Received 5 May 2024; accepted 19 May 2024. Date of publication 31 May 2024; date of current version 20 June 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3406489

Security of Topology Discovery Service in SDN: Vulnerabilities and Countermeasures

SANAZ SOLTANI, ALI AMANLOU, MOHAMMAD SHOJAFAR (Senior Member, IEEE), AND RAHIM TAFAZOLLI (Senior Member, IEEE)

5G/6GIC, Institute for Communication Systems, University of Surrey, GU2 7XH Guildford, U.K.

CORRESPONDING AUTHOR: MOHAMMAD SHOJAFAR (e-mail: m.shojafar@surrey.ac.uk)

This work was supported in part by the U.K. Department for Science, Innovation, and Technology under Project 5G MoDE (Mobile oRAN for highly Dense Environments).

ABSTRACT Software-Defined Network (SDN) controller needs comprehensive visibility of the whole network to provide effective routing and forwarding decisions in the data layer. However, the topology discovery service in the SDN controller is vulnerable to the Topology Poisoning Attack (TPA), which targets corrupting the controller's view on the connected devices (e.g., switches or hosts) to the network and inter-switch link connections. The attack could cause dramatic impacts on the network's forwarding policy by changing the traffic path and even opening doors for Man-in-the-Middle (MitM) and Denial of Service (DoS) attacks. Recent studies presented sophisticated types of TPA, which could successfully bypass several well-known defence mechanisms for SDN. However, the scientific literature lacks a comprehensive review and survey of existing TPAs against topology discovery services and corresponding defence mechanisms. This paper provides a thorough survey to review and analyse existing threats against topology discovery services and a security assessment of the current countermeasures. For this aim, first, we propose a taxonomy for TPAs and categorise the attacks based on different parameters, including the attack aim, exploited vulnerability, location of the adversary, and communication channel. In addition, we provide a detailed root cause analysis per attack. Second, we perform a security assessment on the state-of-the-art security measurements that mitigate the risk of TPAs in SDN and discuss the advantages and disadvantages of each defence concerning the detection capability. Finally, we figure out several open security issues and outline possible future research directions to motivate security research on SDN. The rapid growth of the SDN market and the evolution of mobile networks, including components like the RAN Intelligent Controller (RIC) acting like SDN controller, highlight the critical need for SDN security in the future.

INDEX TERMS Software-defined network, SDN security, topology discovery service, topology poisoning attack.

I. INTRODUCTION

THE DEVELOPMENT of Software Defined Networking (SDN) started in 1996 by conducting several different projects and moved forward through the Stanford Ethan project in 2006 [1], [2]. The project aimed to create a logically centralised controller and a flow-based network focusing on access control security policy [3]. Organizations and businesses adjust their network configurations to keep up with the dynamic flow of information across different locations on the Internet. The complex nature of traditional networks poses challenges for data centres in implementing new services and connecting with various organizations. This is where SDN comes in handy.

The advent of SDN represents a revolutionary evolution in network architecture. In the traditional networking model, control and data forwarding functions were closely integrated within hardware network devices such as switches, routers, and load balancers, leading to rigid, static networks that required manual configurations for even minor changes. Adapting these networks to meet the dynamic demands of modern applications was a cumbersome and time-consuming task. SDN, on the other hand, introduces the concept of

softwarization by decoupling the control plane (i.e., network management and decision-making) from the data plane (i.e., switches) [3]. This decoupling empowers network administrators to configure and manage network behaviour through software, freeing them from the constraints of proprietary hardware. SDN offers an unprecedented level of flexibility and programmability, enabling networks to adapt in real-time to the evolving demands of modern applications and services. This transformation allows for swift network provisioning, dynamic traffic optimization, and rapid response to changing requirements, fundamentally altering the way we design and manage networks.

However, SDN principles amplify the importance of security due to its centralized control, creating a single point of failure that must be safeguarded against attacks. SDN's dynamic, programmable nature introduces new attack vectors, making it essential to protect against potential vulnerabilities that can compromise network traffic and configurations. Additionally, the automation and agility inherent in SDN can lead to rapid network changes, necessitating stringent security measures to prevent misconfigurations and unauthorized access, which can disrupt network operations and compromise data integrity. Two critical factors underscore the importance of SDN security.

Firstly, the rapid growth of the SDN market has made it a lucrative target for malicious actors seeking to exploit vulnerabilities in these dynamic and programmable networks. With the adoption of SDN, a large number of enterprises will benefit from its advantages, and according to the report in [4], the SDN market value was estimated at USD 24.64 billion in 2022, USD 28.37 billion in 2023, growing 15.58% to reach USD 78.52 billion by 2030. Therefore, the security of the SDN networks becomes crucial for many businesses since the attack can impact many of them. Consequently, SDN security should be seriously analyzed because the cost of an attack can be very high. According to a recent study by IBM, the average cost of a data breach reached a record high of USD 4.45 million in 2023—2.3% increase from the previous year's cost of USD 4.35 million, as reported in [5]. Moreover, it represents a noteworthy 15.3% surge from the average cost of USD 3.86 million reported in 2020, according to data from [6]. Consequently, the imperative for a meticulous and comprehensive analysis of SDN security protocols becomes evident, driven by the potentially exorbitant expenses resulting from security breaches in the SDN realm.

Secondly, the evolution of mobile networks towards softwarization further accentuates the significance of SDN security. Mobile networks have become an integral part of daily life for billions of users. The latest mobile network generations, including 5G and the highly anticipated 6G, are designed with a strong emphasis on SDN principles [7]. This shift highlights the mobile network's move toward becoming more software-driven. An excellent example of this transformation can be seen in the introduction of the Open Radio Access Network (Open RAN). In the context of Open RAN, the RIC plays a crucial role by essentially acting as the SDN controller within the network's architecture [8]. The transformation of mobile networks towards softwarization, coupled with the widespread reliance of billions of users on these networks, underscores the critical importance of SDN security. Any vulnerabilities in SDN have the potential to significantly impact the daily activities of these billions of users. As mobile networks increasingly depend on software-defined principles to operate efficiently and provide advanced services, safeguarding the integrity and security of these networks becomes paramount to ensure the uninterrupted and secure experiences of users worldwide.

