SDN controllers did so in a logically distributed manner. Implementing solutions to flow table overflow attacks in a logically distributed controller architecture is a crucial area of research interest because a centralized controller suffers from a single point of failure.

Many existing approaches struggle to adapt to traffic dynamics, resulting in inefficient flow management. Therefore, empirical validation across various network configurations is crucial to address these inefficiencies. Additionally, while some optimized approaches effectively combat flow table overflow attacks, their validation within real-world network environments remains uncertain, leading to inefficiencies in SDN flow tables. Thus, enhancing security, scalability, and practical applicability in real-world network environments necessitates efficient optimization approaches. Moreover, existing approaches often overlook unified evaluation metrics and fail to adopt dynamically adaptive eviction approaches, which are vital for scalability in large SDN networks.

In the intricate world of SDN, the comparative study of hardware and software switches emerges as a vital research avenue. Our study reveals a stark contrast in their usage, with a predominance of software switches (6.81%) in experimental evaluations. This disparity highlights an untapped potential for comprehensive comparative studies. There is a need for in-depth research comparing the performance, scalability, and security of hardware versus software switches under various attack scenarios, particularly flow table overflow attacks. Such research promises to unravel the unique strengths and limitations of each switch type, offering a nuanced understanding of their roles in SDN environments. Pioneering this comparative approach, will pave the way for more adaptive, secure, and efficient network infrastructures, tailored to meet the diverse needs of modern digital ecosystems. This endeavour not only bridges a significant knowledge gap but also propels us towards a future where network solutions are as versatile as the challenges they face.

In the evolving domain of SDN, exploring diverse network topologies represents a pivotal step toward comprehensive research. Our analysis reveals a predominant focus on linear topologies, a scenario that scarcely reflects the multifaceted nature of real-world networks. There is a need for a bold expansion into studying SDN's behaviour across an academic exercise; it is a vital undertaking to understand how SDN solutions perform under varied structural complexities, especially in the face of flow table overflow attacks. By broadening our investigative scope to encompass these diverse topologies, we stand to uncover critical insights into the resilience and adaptability of SDN architectures. This foray into uncharted territory promises to elevate our understanding of SDN, ensuring that our solutions are robust, versatile, and aligned with the intricate realities of modern network infrastructures.

ACKNOWLEDGEMENTS

We appreciate Universiti Putra Malaysia (UPM) for providing an enabling environment for conducting the research work. In addition, O. I. Aladesote would like to thank the Management of Federal Polytechnic, Ile Oluji, Ondo State, Nigeria for their support in pursuing his postgraduate studies.

REFERENCES

[1] L. OCHOA-ADAY, C. CERVELLÓ-PASTOR, and A. FERNÁNDEZ-FERNÁNDEZ, "Discovering the Network Topology: An Efficient Approach for SDN," *Adcaij Adv. Distrib. Comput. Artif. Intell. J.*, vol. 5, no. 2, p. 101, 2016, doi: 10.14201/adcaij201652101108.

[2] L. Ochoa-Aday, C. Cervello-Pastor, and A. Fernandez-Fernandez, "ETDP: Enhanced topology discovery protocol for software-defined networks," *IEEE Access*, vol. 7, pp. 23471–23487, 2019, doi: 10.1109/ACCESS.2019.2899653.