The crucial question is whether the SDN could address the network security and privacy concerns and provide the required defence against various network attacks. Recent research shows that the SDN's programmability significantly improves the designing and development of software-based security defence, such as firewalls and Intrusion Detection Systems (IDS), against various network attacks. However, there are still many security gaps and vulnerabilities which could dramatically impact the network availability and performance. Numerous security threats to the SDN architecture have been outlined and explored in existing literature. Among these, the most critical attacks are those that target the control mechanism in SDN. If such an attack is successful in taking control of the entire network, it can lead to the unauthorized extraction of information or the execution of other harmful activities [9]. The adversary could damage or crash the SDN controller by compromising the critical assets. Network topology information is identified as the main asset in SDN, which needs to be protected against the Topology Poisoning Attack (TPA) [9]. In this attack, the adversary poisons the controller's perception regarding network-connected devices (e.g., switches or hosts) and inter-switch link connections. The attack could significantly impact the controller in traffic routing and forwarding decisions by changing the traffic path, providing the fertile ground to launch a Man-in-the-Middle (MitM) or Denial of Service (DoS) attack. Moreover, it could cause malfunctioning of topology-dependent applications and services, such as mobility management, load balancing, and congestion management, which can have several consequences such as poor Quality of Service (QoS) or Quality of Experience (QoE) to state a few [10].

The root cause of TPA is mainly attributed to several security vulnerabilities in the topology discovery service in the SDN controller, managed by two critical modules, namely, Host Tracking Service (HTS) and Link Discovery Service (LDS). There are some essential vulnerabilities of LDS [11], which could provide a range of attacks that have been categorised as Link Fabrication attacks (LFA) [12]. In LFA, the adversary intends to add a fabricated link between two switches. The adversary uses the security vulnerabilities of the OpenFlow Discovery Protocol (OFDP) to attack the network since the SDN controller leverages this protocol to obtain the topology information. The second

type of attack in TPA is categorised as an Identifier Binding Attack (IBA), resulting from the vulnerabilities in the HTS module. In this attack, the adversary hijacks the network identifier of the victim host, such as location information, IP address, or MAC address. Several defences have been proposed in the literature to mitigate LFA and IBA risks [12], [13], [14]. However, recent studies proposed the more sophisticated LFA and IBA, which could successfully bypass the mentioned defence systems.

The security of the SDN environment is essential for most businesses since the attacks could severely impact many of them and might cause revenue loss and reputation damage for the companies. Consequently, TPA and its root cause need to be seriously analysed because the cost of an attack can be very high [4]. Motivated by the mentioned considerations, we study a detailed analysis of all TPA threats against the SDN controller and the corresponding detection techniques used in the existing defence system.

Survey Organization: The structure of this study is illustrated in Fig. 1. Initially, in Section II, we discuss the related survey papers that studied the security aspect of topology discovery service and highlight our contributions. In Section III, we present a description of security in SDN architecture and provide an overview of the topology discovery service. The state-of-the-art Link Fabrication Attack and Identifier Binding Attacks are discussed and categorised in Sections V and VI, respectively. In Section VIII, the paper delves into the discussion of secured discovery protocols, emphasizing their role and significance in the context of the research. In Section IX, the paper explores real-world scenarios of TPAs in the industry, providing insights into practical implications and challenges. Challenges, open issues, and future research directions are addressed in Section X of the paper. Finally, we present the conclusion and future works in Section XI.

FIGURE 1. Structure of the paper.

II. RELATED HIGH-LEVEL ARTICLES AND THE SCOPE OF THIS SURVEY

In this section, we embark on a thorough examination of existing surveys relevant to our paper's topic, aiming to provide readers with a comprehensive understanding of the broader literature landscape. We analyze the topics covered in these studies and critically assess their limitations. Transitioning from this review, we outline the scope of our own survey, detailing its objectives, methodologies, and target audience. Finally, we highlight the unique insights our survey offers, emphasizing its significant impact on scholarly discourse and its potential to advance understanding in the field.

A. SURVEYS ON SDN SECURITY

This is the first study that provides a comprehensive technical security analysis of existing threats and countermeasures regarding topology discovery service in SDN to the best of our knowledge. The survey in [9] provides a partial review of a few existing threats against topology discovery services, including some related security countermeasures in the literature. However, our survey paper analyses all existing topology poisoning attacks and current defences from various security aspects and provides a detailed root cause analysis per attack. Table 1 provides a comparison of our survey with the surveys in [15], [9], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25] regarding covered topics.

TABLE 1. Comparison of existing survey papers about SDN security. Notations: ●: detailed discussion. ◐: partial discussion (i.e., at least one specialized section is presented but lacks a thorough examination). ○: not supported.

Covered topics	[15] (2024)	[16] (2023)	[17] (2022)	[18] (2021)	[19] (2020)	[20] (2019)	[21] (2018)	[22] (2018)	[23] (2017)	[9] (2016)	[24] (2015)	[25] (2015)	Our survey
Overview of SDN architecture	●	●	●	●	●	◐	●	●	●	●	●	●	●
Overview of SDN security	●	●	●	●	●	●	●	●	●	●	●	●	●
Overview of Topology Discovery Service in SDN	◐	●	◐	◐	○	●	◐	○	○	●	○	◐	●
Discuss the host tracking & link discovery services	◐	◐	◐	◐	○	○	◐	○	○	●	○	◐	●
Propose taxonomy of TPAs	○	○	○	○	○	◐	○	○	○	◐	○	○	●
Review and summarize all existing TPAs	○	○	○	○	○	◐	○	○	○	◐	○	○	●
Deep analysis on the attack causes and root causes	○	○	○	○	○	◐	○	○	○	◐	○	○	●
Detailed security assessment on current measurements	○	○	○	○	○	◐	○	○	○	◐	○	○	●
Discuss the advantages and disadvantage of current measurements	○	○	○	○	○	◐	○	○	○	◐	○	○	●
Evaluate & outline all current strategies to counteract TPAs	○	○	○	○	○	◐	○	○	○	◐	○	○	●
Analyze real-world SDN scenarios & applications with TPAs	○	○	○	○	○	○	○	○	○	○	○	○	●
Discuss the challenges in SDN topology discovery security & ML.	○	○	○	○	○	○	○	○	○	○	○	○	●
Explore the future integration directions of SDN and Open RAN.	○	○	○	○	○	○	○	○	○	○	○	○	●

In Table 1, we utilize a notation system to categorize the level of discussion corresponding to the subjects presented in our table. Specifically, we employ the following notation scheme: Solid Circle indicates a detailed discussion of the subject, offering thorough examination and analysis of the subject presented. Part Solid Circle denotes a partial discussion, where at least one specialized section in the paper is presented, but the examination may lack full depth or comprehensiveness. Whole Circle signifies that the subject is not supported and does not have any accompanying discussion or analysis. A quick glance at Table 1 reveals that our survey comprehensively covers all topics, as all circles are solid.

Kim et al.'s proposed study [15] aims to explore the security implications of SDN architecture. First, they provide an overview of SDN architecture and SDN security. The paper discusses the overview of topology discovery service in SDN, including host tracking and link discovery services, but the coverage is incomplete. They focus on how attackers can infiltrate different layers of SDN and the defensive strategies used to counter such attacks. To achieve this goal, they review reputable literature from conferences and journals. Their analysis results in the development of a classification for SDN attacks, considering their direction of infiltration, root causes, affected components, and common attack patterns.

Furthermore, they assess existing defence mechanisms suggested by researchers to mitigate these attacks. Through their meticulous examination of both attacks and defences, they highlight the vulnerabilities inherent in SDN architecture and pinpoint areas requiring further scrutiny from the security research community in future investigations [15]. The paper covers various aspects of SDN security; however, its breadth of coverage appears to have led to a lack of focus on specific areas such as topology discovery service and its security implications. By attempting to address all facets of SDN security, the paper may have sacrificed depth in certain areas. Consequently, the discussion on topology discovery service and its security is relatively brief and lacks the necessary attention to detail. This broad approach to SDN security coverage may have inadvertently detracted from a more thorough examination of individual components like topology discovery service.

The study proposed by Bhuiyan et al. [16] provides an overview of SDN architecture, SDN security, and Topology Discovery Service. It partially discusses host tracking and link discovery services. The paper delves into a thorough examination of attacks on SDN control planes, categorizing them and structuring the classification based on distinct attack surfaces: the Northbound Interface (NBI), Southbound Interface (SBI), SDN Controller, and interconnections between SDN controllers in a multi-controller setting. This systematic classification furnishes a well-organized comprehension of diverse attack vectors prevalent in SDN environments. Furthermore, the paper introduces a taxonomical representation of the identified attacks, establishing a systematic framework that aids in the analysis and understanding of their characteristics. This taxonomical approach provides a lucid and organized perspective on the attacks, facilitating researchers and practitioners in discerning relationships between different attack types. In addition to attack classification, the research extends to detailed countermeasure taxonomies aligned with the attack taxonomy. The paper delineates corresponding countermeasures for each attack category, offering practical guidance for implementing effective security measures in SDN environments. These countermeasure taxonomies serve as invaluable tools for securing SDN systems against potential threats. The paper concludes with a comprehensive research gap analysis, pinpointing limitations and research needs in SDN security. By shedding light on these gaps, the research contributes valuable insights for future researchers, enabling them to identify potential research directions and address current shortcomings in the field [16].

The proposed survey paper by Rahouti et al. [17] contributes by exploring SDN architecture layers, focusing on operations and functionalities. It underscores the importance of understanding security threats and using real-world use cases to motivate awareness. The paper provides a brief and insufficient overview of Topology Discovery Service in SDN, as well as host tracking and link discovery services. The study analyzes recent SDN-enabled applications in emerging areas like the Internet of Things (IoT), 5G, and Blockchain, emphasizing security enhancements. It provides an up-to-date overview of security challenges, reviews existing solutions categorized by SDN planes, and extracts practical implications. The paper then discusses future

research directions and open challenges, offering a forward-looking perspective for SDN-enabled systems' resilience and security [17].

Jimenez et al. [18] examined the SDN architecture and primary security concerns in it and then suggested ways to detect or mitigate them. Subsequently, they delved into the current security features and methods utilized in addressing these issues, pinpointing unresolved matters that could serve as foundations for future SDN security research initiatives [18]. Al-Heety et al. [19] proposed an in-depth examination of previous studies, categorizing them based on wireless communication, particularly in the context of Vehicular Ad-hoc Network (VANET). The study begins by providing a concise overview of the VANET structure, SDN controller, and its security, outlining their layers and infrastructure details. Following that, it delves into SDN-VANET applications in various wireless communication realms, including the IoT and VANET. The focus is on scrutinizing and comparing SDN-VANET works across multiple parameters. The paper not only analyzes open issues and research directions encountered during the integration of VANET with SDN but also sheds light on current and emerging technologies with real-world applications in vehicular networks. By addressing challenges in VANET infrastructure, the survey serves as a catalyst for enhancing the resilience of future SDN-VANET architectures, specifically in areas like routing protocol, latency, connectivity, and security [19]. However, it's important to note that the paper falls short of addressing the security aspects of SDN on its own, let alone delving into the security of the topology discovery service. Additionally, it should be highlighted that the survey lacks up-to-date information, which could impact its comprehensiveness in capturing the latest advancements and challenges in the field.

Marin et al. [20] systematically analyzed the security defences against topology attacks in current practices. Their investigation revealed six vulnerabilities in TopoGuard, TopoGuard+, Stealthy Probing-Based Verification (SPV), and SecureBinder. The researchers not only proposed and executed attacks against TopoGuard/TopoGuard+ but also presented compelling evidence of vulnerabilities in SPV and SecureBinder. Additionally, they identified significant security flaws in the topology services of Floodlight, a prominent SDN controller. Following responsible disclosure principles, the team notified Floodlight about the vulnerabilities they discovered. The researchers went further to introduce and practically demonstrate two innovative attacks—Topology Freezing and Reverse Loop—that could seriously impair the controller's network perspective. Recognizing that fully eliminating these attacks would necessitate major changes in the Floodlight controller, they suggested practical measures to mitigate such threats. Drawing from their findings, the study discusses potential ways to enhance existing topology countermeasures for better defence against link fabrication and host location hijacking attacks [20]. The paper, however, is not an up-to-date survey as it does not delve into newly discovered vulnerabilities or discuss recently implemented countermeasures. The paper fails to comprehensively discuss SDN architecture and the host tracking and link discovery services are entirely omitted. Notably, there is no focus on machine learning in their analysis. Additionally, the information provided in the paper is constrained by a limited bibliography, potentially limiting the breadth and depth of the coverage of relevant research in the field.