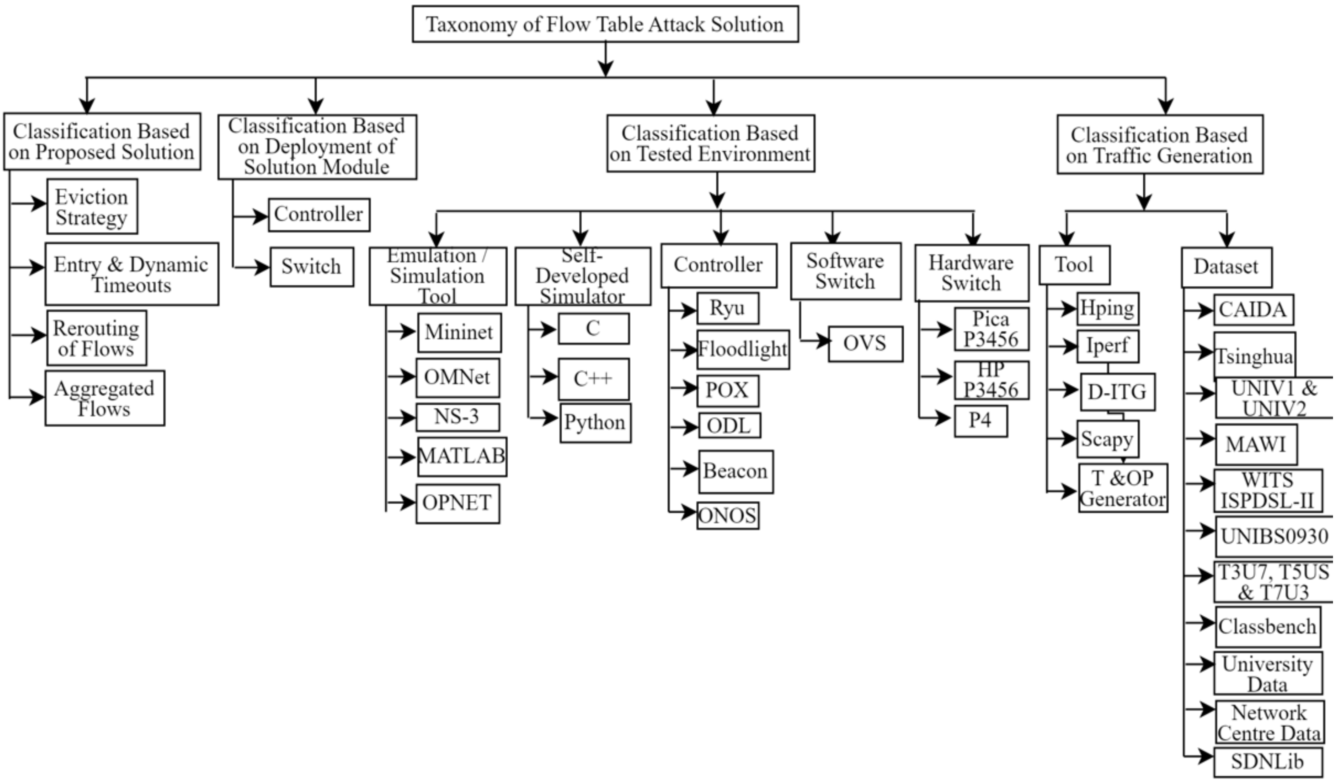

Fig 5. Flow Table Overflow Taxonomy Attacks

TABLE II
MITIGATING FLOW TABLE OVERFLOW ATTACKS USING EVICTION STRATEGY

Ref.	Proposed Solution	Technique Used	Issue Addressed	Metrics Used	Motivation	Merits	Demerits
[80]	A behavior-based priority-aware tagged FTGuard	Priority-based	Network Performance	The resource usage under attack scenarios	The LRU mechanism for flow entry eviction is not efficient and effective	Effective in the prevention of overflow attacks	Consider neither the controller workload nor the additional traffic features
[81]	SFF	Matching period and MWT	Processing delay and controller overhead	Flow Miss Rate and Ratio of delayed packets,	The existing replacement strategies (LFU, LRU, and FIFO) failed to take traffic patterns into account when replacing flow entries.	Increasing the number of matching flow entries reduces the overhead of the controller	High packet processing time and increased switch memory efficiency.
[82]	A two-stage timeout (TST) approach.	FIFO, Random, timeout	The scalability issue of SDN switches' flow tables	Cache hit ratio, discarded packet ratio, rule installation times, Energy saving on TCAM query	The fixed timeout management causes inefficient utilization of the flow table	The retention of only those flow rules that are necessary	No suitable timeout
[83]	WLRU	Linked List	The flow table overflow degrades network scalability	Number of entries, RTT delay, Replied packets	An attempt to detect and mitigate SDN attacks and their internal factors.	The approach improves the network scalability.	The approach is not tested on a larger testbed.
[84]	CAB-ACME	Bucket tree	Efficiency in flow table usage	Cache miss, bandwidth usage, Computational time, Latency, flow setup, cache entries	Proximity of traffic and issue of rule dependency	Reduction in control load and enhancement in efficiency of flow table	It requires adjustment and modification before being implemented across the network.
[85]	Transmission layer disconnection-based strategy	Active Connection Counter	Limited Flow table capacity	Flow entry requirements, flow table construction, control overheads, flow table miss rates, and traffic intensities.	Flow expiry mechanisms, employed to address limited flow table capacity do not guarantee optimal performance	There is an improvement in scalability with little or no costs.	The expedited invalid TCP flow eviction method, which does not work well in elephant traffic situations, needs to be improved.

[86]	SRL	Hash value	Flow table overflow	Service delivery time, CPU utilization, and Time	Issues arise with a delayed TCP handshake and the implementation of temporary forwarding rules while handling a high number of users	Able to prevent flow table overflow	Failed to resolve overhead issues
[87]	Early Eviction	Random, FIFO, and LRU techniques	Flow table overflow	Throughput, delay, and number of replacements.	The timeout approach to overflow falls short of optimum performance because it works effectively if there is enough memory to store large flow rules.	LRU outperforms others in preventing overflow	There is a need to consider more traffic and a dynamic timeout approach.
[88]	A dynamic monitoring approach based on RL	Q-learning	Flow table overflow	Control overhead, overflow occurrences, and flow reinstallations.	Existing works do not capture eviction strategy for UDP flows	A reduction in the rate of flow table overflow and flow entry reinstallation	Needs to be evaluated on more performance metrics to ascertain its effectiveness.
[89]	Smart data logging approach	Multiple Bloom Filters (MBF) and hash function	Overhead and latency issues of flow table overflow	Active flow and table hit ratio.	Existing works do not use data structures focused on space efficiency to avoid flow table overflow	It reduces the rate of flow misses, irrespective of the type of flow	Not compared with existing techniques to ascertain its effectiveness.
[90]	Eviction approach	Setting a threshold	The impact of flow table overflow on latency and packets	Delay	The existing approaches to preventing flow table overflow led to packet loss and an increase in latency	It leads to a reduction in packet loss and latency	The study fails to incorporate more metrics to determine an appropriate eviction threshold in a real time environment
[91]	Dynamic Rule eviction approach	DLFU	Performance of SDN when overflow occurs	N/A	The need to lower the rate of flow table miss and provide protection from dangers brought on by overflow motivates the work.	It reduces the rate of overflow occurrence	Unable to reduce the risks brought on by the overflow attack.
[92]	An eviction algorithm	DIOB/LFU	Flow table security and performance	Bandwidth and packet	The necessity or need to improve flow table performance and security.	It significantly reduces the frequency of overflow	It results in packet loss
[93]	FireGuard	LRU and Token	CFTO	Communication delay, detection delay, CPU Utilization, accuracy.	The existing remedies for repeatedly overloading a single switch are ineffective when dealing with an attack involving complex and covert patterns.	With negligible overheads, the novel strategy is particularly effective in preventing crossfire attacks.	The strategy needs to be implemented on a physical environment
[94]	Proactive technique based on matching probability of the entry's prediction	HMM	Select an entry carefully and efficiently for eviction during table misses or when a timeout occurs	CPU consumption rate, number of misses, matching probability	The existing reactive approaches do not improve the forecast accuracy and increase the rate of table misses.	Eviction of flow entry with the smallest matching probability	Further investigation needs to be carried out to ascertain the variables affecting the performance of the metrics.
[95]	A packet monitoring approach	Filtering approach	Flow table overflow attack	Flow change rate, flow rule count, and packet_in count	The necessity to identify and defend against SDN susceptibilities.	It effectively reduces the attack	Restrict the controller mitigation technique whenever the traffic increases.
[96]	A mitigation approach	Flow entry token bucket and statistics	Flow table overflow attack	The number of transmitted attack flows, flow table consumption on the victim switch	The need to address flow table overflow attacks with a sophisticated attack pattern since the existing simplified approach does not give the desired results.	It lowers attack rates	The flow entries with static timeouts require manual deletion.
[97]	Rate limiting approach	Threshold	Flow table overflow attack	Flow table capacity, packet loss, and impact of bandwidth	The existing works do not ensure a secured SDN environment	It is effective in preventing flow table overflow	High rate of packet loss
[98]	Eviction Approach based on machine learning	Random forest and k-fold cross-validation	effective utilization of the flow table.	The number of capacity misses normalized, and the number of active flow entries	There is a need for a new and better deletion approach to evict flow entry than the current one that produces very low overhead.	It outperforms the LRU scheme regarding flow table utilization and fewer capacity misses.	Do not consider the overhead of the study

[99]	STAR	LRU and Idle timeout	Low table bloat and Overflow	Controller workload, packet delay, throughput, mouse flow, and elephant flow completion percentages	Using the LRU process to evict flow entries causes flow table overflow, as well as the flow table utilization being misinterpreted due to timeout design, resulting in flow table bloat.	Prevent flow table bloat and overflow with low controller workload, low packet delay, and high server throughput.	Need to be tested in a test bed environment.
[100]	TF-IdleTimeout	Confidence interval	Scalability issue in flow table overflow	Flow entry missing and flow dropping	The existing plan failed to consider the dynamic nature of traffic in an SDN-based network.	It enhances the efficiency of TCAM usage	The study does not consider all variables required to determine a suitable idle time out.
[101]	STEREOS	Machine learning	Efficient management of flow table	Normalized number of capacity misses, accuracy, and number of active flows	Existing approaches to evicting flow rules degrade network performance by evicting flow entries incorrectly	It minimizes control overhead, boosts network speed, and lowers packet loss rates	Its performance should be compared with other machine learning methods.
[102]	DTER	Decision tree	Controller overhead arising from overflow	Accuracy rate, true positive, false positive rates, workload, elephant, and mice flow completion percentage without packet drop, packet delay, and throughput.	Overhead that arises due to huge route request to the controller in a large data center and the limited flow table space	Reduction in controller workload and prevention flow table overflow	More metrics (throughput and latency) should be considered to measure its performance.
[103]	SAIA	LRU and statistical analysis	Network performance	CPU usage, throughput, and detection rate	There has not been comprehensive research on table overflow LDoS attacks, and the available research centered on mitigation solutions for table overflow in a typical network.	Effective in detecting and preventing LDoS attacks	Require an intelligent algorithm to enhance the detection accuracy of the attack.