The proposed paper by Abdou et al. [21] outlines five control functions crucial for the effective operation of a realistic production network in delivering essential network services. Explores potential threats and defence mechanisms specific to these functions within Layer 2 networks, Layer 3 networks, and SDNs. Introduces a fresh evaluation framework for an unbiased comparison of security in both network paradigms, utilizing two threat models to specify the attacker's position within the network. It is important to note that this paper solely concentrates on control plane security and intentionally omits the discussion of TPAs and the security of the topology discovery service. They also present a summary of SDN architecture and its security aspects. The document examines the overview of topology discovery service within SDN, encompassing host tracking and link discovery services, but the coverage is extremely incomplete [21].

The proposed comprehensive review by Farris et al. [22] focuses on analyzing security mechanisms based on SDN and Network Functions Virtualization (NFV) for enhancing the protection of IoT systems. It begins with a systematic exploration of IoT security threats and the additional requirements in IoT environments. The authors briefly cover traditional security approaches for IoT, emphasizing authentication, encryption, access control, and detection solutions. The review then delves into an in-depth analysis of SDN and NFV architectures and security mechanisms, providing background information on these network paradigms and their integration into IoT systems. Notably, this survey is the first in the literature to systematically investigate the joint use of SDN- and NFV-based security mechanisms in the diverse landscape of IoT systems. Key security features are identified, offering a comprehensive overview of SDN/NFV-based security solutions and their application scenarios in cloud, core, and access IoT networks. The analysis includes a comparison with conventional security solutions, highlighting advantages and complementarity in various IoT environments and drawing lessons from experiences gained [22].

This survey proposed by Dargahi et al. [23] comprises an examination of recently introduced frameworks concerning the stateful SDN data plane. First, an overview of SDN architecture and its security is discussed. The proposed paper includes an assessment of the vulnerabilities present in existing stateful SDN data plane proposals. The survey further provides a tangible illustration of potential attacks on applications utilizing stateful SDN data plane switches. The focus is on case studies extracted from

diverse literature sources, strategically chosen to underscore distinct vulnerabilities. Additionally, the survey discusses the security implications inherited from traditional SDN and the enhancements in security introduced by the stateful SDN data plane. It engages in a discourse on potential defence strategies and offers recommendations for designing applications resilient to the aforementioned vulnerabilities. The survey concludes with insights into possible future research directions [23].

The survey paper proposed by Khan et al. [9] delves into a comprehensive exploration of SDN, shedding light on its layered architecture and the various threats it faces. A significant focus is placed on topology discovery, emphasizing its crucial role in both traditional networks and SDN environments. The paper goes on to establish a systematic thematic taxonomy, categorizing topology discovery into distinct groups, encompassing objectives, controller platforms, dependent services, discovery entities, and controller services. Furthermore, the survey discusses the landscape of topology discovery threats, presenting a classification that elucidates state-of-the-art security solutions. This includes insights into attack entities, controller vulnerabilities, attack types, and the occurrence of threats. The paper concludes by paving the way for future research directions in the realm of topology discovery within SDN, offering insights into potential areas of exploration and recommendations for plausible solutions [9]. However, it is essential to note a limitation in the surveyed paper, published in 2016, as it may not encompass the latest developments in the field. Given the dynamic nature of technology and the rapid evolution of SDN, there is a possibility that more recent research and advancements in topology discovery have emerged since the paper's publication. Moreover, their proposed paper does not include recent topology discovery attacks and an updated taxonomy. Following the proposed publication by Khan et al. [9], numerous countermeasures, primarily leveraging machine learning, have been proposed. It's important to note that these advancements are not covered in the surveyed paper, indicating a potential gap in addressing the evolving landscape of SDN security.

In the article proposed by Scott-Hayward and colleagues [24], an examination of SDN and its security is conducted, offering insights into advancements from both the research community and industry. The article delves into the challenges associated with safeguarding the network against persistent attackers and outlines the need for a comprehensive security architecture in the context of SDN. Furthermore, the authors highlight future research directions crucial for advancing network security in SDN [24]. However, it's important to highlight that the article does not cover TPAs and the security of the topology discovery service. Our survey addresses this gap for a more inclusive examination of SDN security.

Ahmad et al. [25] aspire to furnish a thorough depiction of SDN security by elucidating security issues and remedies pertaining to each distinct SDN plane—the application plane, control plane, and data plane. First, The paper examines the SDN architecture and its security aspects. Their work encompasses the delineation of network-wide security solutions and security development platforms within the SDN framework. Furthermore, they categorize security solutions in alignment with the International Telecommunication Union's Telecommunication (ITU-T) security recommendations and provide a succinct overview of the costs associated with diverse security architectures [25]. However, it is worth noting that the paper offers a deficient overview of the Topology Discovery Service in SDN, alongside host tracking and link discovery services. Despite providing valuable insights into network-wide security solutions and security development platforms, the review lacks coverage of TPA and the specific security considerations tied to the topology discovery service.

In our thorough examination of related review and survey papers, a noticeable gap emerged as none of them specifically addressed the crucial topic of Security of Topology Discovery Service in SDN, particularly in relation to Topology Poisoning attacks. Moreover, a comprehensive and up-to-date review of this critical aspect was conspicuously absent from the existing literature. Recognizing this void, we undertook the present survey to fill this gap and contribute to the understanding and awareness of security challenges in the realm of Topology Discovery Service within the context of SDN. Our survey aims to shed light on potential threats and vulnerabilities, providing a timely and thorough exploration of this essential but often overlooked facet of SDN security.

B. OUR SCOPE AND CONTRIBUTIONS

In this paper, we provide an extensive view of the topology discovery service in SDN, focusing on the security aspects. We intend to deliver the following contributions in this paper,

- Our survey provides a comprehensive exploration of the security landscape surrounding Topology Discovery Services in SDN. We trace the evolution of these services and analyze their vulnerabilities, with a particular emphasis on the emerging threat of topology poisoning attacks. This dual perspective not only offers a detailed tutorial on the subject but also paves the way for a deeper understanding of the path towards securing Topology Discovery Services in future SDN environments.
- We offer a comprehensive taxonomy, systematically categorizing TPAs. This classification considers various parameters such as the attack objective, target entity susceptibility, adversary's location, and the communication channel type.
- Our survey thoroughly examines TPAs, assessing their security implications. We also analyze state-of-the-art measures against TPAs in SDN, discussing challenges, and effective countermeasures, and evaluating detection capabilities for various types of attacks.
- Conducting a thorough review of current literature and ongoing research endeavours, our survey consolidates

the existing knowledge base on the security aspects of Topology Discovery Services in SDN, its vulnerabilities and its countermeasures.
- Explored the practical implications of TPAs by analyzing their manifestations in real-world scenarios, including Software-Defined Industrial Networks, Vehicular Networks, and Open RAN 5G environments, providing valuable insights into the specific challenges posed in these contexts.
- In addition to addressing current challenges, our paper outlines open issues and suggests future research directions. This includes exploring the secure integration of Topology Discovery Services with emerging technologies, emphasizing the role of machine learning and Open RAN in the 6G SDN domain.

C. METHODOLOGICAL APPROACH

In this section, we elucidate our meticulous methodology for selecting articles and conducting the survey. We commence by detailing our comprehensive search strategy, leveraging prominent online databases and crafting effective search strings to ensure a thorough exploration of relevant literature.

1) SEARCH STRATEGY

Crafting an effective search strategy is crucial for conducting a thorough survey. In our pursuit of vital research articles, we leveraged prominent online databases and digital libraries such as IEEE Xplore, Google Scholar, ACM Digital Library, Springer Link, Science Direct, Frontiers, Wiley and MDPI. To streamline our search, we devised multiple search strings and keywords. These included terms like Topology Discovery Service, Software-Defined Network, SDN, Security, Topology Poisoning Attack, TPA, Link Fabrication Attack, and Identifier Binding Attack. Furthermore, we employed Boolean logical operators to enhance the precision and accuracy of our search results. Below are some of the keywords and search strings utilized in this survey.

2) INCLUSION AND EXCLUSION CRITERIA

The criteria for selecting research papers are clearly outlined as follows,

1) Papers must be written and presented in English.
2) Focus should be on the SDN topology discovery service and its security.
3) Papers should be published between 2011 and 2024.
4) Publications must appear in academic journals, conference proceedings or letters.

Additionally, the following exclusion criteria were established to refine the selection process.

1) Duplicate papers were eliminated by searching across various online databases.
2) Papers with irrelevant titles were disregarded.
3) Papers lacking clear information were excluded.
4) Short papers containing fewer than 3 pages were not considered.
5) Papers from unknown journals or conferences were omitted.
6) Any papers not directly relevant to our topic were also excluded.

3) QUALITY ASSESSMENT RULES

The final step in compiling the list of selected research articles involves the application of Quality Assessment Rules (QARs). These rules are utilized to assess the quality of the chosen research articles. A set of 10 QARs has been proposed, with each rule carrying a score out of 10. The scoring criteria are as follows: a full answer earns 1 point, an above-average response earns 0.75 points, an average response earns 0.5 points, and a completely unanswered question receives 0 points. The total score for each article is determined by adding up the scores for all 10 QARs. Articles scoring 7 or lower are excluded from the analysis. The quality assessment rules employed to evaluate the articles are as follows,

1) Is the paper well-organized?
2) Are the research article's objectives clearly outlined?
3) Does the paper thoroughly explain the methodology used for countermeasures?
4) Are the various sections of the article coherent?
5) Does the paper include relevant background information?
6) Are the datasets utilized for training and testing clearly identified?
7) Does the paper include experiments conducted on diverse datasets?
8) Are the experiments conducted in a reasonable manner?
9) Does the paper comprehensively identify and report the results of experiments?
10) Ultimately, does the paper contribute value to the research community?

4) DATA EXTRACTION

Our survey involved a meticulous data retrieval process aimed at uncovering and assimilating insights from scholarly articles pertaining to the security aspects of SDN topology discovery services. Each chosen paper underwent rigorous scrutiny to ensure alignment with our predefined inclusion and exclusion criteria. Subsequently, we conducted a thorough examination to extract relevant data essential for this survey. Our focus encompassed various facets including research objectives, study contexts, proposed attack vectors, countermeasure architectures, algorithms employed, methodologies adopted, datasets utilized, and study outcomes. In the concluding phase, we meticulously synthesized and analyzed the amassed research data, furnishing a comprehensive overview of existing literature and delineating potential pathways for future research endeavours.

Fig. 2 illustrates the overview of our study selection process. Initially, we conducted a systematic search across various digital libraries and databases, yielding a total

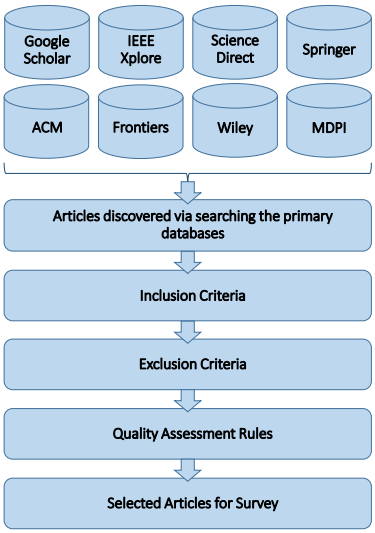

FIGURE 2. Methodological approach for article selection and survey conduct.

of 4557 research papers. Through a meticulous procedure involving stringent inclusion and exclusion criteria, rigorous quality assessments, and the elimination of duplicate papers, we identified research papers that met our standards for inclusion. These selected papers constitute a valuable and diverse body of literature, offering a wide array of perspectives, methodologies, and findings pertinent to our research topic. The thorough scrutiny applied during the selection process ensures that these selected papers possess the requisite quality and relevance to make substantial contributions to our survey, thereby enhancing the depth and breadth of our analysis. Moreover, all of these selected papers, sourced from high-impact digital libraries, are highly reliable due to their substantial number of citations and the esteemed reputation of their authors within the research community. Moreover, it is essential to note that we have incorporated several technical specifications sourced from both the O-RAN Alliance and 3GPP to delineate pivotal concepts within our research framework. These specifications have been rigorously verified and hold significant prominence in our investigation.

III. SOFTWARE DEFINED NETWORKS

The advent of the SDN paradigm changed the network methodology from node management to network management, simplifying the legacy network complexity and reducing Capex/Opex costs in FCAPS (fault management, configuration management, accounting management, performance management, and security management) [26]. In this section, we delve into the intricate workings of SDN and explore its various aspects. First, we thoroughly examine SDN architecture and its underlying principles, delineating its layers and operational mechanisms in Section III-A. Following this, we shift our focus to recent advancements and practical applications of SDN across diverse domains in Section III-B, showcasing its transformative potential in network management and operations. Lastly, we shed light on the critical role of the OpenFlow protocol in Section III-C, providing an overview of its functionalities and the communication protocols between the SDN controller and OpenFlow-compliant switches. For convenience, Table 2 refers to all abbreviations used in this paper.