TABLE III
MITIGATING FLOW TABLE OVERFLOW ATTACKS USING FLOW ENTRY TIMEOUT OR DYNAMIC TIMEOUT

Ref.	Proposed Solution	Technique Used	Issue Addressed	Metrics Used	Motivation	Merits	Demerits
[104]	The flow table overload, which results in DDoS attacks	A dynamic hybrid timeout approach	TCAM Memory durability and network performance	Flow table memory status with different switches and time	Existing related works relied on idle timeout to solve the problem, which is inefficient when dealing with network flows with a limited number of packets and short duration.	Improvement in memory utilization of flow table	Utilizing long-lived flows with a short packet inter-arrival time results in reduced efficiency.
[105]	Idle–hard timeout allocation (IHTA)	LRU	Scalability issue	Number of packet_in messages, flow duration	Timeouts that are inappropriately set result in the early eviction of active flows	It reduces the packet_in messages, thereby enhancing the efficiency and scalability of the flow table.	It is impracticable to get the precise packet count before the flow's conclusion via experiment.
[106]	Adaptive and hybrid idle–hard timeout allocation (AH-IHTA) strategy	FIFO, Random, LRU, and DFE	Prevention of flow rules from expiring frequently	Flows packet count, packet_in event	The fixed timeout approach to managing flow entries is inefficient and ineffective in dealing with the ever-changing characteristics of traffic flow.	It outperforms IHTA by reducing the number of packet_in messages to 72% as against IHTA, which has a 35.2% reduction.	Not suitable for applications in a multi-controller environment
[107]	Adaptive flow table Mgt. scheme AFTM)	Timeouts setting	Inefficient use of limited flow table space.	Packet drop, and Extra table miss	Existing works did not consider the use of timeout settings.	Reduction in table miss	The approach is not tested on a larger testbed.
[108]	HQTimer	Hybrid timeout mechanism and a Q-learning approaches	Rules dependency issues	Table-hit rates, overflow numbers, total installations	The rule dependency issue introduced by wildcard rules makes it difficult to maintain the network's semantics and	Efficient in a small-scale network and does not require switch modifications.	Not effective in a large-scale network environment

					create a timeout mechanism.		
[109]	History-based dynamic timeout	Bloom filter	Flow table overflow attack	The number of DoS flows over and the number of normal flows.	The hard and soft timeouts are ineffective in reducing the number of flow rules that cause overflow.	It leads to a reduction in the number of flow rules	Tested using a small testbed.
[110]	CEOF	Hierarchical Agglomerative Clustering and HyperLogLog	Scalability issue in flow table overflow	Compression ratio, space saving, flow processing time, packet_in request, flow entries, count, throughput, and retransmission error rate	The existing approaches are limited to generating outstanding results in uncertain conditions. Hence, there is a need for an approach that will give excellent results.	The experimental results show that the approach can effectively prevent flow table overflow	The scheme needs to have its QoS and security improved

TABLE IV
MITIGATING FLOW TABLE OVERFLOW ATTACKS BY REROUTING FLOWS

Ref.	Proposed Solution	Technique Used	Issue Addressed	Metrics Used	Motivation	Merits	Demerits
[111]	NFV-GUARD	NFV	Flow table security and performance	Flow table occupancy, RTT, and CPU usage	Sending flow entries to network nodes on the physical infrastructure appears to be effective only in large networks.	It reduces the controller loads and the delay	Could not effectively mitigate flow table overflow in a large network.
[112]	A Peer Support Strategy	Poisson distribution.	The limited size of the flow table, which attackers use to cause overloading	Holding time	No research has dealt with a specific attack by thoroughly investigating and proffering solutions to the attack.	It minimizes violation of quality of service	It yields an additional delay and relies on the resources of the idle flow table
[113]	A stochastic differential equation-based defense for overflow attack	BPNN algorithm and flow table sharing	Flow table overflow attack occasioned by the limited flow table	Recognition rate, time-consuming	Due to the poor defense capability, present centralized detection solutions for overflow attacks result in insufficient SDN network protection and irreversible losses.	The proposed method performs better in detection rate and takes less time to run.	Do not consider the situation where the neighboring switches are totally or nearly full.
[114]	CPD	Taylor series	Flow table overflow attack	The number of flows, CPU utilization, frequency of table overflow, and the number of dropped packets.	Existing approaches are computationally costly, result in more table misses, and have scalability issues	It lowers the network's table miss rate.	It is unsuitable when traffic is dynamic.
[115]	Hybrid method	Discrete-time finite-state Markov chain, fuzzy classifier, and L1-extreme learning machine	Flow table overflow	Delay, failure ratio, and holding time	The use of peer-to-peer topology in addressing the flow table overflow caused much switch damage, which resulted in the replacement of many damaged switches.	It resolves the security problems (flow table overflow and link spoofing attacks) in SDN	It relies on the resources of the flow table
[116]	DIFF, A dynamic routing scheme	LRU	Flow table overflow and inefficient bandwidth allocation	Controller workload, packet delay, and throughput	Unbalanced flow-table usage at different switches. Traditional, inefficient bandwidth allocation, and flow table bloat	Enhances network throughput and regulates flow table utilization	Does not consider experimental research to determine the appropriate threshold for idle timeout
[117]	File Table Sharing	FTS	Overhead in flow table overflow	Number of entries and distance	The limited size of a flow table results in a significant increase in the number of packet_in messages between the switch and the controller when a table miss occurs.	It reduces the number of control messages and is simple to implement.	The victim switch may flood the nearby switches in the event of a significant attack, leading to a DoS attack.

TABLE V
MITIGATING FLOW TABLE OVERFLOW ATTACKS THROUGH AGGREGATED FLOW ENTRIES MECHANISM

Ref.	Proposed Solution	Technique Used	Issue Addressed	Metrics Used	Motivation	Merits	Demerits
[118]	In-switch dynamic flow aggregation (IDFA)	Degradation and repermutation algorithms	Link delay arising from flow table overflow issue	Compression ratio, convergence time, overflow time, and number of redundant flow entries.	The failure of FTRS, a previously used method, to generate an efficient outcome in terms of compression ratio, convergence time, and overflow rate	This method reduces the number of entries in the flow table	When used in a large-scale network environment, this could produce an additional overhead.
[119]	Flow rule aggregation scheme	Best-fit heuristic	Flow table overflow attack	Average delay, packets dropped, average throughput, and flow rules	The research is motivated by the idea of using specific QoS measures (delay and packet loss) to determine a routing path.	It improves throughput while lowering average delay and packet loss.	Need to consider the Internet traffic in the study
[120]	A cluster-based management of flow entries	k-means clustering	Flow table overflow and unnecessary overhead	Throughput, number of clusters.	The per-flow SDN management is inappropriate for high-traffic networks, as it causes flow table overflow and additional processing overhead.	It minimizes the flow table consumption and improves routing performance.	Need to be tested in a large test bed environment.
[121]	A bit and subset weaving-based flow aggregation	Merging of flows	Flow table overflow	Average compression ratio, average number of times to trigger flow aggregation, and average aggregation time	The per-flow management of the flow table cannot solve or reduce flow table overflow.	It resolves the flow table problem reasonably.	The overhead increases with the number of messages transmitted.
[122]	Guaranteed minimum progress and bounded maximum Algorithm	N/A	Flow table overbooking isolation guarantees problem (FOLA)	Minimum progress, maximum flow table overflow, network throughput.	Degradation in network performance and packet losses arising from massive flow rules replacement when overflow occurs.	Increase network throughput	Need to be tested in a large network environment.
[123]	Agg-ExTable scheme	HMM, Pruning, and Quine–McCluskey algorithm	Memory overhead	Flow processing time, match rate, accuracy	Existing flow table systems that ensure isolation across flows are ineffective because they do not account for flow table size.	It performs better than existing approaches in terms of network throughput	It consumes a lot of memory and solely addresses TCAM constraints.