TABLE 2. Summary of main abbreviations.

Abbreviation	Description
ACL	Access Control List
API	Application Programming Interface
BDDP	Broadcast Domain Discovery Protocol
BMP	Bearer Migration Poisoning
DQL	Deep Q-learning
DHCP	Dynamic Host Configuration Protocol
DoS	Denial of Service
DPID	Datapath Identifier
HLHA	Host Location Hijacking Attack
HMAC	Keyed-Hash Message Authentication Code
HTS	Host Tracking Service
IBA	Identifier Binding Attack
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
LFA	Link Fabrication Attack
LDS	Link Discovery Service
LLA	Link Latency Attack
LLI	Link Latency Inspector
LLDP	Link Layer Discovery Protocol
LLDPDU	Link Layer Discovery Protocol Data Unit
MHL	Multi-hop Link
MitM	Man-in-the-Middle
ML	Machine Learning
NBI	NorthBound Interface
NFV	Network Functions Virtualization
NOS	Network Operation System
nRT RIC	Near-Real Time RAN Intelligent Network
OFDP	Open Flow Discovery Protocol
ONF	Open Networking Foundation
O-CU-CP	O-RAN Central Unit – Control Plane
O-CU-UP	O-RAN Central Unit – User Plane
RADIUS	Remote Access Dial-In User Service
RAN	Radio Access Network
RIC	RAN Intelligent Controller
RL	Reinforcement Learning
RTT	Round Trip Time
SDL	Shared Data Layer
SDN	Software-Defined Network
SBI	SouthBound Interface
TPA	Topology Poisoning Attack
TLS	Transport Layer Security
TLV	Type-Length-Value
TTL	Time-To-Live

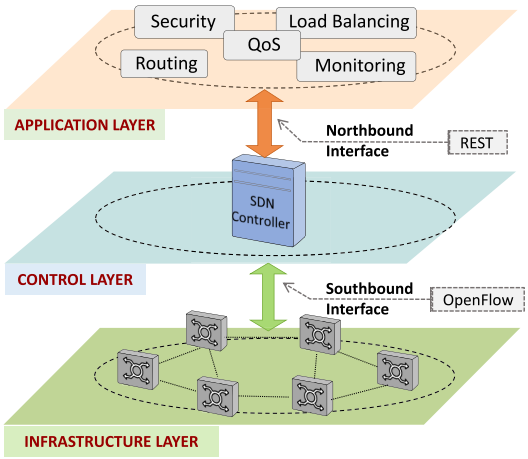

FIGURE 3. Software Defined Network architecture.

A. SDN ARCHITECTURE

SDN architecture presents a promising solution to address the limitations of traditional IP network architecture in terms of control, scalability, and management. The fundamental concept of SDN involves separating the control plane from the data plane. In this approach, the control logic is extracted from network hardware and centralized into a software-based entity known as the SDN controller. This stands in contrast to conventional IP networks, where routers handle both packet forwarding (data plane) and execute routing protocols to discover network paths and make routing decisions (control plane). Illustrated in Fig. 3 is an overview of the SDN architecture, consisting of three primary layers.

- *Data plane.* The data plane, or infrastructure layer, comprises a network of interconnected forwarding elements, commonly referred to as SDN switches. These switches can be either physical hardware devices or software-based virtual switches, like Open vSwitch [27]. Unlike traditional IP networks, SDN switches lack the capability to analyze incoming packets autonomously and instead forward all packets to the control plane. This simplification enables SDN switches to be more straightforward and cost-effective compared to traditional routers, resulting in significant simplification of network configuration and management.
- *Control plane.* The control layer encompasses a logically centralized, software-based SDN controller. This controller is tasked with overseeing and setting up the individual SDN switches by implementing the necessary forwarding rules. A primary responsibility of the controller is to establish and sustain a comprehensive overview of the network. Topology discovery, which is the main focus of this paper, is a crucial function offered by the SDN control layer.
- *Application plane.* The application layer comprises a collection of network management applications where high-level network policy decisions are formulated and executed. Through the installation of specific rules, a controller application can instruct SDN switches to carry out various functions, including routing, switching, firewalling, network address translation, load balancing, and more.

The connection between the control layer and the infrastructure layer, which is utilized for overseeing and setting up the individual SDN switches, is commonly known as South Bound Interfaces (SBI). The SDN controller employs different SBIs through management and control protocols like OpenFlow [28], NETCONF [29], and OpFlex [30]. OpenFlow stands out as the predominant standard for southbound interfaces, extensively utilized and integrated into SDN environments. Section III-C will delve further into OpenFlow, offering relevant details for our subsequent discussions in the paper.

SDN also empowers network application developers to access a simplified representation of data plane hardware and dynamically program the network as needed via standard Northbound Interfaces (NBI) like REST [31] and OpenStack Neutron [32]. Orchestration software and network applications, encompassing tasks such as routing, load balancing, security, and third-party applications, obtain essential services from the controller using NBI [33]. This capability has led to the widespread reference of the controller layer as the Network Operation System (NOS) [34].

B. ADVANCEMENTS AND APPLICATIONS OF SDN

SDN significantly enhances network programmability, fostering a culture of rapid innovation. Through an application operating on the controller, new network services, applications, and policies can be easily implemented. This application governs the data plane's forwarding elements via appropriate abstractions and a well-defined Northbound Interface (NBI) and Southbound Interface (SBI). Furthermore, SDN's ability to facilitate network virtualization is crucial in various deployment scenarios, especially in data centre applications, achieved through tools like FlowVisor [35] or OpenVirteX [36]. Simultaneously, Mininet [37] has gained prominence as a lightweight virtualization-based simulator for emulating large-scale SDNs. These advantages, among others, have spurred significant industry interest, leading to a surge in SDN-enabled switches and devices offered by both established and emerging vendors, alongside a variety of SDN controller platforms.

The transition from traditional infrastructure to SDN is cost-effective and technically feasible. Consequently, hybrid SDN has emerged as a preferred transitional solution for many organizations, blending the strengths of traditional networking and SDN technology in a balanced compromise [38]. Hybrid SDN integrates centralized and decentralized approaches to configure, control, and manage network behaviour. Unlike traditional switches that use distributed algorithms such as Interior Gateway Protocol (IGP) for traffic routing management, an SDN controller

routes traffic based on a global perspective. This integration results in a hybrid SDN architecture where some traffic follows conventional routes while others are controlled by the SDN controller. Google's B4 project [39] exemplifies this approach, where an SDN application is integrated with a routing protocol to enable interaction between SDN switches and traditional routing protocols like OSPF.

C. OPENFLOW PROTOCOL

OpenFlow [28] is an open standard protocol for controller-to-switch communication, designed and developed by the Open Networking Foundation (ONF) [40]. The SDN controller uses the OpenFlow channel to pass the required instructions and messages toward the OpenFlow switch. To provide a secured communication channel, OpenFlow offers the Transport Layer Security (TLS) [41] protocol to encrypt all transmitted messages. OpenFlow permits a controller to retrieve and adjust the forwarding rules of SDN switches, granting control over the network's traffic flow.

1) OPENFLOW SDN SWITCH

An OpenFlow switch needs to support a flow-table pipeline in which every switch contains a chain of flow tables, starting at zero. The switch installs received instructions as flow table entries in the flow tables. Every entry in the flow table includes three elements: rule, action, and statistics.

An OpenFlow switch must adhere to the fundamental match-action model. This means that every incoming packet is compared against a series of rules, and the action or sequence of actions linked with the matching rule is carried out. The matching process operates on a flow basis and is relatively detailed. OpenFlow supports various match fields, including the ingress port of the switch, different packet header attributes like MAC source and destination addresses, IP source and destination addresses, as well as UDP/TCP source and destination port numbers, among others.

One of the primary functions of an OpenFlow switch is to direct a packet to a specific port on the switch or to discard the packet [42]. These forwarding rules not only include physical ports but also encompass virtual ports such as ALL (which sends the packet out through all physical ports), CONTROLLER (sending it to the SDN controller via an OpenFlow Packet-In message), and FLOOD (similar to ALL but excluding the port where the packet entered). When a switch receives data packets through any of its ports that it cannot forward based on existing rules, it can send them to the controller. This is achieved using an OpenFlow Packet-In message (`OFPT-PACKET-IN`). For instance, this mechanism is employed when a switch encounters a packet that doesn't match any forwarding rule. In such instances, the default action is for the switch to send the packet to the controller encapsulated within an `OFPT-PACKET-IN` message.

After receiving the packet, the controller can determine and implement the appropriate forwarding rules for the new flow. Alternatively, sending packets to the controller can be managed through specific rules established on the switch itself. For instance, in the present SDN topology discovery process, each switch comes with a pre-configured rule indicating that all topology discovery packets (identified by their EtherType 0x88cc) should be directed to the CONTROLLER port. In addition to the Packet-In function, OpenFlow also supports a Packet-Out message (`OFPT-PACKET-OUT`). With this message, the controller can dispatch a data packet to a switch along with instructions on how to handle it, presented as a list of actions. For instance, the controller might instruct the switch to forward the packet through a designated port. Both OpenFlow Packet-In and Packet-Out messages play crucial roles in the topology discovery mechanisms elaborated in the subsequent sections.

Additionally, the SDN controller uses the OpenFlow channel to collect required network information from connected switches. To query the switch capabilities and active port identities, it uses the `OFPT-FEATURES-REPLY` message. Traffic statistic measurements such as the total number of sent or received packets can be reported using the `OFPT-STATS-REPLY` message [42].

2) PACKET COMMUNICATION PROCESS

The OpenFlow switch examines the header fields of incoming packets and checks them against entries in the flow table. If there's a match, the packet is forwarded as instructed. Otherwise, it's forwarded based on a default entry in the table-miss. Typically, unmatched packets are sent to the controller for further processing through `OFPT-PACKET-IN` messages. Each application on the controller is responsible for analyzing these messages, extracting pertinent information, and making appropriate decisions.

In the network, there are two kinds of `OFPT-PACKET-IN` messages depending on where they come from. One type occurs when the switch initiates a forwarding action by sending an `OFPT-PACKET-IN` message containing a data frame to the controller. The other type involves the controller triggering a message to the switch, resulting in an `OFPT-PACKET-IN` message containing control packets.

After receiving `OFPT-PACKET-IN` messages, the controller delegates them to the relevant applications for handling. For link discovery within the network, the link discovery application initiates the process by sending an `OFPT-PACKET-OUT` message containing an LLDP packet to the switch. Subsequently, it retrieves specific fields (like DataPath ID and Inport) from the `OFPT-PACKET-IN` messages sent by the switch to establish and maintain links. Meanwhile, the topology application utilizes this link information to create a comprehensive network topology view. Additionally, the forwarding application monitors `OFPT-PACKET-IN` messages triggered by data packets. It extracts essential fields (such as source and destination IP addresses) from these messages to determine the route for the data flow. It then issues either an `OFPT-FLOW-MOD` message (representing a flow rule) or an `OFPT-PACKET-OUT`

message (representing a flood instruction) to facilitate packet forwarding accordingly.

IV. TOPOLOGY DISCOVERY SERVICE

Topology discovery service in network environments refers to the technology aimed at automatically identifying the components of a network, such as routers and switches, and establishing their relationships. This process is crucial for effective network management, providing insights into the network's structure and connectivity. In the specific context of SDN, the controller needs to maintain a consistent network topology for the infrastructure layer to understand all connected hosts, switches, and inter-link switch connections to effectively manage the network.

In this section, we delve into the critical aspect of topology discovery service within SDN and its significance in network management and security. First, we highlight the paramount importance of topology discovery in maintaining network integrity and optimizing traffic flow. This foundational understanding is further explored in Section IV-A, where we underscore the fundamental role of topology discovery. Following this, in Section IV-B, we trace the evolution of topology discovery methods from traditional networks to SDN environments, demonstrating how these methods have adapted to meet the unique challenges posed by SDN architectures. Sections IV-C–IV-E focus on switch, host, and link discovery processes, respectively, essential for effective network management. Section IV-F addresses the dynamic nature of network topology in SDN environments and the mechanisms used to update topology information in real-time. Lastly, Section IV-G discusses security considerations associated with topology discovery, ensuring network integrity and resilience against potential threats.

A. IMPORTANCE OF TOPOLOGY DISCOVERY

Many topology-based applications and services within SDN, including packet routing, mobility management, load balancing, and congestion management, depend on the controller's access to accurate network topology information. Therefore, in SDN environments, a highly accurate and efficient topology discovery service becomes essential. It serves as the foundation for optimizing network configuration, identifying bottlenecks, pinpointing faults, and uncovering potential risks. Any inaccuracies or outdated information regarding network topology within the SDN Controller can directly impact the services and applications dependent on SDN. Therefore, the topology discovery service within the SDN Controller holds significant importance.