TABLE VI
SELECTED ARTICLES' CLASSIFICATION BASED ON APPROACHES USED

Approaches	Research Articles	% of Articles
Eviction strategy	[80-103]	54.54
Entry and Dynamic Timeouts	[104-110]	15.91
Rerouting of Flows	[111–117]	15.91
Aggregated Flows	[118–123]	13.64

TABLE VII
SELECTED ARTICLES' CLASSIFICATION BASED ON THE DEPLOYMENT OF THE MITIGATION AND PREVENTION MODULE

Mitigation/Prevention Module Deployment	Research Articles	% of Articles
Controller	[80, 84, 86–88, 92–100, 102, 103, 106–116, 122]	63.64
Switch	[81–83, 85, 89–91, 101, 104, 105, 117–121, 123]	36.36

TABLE VIII
SELECTED ARTICLES' CLASSIFICATION BASED ON TESTBED ENVIRONMENT

Testbed	Hardware/ Software	Research Articles	% of Articles
Emulation/ Simulation Software	Mininet	[80, 81, 83, 87, 88, 90, 92, 93, 95, 97, 100, 102-106, 110-112, 117-121, 123]	54.55
	OMNET++	[115]	2.27
	NS-3	[101]	2.27
	MATLAB	[94, 123]	4.55
	OPNET	[99, 116]	4.55
Self-Developed Software	C++	[84]	2.27
	C	[85]	2.27
	Python	[82, 101, 113, 116]	9.09
Controller	Ryu	[83, 87, 88, 93, 100, 102-106, 109-111, 122]	31.82
	Floodlight	[80, 95, 118, 121, 123]	11.36
	POX	[95, 117, 124]	6.82
	ODL	[83, 94, 96]	6.82
	BEACON	[92, 97]	4.55
	ONOS	[81]	2.27
Software Switch	OVS	[80–83, 87–102, 103–106, 109–112, 115–119, 121–124]	84.08
Hardware Switch	Pica83297 Switch	[84]	2.27
	HP2920 Switch	[84]	2.27
	P4 Switch	[85]	2.27

TABLE IX
SELECTED ARTICLES' CLASSIFICATION BASED ON TRAFFIC GENERATION

Traffic Generation	Tool / Dataset	Research Articles	% of Articles
Traffic Generating Tool	Hping	[96]	2.27
	Iperf	[88, 106, 111, 120]	9.09
	D-ITG	[87, 119]	4.55
	Scapy	[92, 97, 100, 109]	9.09
	TCPReplay & Ostinato Packet Generator	[81]	2.27
Dataset	CAIDA	[82, 96]	4.55
	Tsinghua Campus Network Lab.	[80]	2.27
	UNIV1 & UNIV2	[86, 89, 98, 101, 105-107]	15.91
	UNIV1, UNIV2 & MAW1	[108]	2.27
	WITS ISPDSL-II	[86]	2.27
	UNIBS0930 & UNIBS1001	[101]	2.27
	T3U7, T5US & T7U3	[88]	2.27
	UNIV1	[120]	2.27
	Classbench & Syn MAWI	[108]	2.27
	CAIDA & MAWI	[102, 110]	4.55
	University data	[114]	2.27
	Network center data	[114]	2.27
	SDNLib	[122]	2.27

[3] A. Mateen, Q. Zhu, S. Afsar, and S. A. Sahil, "Effect of encryption delay on TCP and UDP transp
[4] M. Jarschel, T. Zinner, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for SDN," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 210–217, 2014, doi: 10.1109/MCOM.2014.6829966.
[5] F. Ieee *et al.*, "Software-Defined Networking : A Comprehensive Survey," vol. 103, no. 1, 2015.
[6] H. Elzain and W. Yang, "Decentralizing software-defined wireless mesh networking (D-SDWMN) control plane," *Lect. Notes Eng. Comput. Sci.*, vol. 2235, 2018
[7] X. Liya, D. Anyuan, G. Mingzhu, S. Jiaoli, and G. Guangyong, "A MADM-based handover management in software-defined 5G network," *Eng. Lett.*, vol. 27, no. 4, pp. 842–849, 2019.
[8] I. S. Hwang, A. Rianto, E. Ganesan, A. F. Pakpahan, and A. T. Liem, "A Generic Peer-to-peer File Sharing Architecture for Software-defined TWDMA-PON," *Lect. Notes Eng. Comput. Sci.*, vol. 2239, pp. 120–123, 2019.
[9] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks: A survey," *ACM Comput. Surv.*, vol. 47, no. 2, 2014, doi: 10.1145/2655690.
[10] H. N. Noura, R. Melki, A. Chehab, and M. M. Mansour, "A Physical Encryption Scheme for Low-Power Wireless M2M Devices: a Dynamic Key Approach," *Mob. Networks Appl.*, vol. 24, no. 2, pp. 447–463, 2019, doi: 10.1007/s11036-018-1151-7.
[11] Q. Guo, X. Guo, Y. Bai, R. Patel, E. Ipek, and E. G. Friedman, "RESISTIVE TERNARY CONTENT ADDRESSABLE MEMORY SYSTEMS FOR DATA-INTENSIVE COMPUTING," 2015. Availabe: https://ieeexplore.ieee.org/stampPDF/getPDF.jsp?tp=&arnumber=7274248&ref
[12] K. Mathan and T. Ravichandran, "Data Intelligent Low Power High Performance TCAM for IP-Address Lookup Table," *Circuits Syst.*, vol. 07, no. 11, pp. 3734–3745, 2016, doi: 10.4236/cs.2016.711313.
[13] M. Rzepka, P. Borylo, A. Lason, and A. Szymanski, "PARD: Hybrid Proactive and Reactive Method Eliminating Flow Setup Latency in SDN," *J. Netw. Syst. Manag.*, vol. 28, no. 4, pp. 1547–1574, 2020, doi: 10.1007/s10922-020-09550-z.
[14] X. Y. Miao, Q. N. Yang, C. J. Feng, R. M. Bao, K. Zhao, and L. Z. Xiao, "DevoFlow: Scaling Flow Management for High-Performance Networks," *Sel. Pap. Photoelectron. Technol. Comm. Conf. held June–July 2015*, vol. 9795, p. 979532, 2015, doi: 10.1117/12.2209580.
[15] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "Infinite CacheFlow in software-defined networks," *HotSDN 2014 - Proc. ACM SIGCOMM 2014 Work. Hot Top. Softw. Defin. Netw.*, pp. 175–180, 2014, doi: 10.1145/2620728.2620734.
[16] R. Bauer and M. Zitterbart, "An Optimization-based Approach for Flow Table Capacity Bottleneck Mitigation in Software-Defined Networks," 2021, [Online]. Available: http://arxiv.org/abs/2109.08482
[17] B. Isyaku, M. S. Mohd Zahid, M. Bte Kamat, K. Abu Bakar, and F. A. Ghaleb, "Software Defined Networking Flow Table Management of OpenFlow Switches Performance and Security Challenges: A Survey," *Futur. Internet*, vol. 12, no. 9, p. 147, 2020, doi: 10.3390/fi12090147.
[18] E. Do Kim, Y. Choi, S. I. Lee, and H. J. Kim, "Enhanced flow table management scheme with an LRU-based caching algorithm for SDN," *IEEE Access*, vol. 5, pp. 25555–25564, 2017, doi: 10.1109/ACCESS.2017.2771807.
[19] L. Yang and H. Zhao, "DDoS attack identification and defense using SDN based on machine learning method," *Proc. - 2018 15th Int. Symp. Pervasive Syst. Algorithms Networks, I-SPAN 2018*, pp. 174–178, 2019, doi: 10.1109/I-SPAN.2018.00036.
[20] M. Paliwal, D. Shrimankar, and O. Tembhurne, "Controllers in SDN: A review report," *IEEE Access*, vol. 6, no. March 2011, pp. 36256–36270, 2018, doi: 10.1109/ACCESS.2018.2846236.
[21] M. Priyadarsini and P. Bera, "Software defined networking architecture, traffic management, security, and placement: A survey," *Comput. Networks*, vol. 192, no. June 2020, p. 108047, 2021, doi: 10.1016/j.comnet.2021.108047.
[22] I. Conference and D. Hutchison, *11th International Conference on Passive and Active Network Measurement, PAM 2010*, vol. 6032 LNCS. 2010.
[23] S. Bera, S. Misra, and A. Jamalipour, "FlowStat: Adaptive Flow-Rule Placement for Per-Flow Statistics in SDN," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 530–539, 2019, doi: 10.1109/JSAC.2019.2894239.
[24] S. Bera, S. Misra, and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1994–2008, 2017, doi: 10.1109/JIOT.2017.2746186.
[25] L. Ochoa Aday, C. Cervelló Pastor, and A. Fernández Fernández, "Current Trends of Topology Discovery in OpenFlow-based Software Defined Networks," Int. J. Distrib. Sensor Netw., vol. 5, no. 2, pp. 1–6, 2015, [Online]. Available: http://upcommons.upc.edu/handle/2117/77672
[26] A. Shalimov, D. Zimarina, and V. Pashkov, "Advanced Study of SDN / OpenFlow controllers," 2013.
[27] S. Kaur, J. Singh, and N. S. Ghumman, "Network Programmability Using POX Controller," *Int. Conf. Commun. Comput. Syst.*, no. August, p. 5, 2014, doi: 10.13140/RG.2.1.1950.6961.
[28] Z. K. Khattak, M. Awais, and A. Iqbal, "Performance evaluation of OpenDaylight SDN controller," *Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS*, vol. 2015-April, no. April 2015, pp. 671–676, 2014, doi: 10.1109/PADSW.2014.7097868.
[29] I. Z. Bholebawa and U. D. Dalal, "Performance analysis of SDN/openflow controllers: POX versus floodlight," *Wirel. Pers. Commun.*, vol. 98, no. 2, pp. 1679–1699, 2018, doi: 10.1007/s11277-017-4939-z.
[30] N. Gude, J. Pettit, and S. Shenker, "NOX: Towards an Operating System for Networks".
[31] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): a survey," *Secur. Commun. Networks*, vol. 9, no. 18, pp. 5803–5833, 2016, doi: 10.1002/sec.1737.
[32] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 4, pp. 1955–1980, 2014, doi: 10.1109/COMST.2014.2320094.
[33] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Comput. Networks*, vol. 75, no. PartA, pp. 453–471, 2014, doi: 10.1016/j.comnet.2014.10.015.
[34] S. Rao, "SDN and its use-cases-NV and NFV," *Network*, vol. 2, p. H6, 2014, [Online]. Available: http://www.nectechnologies.in/en_TI/pdf/NTI_whitepaper_SDN_NFV.pdf
[35] A. Blenk, A. Basta, J. Zerwas, M. Reisslein, and W. Kellerer, "Control Plane Latency with SDN Network Hypervisors: The Cost of Virtualization," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 3, pp. 366–380, 2016, doi: 10.1109/TNSM.2016.2587900.
[36] L. Leonardi, L. Lo Bello, and S. Aglianò, "Priority-based bandwidth management in virtualized software-defined networks," *Electron.*, vol. 9, no. 6, pp. 1–21, 2020, doi: 10.3390/electronics9061009.
[37] R. D. Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori, "VeRTIGO: Network virtualization and beyond," *Proc. - Eur. Work. Softw. Defin. Networks, EWSDN 2012*, no. March 2015, pp. 24–29, 2012, doi: 10.1109/EWSDN.2012.19.

[38] P. Skoldstrom and W. John, “Implementation and evaluation of a carrier-grade openflow virtualization scheme,” *Proc. - 2013 2nd Eur. Work. Softw. Defin. Networks, EWSDN 2013*, pp. 75–80, 2013, doi: 10.1109/EWSDN.2013.19.

[39] E. Salvadori, R. D. Corin, A. Broglio, and M. Gerola, “Generalizing virtual network topologies in OpenFlow-based networks,” *GLOBECOM - IEEE Glob. Telecommun. Conf.*, no. March 2015, 2011, doi: 10.1109/GLOCOM.2011.6134525.

[40] R. Sherwood, G. Gibb, and M. Kobayashi, “Carving research slices out of your production networks with open flow,” *Comput. Commun. Rev.*, vol. 40, no. 1, pp. 129–130, 2010, doi: 10.1145/1672308.1672333.

[41] Z. Bozakov and P. Papadimitriou, “AutoSlice: Automated and scalable slicing for software-defined networks,” *Conex. Student 2012 - Proc. ACM Conf. 2012 Conex. Student Work.*, no. December, pp. 3–4, 2012, doi: 10.1145/2413247.2413251.

[42] A. Al-Shabibi, M. de Leenheer, M. Gerola, A. Koshibe, W. Snow, and G. Parulkar, “OpenVirteX: A Network Hypervisor,” *Open Netw. Summit 2014 - Res. Track, ONS 2014*, 2014.

[43] A. Blenk, A. Basta, and W. Kellerer, “HyperFlex: An SDN virtualization architecture with flexible hypervisor function allocation,” *Proc. 2015 IFIP/IEEE Int. Symp. Integr. Netw. Manag. IM 2015*, pp. 397–405, 2015, doi: 10.1109/INM.2015.7140316.

[44] G. Yang, C. Shin, Y. Yoo, and C. Yoo, “A Case for SDN-based Network Virtualization,” *Proc. - IEEE Comput. Soc. Annu. Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst. MASCOTS*, [2021, doi: 10.1109/MASCOTS53633.2021.9614291.

[45] Y. Yoo, G. Yang, M. Kang, and C. Yoo, “Adaptive control channel traffic shaping for virtualized SDN in clouds,” *IEEE Int. Conf. Cloud Comput. CLOUD*, vol. 2020-Octob, pp. 22–24, 2020, doi: 10.1109/CLOUD49709.2020.00013.

[46] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, “Named Data Networking: A survey,” *Comput. Sci. Rev.*, vol. 19, no. January 2016, pp. 15–55, 2016, doi: 10.1016/j.cosrev.2016.01.001.

[47] N. Aloulou, M. Ayari, M. F. Zhani, and L. Saidane, “A popularity-driven controller-based routing and cooperative caching for named data networks,” *2015 Int. Conf. Netw. Futur. NOF 2015*, 2015, doi: 10.1109/NOF.2015.7333300.

[48] O. Karrakchou, N. Samaan, and A. Karmouch, “ENDN: An Enhanced NDN Architecture with a P4-programmabIe Data Plane,” *ICN 2020 - Proc. 7th ACM Conf. Information-Centric Netw.*, pp. 1–11, 2020, doi: 10.1145/3405656.3418720.

[49] A. Lara, A. Kolasani, and B. Ramamurthy, “Network innovation using open flow: A survey,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 493–512, 2014, doi: 10.1109/SURV.2013.081313.00105.

[50] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014, doi: 10.1109/SURV.2014.012214.00180.

[51] L. C. Costa *et al.*, “OpenFlow data planes performance evaluation,” *Perform. Eval.*, vol. 147, p. 102194, 2021, doi: 10.1016/j.peva.2021.102194.

[52] Siamak Azodolmolky, *Software Defined Networking with OpenFlow*, illustration ed. Packt Publishing 2013, pp. 62-138.

[53] Open Networking Foundation, “OpenFlow Switch Specification (Version 1.5.1),” *Current*, vol. 0, pp. 1–36, 2015, [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf

[54] “Pantou: Openflow 1.0 for openwrt. http://www.openflow.org/wk/index.php/ Open- Flow 1.0 for OpenWRT. Available: https://www.bing.com/search?q=Pantou%3A+Openflow+1.0+for+openwrt.+http%3A%2F%2Fwww.openflow.org%2Fwk%2Findex.php%2F+Open-+Flow+1.0+for+OpenWRT.&cvid=bb8ca5379f5446419bf1769c4586bef5&aqs=edge..69i57j69i58.1938j0j9&FORM=ANAB01&PC=U531 (accessed May 31, 2022).

[55] E. L. Fernandes *et al.*, “The road to BOFUSS: The basic OpenFlow userspace software switch,” *J. Netw. Comput. Appl.*, vol. 165, no. May, p. 102685, 2020, doi: 10.1016/j.jnca.2020.102685.

[56] T. Bakhshi, “State of the art and recent research advances in software defined networking,” *Wirel. Commun. Mob. Comput.*, vol. 2017, 2017, doi: 10.1155/2017/7191647.

[57] P. Software defined Networks: A comprehensive approach.Goransson, C. Black, and T. Culver, *Software defined Networks: A comprehensive approach.*, vol. 53, no. 9. 2017.

[58] O. Blial, M. Ben Mamoun, and R. Benaini, “An Overview on SDN Architectures with Multiple Controllers,” *J. Comput. Networks Commun.*, vol. 2016, 2016, doi: 10.1155/2016/9396525.

[59] Y. Su, T. Peng, X. Zhong, and L. Zhang, “Matching model of flow table for networked big data,” pp. 1–14, 2017, doi: 10.1007/978-981-13-6834-9_5.

[60] D. Kreutz, F. M. V. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” *HotSDN 2013 - Proc. 2013 ACM SIGCOMM Work. Hot Top. Softw. Defin. Netw.*, pp. 55–60, 2013, doi: 10.1145/2491185.2491199.

[61] R. Klöti, V. Kotronis, and P. Smith, “OpenFlow: A security analysis,” *Proc. - Int. Conf. Netw. Protoc. ICNP*, 2013, doi: 10.1109/ICNP.2013.6733671.

[62] Q. Li, J. Cao, M. Xu, and K. Sun, “Flow Table Overflow Attacks,” *Encycl. Wirel. Networks*, pp. 487–489, 2020, doi: 10.1007/978-3-319-78262-1_297.

[63] S. Gao, Z. Li, B. Xiao, and G. Wei, “Security Threats in the Data Plane of Software-Defined Networks,” *IEEE Netw.*, vol. 32, no. 4, pp. 108–113, 2018, doi: 10.1109/MNET.2018.1700283.

[64] L. Xinlong and C. Zhibin, “DDoS Attack Detection by Hybrid Deep Learning Methodologies,” *Secur. Commun. Networks*, vol. 2022, 2022, doi: 10.1155/2022/7866096.

[65] T. Ren and Y. Xu, “Analysis of the New Features of OpenFlow 1.4,” *Proc. 2nd Int. Conf. Information, Electron. Comput.*, vol. 59, no. Icieac, pp. 73–77, 2014, doi: 10.2991/icieac-14.2014.17.

[66] Open Networking Foundation, “OpenFlow 1.4 Specifications,” *Onf*, vol. 0, pp. 1–36, 2013.

[67] S. Khorsandroo, A. G. Sánchez, A. S. Tosun, J. M. Arco, and R. Doriguzzi-Corin, “Hybrid SDN evolution: A comprehensive survey of the state-of-the-art,” *Comput. Networks*, vol. 192, p. 107981, 2021, doi: 10.1016/j.comnet.2021.107981.

[68] P. Bosshart *et al.*, “P4: Programming protocol-independent packet processors. Computer Communication Review, 44(3),” *Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[69] Kfoury Elie, Crichigno Jorge, and Bou-Harb Elias, “Exhaustive P4 survey,” vol. 4, 2017.

[70] Y. Gao and Z. Wang, “A Review of P4 Programmable Data Planes for Network Security,” *Mob. Inf. Syst.*, vol. 2021, 2021, doi: 10.1155/2021/1257046.

[71] K. Kannan and S. Banerjee, “Compact TCAM: Flow entry compaction in TCAM for power aware SDN,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7730 LNCS, pp. 439–444, 2013, doi: 10.1007/978-3-642-35668-1_32.

[72] S. Banerjee and K. Kannan, “Tag-In-Tag: Efficient flow table management in SDN switches,” *Proc. 10th Int. Conf. Netw. Serv. Manag. CNSM 2014*, pp. 109–117, 2014, doi: 10.1109/CNSM.2014.7014147.

[73] C. R. Meiners, A. X. Liu, and E. Torng, “Bit weaving: A non-prefix approach to compressing packet classifiers in TCAMs,” *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 488–500, 2012, doi: 10.1109/TNET.2011.2165323.

[74] M. Malboubi, L. Wang, C. N. Chuah, and P. Sharma, “Intelligent SDN based traffic (de)Aggregation and Measurement Paradigm (iSTAMP),” *Proc. - IEEE INFOCOM*, pp. 934–942, 2014, doi: 10.1109/INFOCOM.2014.6848022.

[75] W. Braun and M. Menth, “Wildcard compression of inter-domain routing tables for OpenFlow-based software-defined networking,” *Proc. - 2014 3rd Eur. Work. Software-Defined Networks, EWSDN 2014*, vol. 12307, no. September, pp. 25–30, 2014, doi: 10.1109/EWSDN.2014.23.

[76] M. Rifai *et al.*, “Too many SDN rules? compress them with MINNIE,” *2015 IEEE Glob. Commun. Conf. GLOBECOM 2015*, pp. 0–6, 2015, doi: 10.1109/GLOCOM.2014.7417661.

[77] T. A. Pascoal, I. E. Fonseca, and V. Nigam, “Slow denial-of-service attacks on software defined networks,” *Comput. Networks*, vol. 173, no. October 2019, 2020, doi: 10.1016/j.comnet.2020.107223.

[78] C. R. Meiners, A. X. Liu, and E. Torng, “TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs,” *Proc. - Int. Conf. Netw. Protoc. ICNP*, vol. 18, no. 2, pp. 266–275, 2007, doi: 10.1109/ICNP.2007.4375857.

[79] G. Yang, B. Yu, W. Jeong, and C. Yoo, “FlowVirt : Flow Rule Virtualization for Dynamic Scalability of Programmable Network Virtualization,” *2018 IEEE 11th Int. Conf. Cloud Comput.*, pp. 350–358, 2018, doi: 10.1109/CLOUD.2018.00051.

[80] M. Zhang, J. Bi, J. Bai, Z. Dong, Y. Li, and Z. Li, “FTGuard: A priority-aware strategy against the flow table overflow attack in SDN,” *SIGCOMM Posters Demos 2017 - Proc. 2017 SIGCOMM Posters Demos, Part SIGCOMM 2017*, pp. 141–143, 2017, doi: 10.1145/3123878.3132015.

[81] N. Kim, D. Kim, Y. Jang, C. Lee, and B. D. Lee, “A new flow entry replacement scheme considering traffic characteristics in software-defined networks,” *Appl. Sci.*, vol. 10, no. 10, 2020, doi:

10.3390/app10103590.
[82] X. Li and Y. Huang, "A flow table with two-stage timeout mechanism for SDN switches," *Proc. - 21st IEEE Int. Conf. High Perform. Comput. Commun. 17th IEEE Int. Conf. Smart City 5th IEEE Int. Conf. Data Sci. Syst. HPCC/SmartCity/DSS 2019*, pp. 1804–1809, 2019, doi: 10.1109/HPCC/SmartCity/DSS.2019.00248.
[83] H. Nurwarsito, "Implementation of WLRU Algorithm to Improve Scalability in Software Defined Network," *Proc. SIET '20 5th Int. Conf. Sustain. Inf. Eng. Technol.*, pp. 165–170, 2020, doi: h?ps://doi.org/10.1145/3427423.3427444. 1.
[84] B. Yan, Y. Xu, and H. J. Chao, "Adaptive Wildcard Rule Cache Management for Software-Defined Networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 962–975, 2018, doi: 10.1109/TNET.2018.2815983.
[85] W. K. Jia, R. Ying, and X. Shi, "Deploying a Fast Detection and Eviction Mechanism of Invalid Connection-Oriented Flow-Entries in SDNs: A Scalability Approach," *IEEE Access*, vol. 8, pp. 208669–208682, 2020, doi: 10.1109/ACCESS.2020.3036437.
[86] S. Shirali-Shahreza and Y. Ganjali, "Delayed Installation and Expedited Eviction: An Alternative Approach to Reduce Flow Table Occupancy in SDN Switches," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1547–1561, 2018, doi: 10.1109/TNET.2018.2841397.
[87] U. Humayun, M. Hamdan, and M. N. Marsono, "Early Flow Table Eviction Impact on Delay and Throughput in Software-Defined Networks," no. August, pp. 7–12, 2021, doi: 10.1109/iccsce52189.2021.9530933.
[88] H. Choi, S. M. Raza, M. Kim, and H. Choo, "UDP flow entry eviction strategy using q-learning in software defined networking," *16th Int. Conf. Netw. Serv. Manag. CNSM 2020, 2nd Int. Work. Anal. Serv. Appl. Manag. AnServApp 2020 1st Int. Work. Futur. Evol. Internet Protoc. IPFutu*, no. 2019, 2020, doi: 10.23919/CNSM50824.2020.9269098.
[89] R. Challa, Y. Lee, and H. Choo, "Intelligent eviction strategy for efficient flow table management in OpenFlow Switches," *IEEE NETSOFT 2016 - 2016 IEEE NetSoft Conf. Work. Software-Defined Infrastruct. Networks, Clouds, IoT Serv.*, pp. 312–318, 2016, doi: 10.1109/NETSOFT.2016.7502427.
[90] M. K. A. Khan, V. K. Sah, P. Mudgal, and S. Hegde, "Minimizing Latency Due to Flow Table Overflow by Early Eviction of Flow Entries in SDN," *2018 9th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2018*, pp. 1–4, 2018, doi: 10.1109/ICCCNT.2018.8493926.
[91] H. Luo, W. Li, Y. Qian, and L. Dou, "Mitigating SDN Flow Table Overflow," *Proc. - Int. Comput. Softw. Appl. Conf.*, vol. 1, pp. 821–822, 2018, doi: 10.1109/COMPSAC.2018.00137.
[92] W. You, K. Qian, and Y. Qian, "Software-defined network flow table overflow attacks and countermeasures," *Int. J. Soft Comput. Netw.*, vol. 1, no. 1, p. 70, 2016, doi: 10.1504/ijscn.2016.077044.
[93] J. Xu, L. Wang, C. Song, and Z. Xu, "An Effective Table-Overflow Attack and Defense in Software-Defined Networking," *Proc. - 2019 IEEE 44th Local Comput. Networks Symp. Emerg. Top. Networking, LCN Symp. 2019*, pp. 10–17, 2019, doi: 10.1109/LCNSymposium47956.2019.9000663.
[94] G. Huang and H. Y. Youn, "Proactive eviction of flow entry for SDN based on hidden Markov model," *Front. Comput. Sci.*, vol. 14, no. 4, 2020, doi: 10.1007/s11704-018-8048-2.
[95] P. T. Duy, L. D. An, and V. H. Pham, "Mitigating flow table overloading attack with controller-based flow filtering strategy in SDN," *ACM Int. Conf. Proceeding Ser.*, pp. 154–158, 2019, doi: 10.1145/3371676.3371706.
[96] T. Xu, D. Gao, P. Dong, C. H. Foh, and H. Zhang, "Mitigating the table-overflow attack in software-defined networking," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 4, pp. 1086–1097, 2017, doi: 10.1109/TNSM.2017.2758796.
[97] Y. Qian, W. You, and K. Qian, "OpenFlow flow table overflow attacks and countermeasures," *EUCNC 2016 - Eur. Conf. Networks Commun.*, pp. 205–209, 2016, doi: 10.1109/EuCNC.2016.7561033.
[98] H. Yang and G. F. Riley, "Machine learning based flow entry eviction for OpenFlow switches," *Proc. - Int. Conf. Comput. Commun. Networks, ICCCN*, vol. 2018-July, pp. 1–6, 2018, doi: 10.1109/ICCCN.2018.8487362.
[99] Z. Guo, R. Liu, Y. Xu, A. Gushchin, A. Walid, and H. J. Chao, "STAR: Preventing flow-table overflow in software-defined networks," *Comput. Networks*, vol. 125, pp. 15–25, 2017, doi: 10.1016/j.comnet.2017.04.046.
[100] M. Lu, W. Deng, and Y. Shi, "TF-IdleTimeout: Improving efficiency of TCAM in SDN by dynamically adjusting flow entry lifecycle," *2016 IEEE Int. Conf. Syst. Man, Cybern. SMC 2016 - Conf. Proc.*, no. 61471060, pp. 2681–2686, 2017, doi: 10.1109/SMC.2016.7844645.
[101] H. Yang, G. F. Riley, and D. M. Blough, "STEREOS: Smart Table EntRy Eviction for OpenFlow Switches," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 377–388, 2020, doi: 10.1109/JSAC.2019.2959184.
[102] P. Nallusamy, S. Saravanen, and M. Krishnan, "Decision Tree-Based Entries Reduction scheme using multi-match attributes to prevent flow table overflow in SDN environment," *Int. J. Netw. Manag.*, vol. 31, no. 4, pp. 1–20, 2021, doi: 10.1002/nem.2141.
[103] S. Xie, C. Xing, G. Zhang, and J. Zhao, "A Table Overflow LDoS Attack Defending Mechanism in Software-Defined Networks," *Secur. Commun. Networks*, vol. 2021, 2021, doi: 10.1155/2021/6667922.
[104] B. Sooden and M. R. Abbasi, "A Dynamic Hybrid Timeout Method to Secure Flow Tables Against DDoS Attacks in SDN," *ICSCCC 2018 - 1st Int. Conf. Secur. Cyber Comput. Commun.*, pp. 29–34, 2018, doi: 10.1109/ICSCCC.2018.8703307.
[105] B. Isyaku, M. B. Kamat, K. Bin Abu Bakar, M. S. Mohd Zahid, and F. A. Ghaleb, "IHTA: Dynamic Idle-Hard Timeout Allocation Algorithm based OpenFlow Switch," *ISCAIE 2020 - IEEE 10th Symp. Comput. Appl. Ind. Electron.*, pp. 170–175, 2020, doi: 10.1109/ISCAIE47305.2020.9108803.
[106] B. Isyaku, K. A. Bakar, M. S. M. Zahid, and M. N. Yusuf, "Adaptive and hybrid idle–hard timeout allocation and flow eviction mechanism considering traffic characteristics," *Electron.*, vol. 9, no. 11, pp. 1–18, 2020, doi: 10.3390/electronics9111983.
[107] Y. Shen, C. Wu, Q. Cheng, and D. Kong, "AFTM: An Adaptive Flow Table Management Scheme for OpenFlow Switches," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Dec. 2020, pp. 917–922. doi: 10.1109/HPCC-SmartCity-DSS50907.2020.00122.
[108] Q. Li, N. Huang, D. Wang, X. Li, Y. Jiang, and Z. Song, "HQTimer: A Hybrid Q-Learning-Based Timeout Mechanism in Software-Defined Networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 1, pp. 156–166, 2019, doi: 10.1109/TNSM.2018.2890754.
[109] S. K. Noh, M. Kang, and M. Park, "Protection against Flow Table Overflow Attack in Software Defined Networks," *Int. Conf. Inf. Netw.*, vol. 2021-Janua, pp. 486–490, 2021, doi: 10.1109/ICOIN50884.2021.9333889.
[110] N. Priyanka, T. R. Reshmi, and K. Murugan, "CEOF: Enhanced Clustering-based Entries Optimization scheme to prevent Flow table overflow," *Wirel. Networks*, vol. 28, no. 1, pp. 69–83, 2022, doi: 10.1007/s11276-021-02823-8.
[111] M. Soylu, L. Guillen, S. Izumi, T. Abe, and T. Suganuma, "NFV-GUARD: Mitigating Flow Table-Overflow Attacks in SDN Using NFV," *Proc. 2021 IEEE Conf. Netw. Softwarization Accel. Netw. Softwarization Cogn. Age, NetSoft 2021*, pp. 263–267, 2021, doi: 10.1109/NetSoft51509.2021.9492584.
[112] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, and J. Shen, "Defending against flow table overloading attack in software-defined networks," *IEEE Trans. Serv. Comput.*, vol. 12, no. 2, pp. 231–246, 2019, doi: 10.1109/TSC.2016.2602861.
[113] X. Zhao, Q. Wang, Z. Wu, and R. Guo, "Method for Overflow Attack Defense of SDN Network Flow Table Based on Stochastic Differential Equation," *Wirel. Pers. Commun.*, vol. 117, no. 4, pp. 3431–3447, 2021, doi: 10.1007/s11277-021-08086-y.
[114] J. Xu, L. Wang, C. Song, and Z. Xu, "Proactive Mitigation to Table-Overflow in Software-Defined Networking," *Proc. - IEEE Symp. Comput. Commun.*, vol. 2018-June, pp. 719–725, 2018, doi: 10.1109/ISCC.2018.8538670.
[115] I. H. Abdulqadder, D. Zou, I. T. Aziz, and B. Yuan, "Validating User Flows to Protect Software Defined Network Environments," *Secur. Commun. Networks*, vol. 2018, 2018, doi: 10.1155/2018/1308678.
[116] Z. Guo *et al.*, "Balancing flow table occupancy and link utilization in software-defined networks," *Futur. Gener. Comput. Syst.*, vol. 89, pp. 213–223, 2018, doi: 10.1016/j.future.2018.06.011.
[117] S. Qiao, C. Hu, X. Guan, and J. Zou, "Taming the flow table overflow in OpenFlow switch," *SIGCOMM 2016 - Proc. 2016 ACM Conf. Spec. Interes. Gr. Data Commun.*, pp. 591–592, 2016, doi: 10.1145/2934872.2959063.
[118] T. Y. Chao, K. Wang, L. Wang, and C. W. Lee, "In-switch dynamic flow aggregation in software defined networks," *IEEE Int. Conf. Commun.*, pp. 2–7, 2017, doi: 10.1109/ICC.2017.7997429.
[119] S. Iot, N. Saha, S. Misra, and S. Bera, "QoS-Aware Adaptive Flow-rule Aggregation in Software-Defined IoT," pp. 2–7, 2018.
[120] Y. F. Liu, K. C. J. Lin, and C. C. Tseng, "Dynamic Cluster-based Flow Management for Software Defined Networks," *IEEE Trans. Serv. Comput.*, vol. 1374, no. c, pp. 1–11, 2019, doi: 10.1109/TSC.2019.2943852.
[121] T. H. Tsai, K. Wang, and T. Y. Chao, "Dynamic flow aggregation in SDNs for application-aware routing," *2016 10th Int. Symp. Commun.*

Syst. Networks Digit. Signal Process. CSNDSP 2016, pp. 6–10, 2016, doi: 10.1109/CSNDSP.2016.7573946.

[122]T. W. Chang, Z. H. Huang, Y. J. Chang, J. J. Kuo, and M. J. Tsai, "Isolation Guarantees with Flow Table Overflow in Software-Defined Networks," *2020 IEEE Glob. Commun. Conf. GLOBECOM 2020 - Proc.*, pp. 0–5, 2020, doi: 10.1109/GLOBECOM42002.2020.9322620.

[123]C. Wang and H. Y. Youn, "Entry aggregation and early match using hidden markov model of flow table in SDN," *Sensors (Switzerland)*, vol. 19, no. 10, 2019, doi: 10.3390/s19102341.

[124]F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN Control: Survey, Taxonomy, and Challenges," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 1, pp. 333–354, 2018, doi: 10.1109/COMST.2017.2782482.

[125]K. Sood, S. Yu, and Y. Xiang, "Performance analysis of software-defined network switch using M/Geo/1 model," *IEEE Commun. Lett.*, vol. 20, no. 12, pp. 2522–2525, 2016, doi: 10.1109/LCOMM.2016.2608894.

[126]H. G. Ahmed and R. Ramalakshmi, "Performance Analysis of Centralized and Distributed SDN Controllers for Load Balancing Application," *Proc. 2nd Int. Conf. Trends Electron. Informatics, ICOEI 2018*, no. May, pp. 758–764, 2018, doi: 10.1109/ICOEI.2018.8553946.

[127]S. Kaur, K. Kumar, N. Aggarwal, and G. Singh, "A comprehensive survey of DDoS defense solutions in SDN: Taxonomy, research challenges, and future directions," *Comput. Secur.*, vol. 110, p. 102423, 2021, doi: 10.1016/j.cose.2021.102423.