B. TOPOLOGY DISCOVERY IN TRADITIONAL NETWORKS AND ITS ADAPTATION IN SDN

Topology discovery service in SDN leverages the foundational principles of general network topology discovery while introducing enhancements to accommodate the dynamic and programmable nature of SDN environments. Thus, we explore the main methods that exist for discovering and understanding network topologies, which can be broadly classified as discovery using Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), Routing Information Protocol (RIP), Open Shortest Path First (OSPF), Link-Layer method based on Simple Network Management Protocol (SNMP) and Discovery using IP-Layer method based on SNMP [43].

ICMP convey crucial information such as network and router availability, and terminal reachability status, among others, providing insights into the network's current condition. Leveraging these attributes, ICMP proves valuable in topology discovery. ARP is vital in networking, translating IP addresses to physical addresses for message transmission. Each network entity maintains an ARP cache. Consequently, by referencing the ARP table of any switch or router, all entities within the network can be identified. The RIP exchange the route information between network nodes. The RIP employs hop count as a routing metric. We can deduce the connections between routers through the hop counts. Thus, RIP can be used for topology discovery. OSPF gathers link state information from available routers and constructs a topology map of the network.OSPF detects changes in the topology, such as link failures, very quickly and converges on a new loop-free routing structure within seconds. The topology of the network can be generated by collecting the OSPF messages. The Link Layer Discovery Protocol (LLDP) based on SNMP get the transmit table of switches through SNMP protocol. The LLDP is introduced through the IEEE802.1AB [44] standard developed by IEEE in 2002. The transmit table is a Management Information Base (MIB) table mapping the ports of switches to the physical addresses of the entities that linked to the ports and can be used in topology discovery. Using SNMP for IP-layer discovery involves gathering data from the IPRouterTable, which contains details like destination IP addresses, router equipment, and the IP address of the next hop. This information allows us to construct network topology.

The SDN topology discovery service improves above traditional discovery techniques to better suit the dynamic and flexible characteristics of SDN environments. One notable improvement involves the integration of the OpenFlow Discovery Protocol (OFDP), which employs the LLDP method for efficient link discovery within SDN networks.

SDN achieves network topology discovery through three primary methods: Switch Discovery Service, HTS (Host Tracking Service) and LDS (Link Discovery Service).

C. SWITCH DISCOVERY SERVICE

An OpenFlow-based switch is assumed to be configured with the IP address and TCP port number of its controller. On startup, a switch will contact its controller on the corresponding IP address and TCP port, and establish a TLS of the session to secure the connection. Consequently, to enter the network, the Switch initiates a TCP session through a three-way handshake (`SYN`, `SYN/ACK`, `ACK`) to kickstart communication with the controller. Following this,

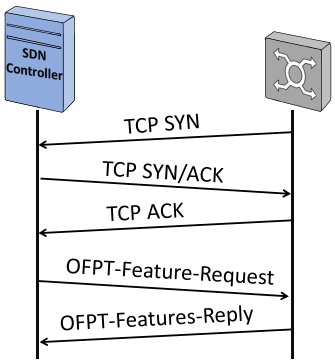

FIGURE 4. Switch Discovery Service.

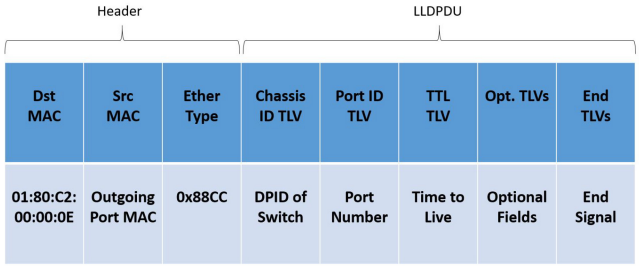

FIGURE 5. LLDP packet frame.

the controller dispatches an `OFPT-Features-Request` message to the switch, soliciting its current setup such as the MAC address and network interfaces. Subsequently, the switch responds with an `OFPT-Features-Reply` message containing the requested details. The controller then stores and utilizes this data for subsequent network management activities, including re-processing of topology discovery. Fig. 4 illustrates the switch discovery process in SDN.

D. HOST TRACKING SERVICE

A network node is specified based on various identifiers which are related to the different network layers. The identifiers address traffic flow from a source to destination(s). They also impose network and security policies, such as load balancing rules and an Access Control List (ACL). In SDN, key identifiers are specified in network layers. Data Path Identifier (DPID) and *port number* are defined for each switch in the physical layer which is also referred to as network location. Media Access Control (MAC) address and Internet Protocol (IP) address are used in data link and network layers, respectively, to route the flows between switches. When the identifier is allocated to the network node, it is bound to the current identifiers of the node. For example, in an Address Resolution Protocol (ARP) request message, the nodes' IP address is bound to the received MAC address. This process is called identifier binding which associates identifiers of a node to each other. If the binding results from a protocol such as ARP, Dynamic Host Configuration Protocol (DHCP), or predefined configuration, it is known as an explicit binding. However, if it is realized from receiving network traffic in `PACKET-IN` message, such as binding MAC address to switch DPID and port in HTS, it is called an implicit binding. The identifier binding process is critical for the correct functioning of the network protocols because the protocols use the mapping of identifiers to deliver the packet from the source to the destination(s).

In a dynamic network, hosts might frequently join, migrate or leave the network, impacting the network topology. To keep track of the host's location, HTS continuously monitors the `PACKET-IN` message received from switches. When a host joins the network, HTS creates a host profile and inserts host location information extracted from the received `PACKET-IN` message into the host profile. This data could be a combination of IP address, MAC address, and location information (associated switch DPID, switch port, and last seen timestamp). A host might migrate between two switches in the network, triggering the location update in the host profile. In such a case, upon receiving the `PACKET-IN` message, the HTS realizes that information elicited from the `PACKET-IN` message is different from the corresponding recorded data and subsequently updates the host profile with the new location information. If a host leaves the network, the corresponding switch sends a `Port-down` message toward the controller. By receiving the message, HTS removes the host profile and changes the status of the port to down. Apart from the mentioned events, HTS regularly sends probing `Packet-In` messages to query host location information and keep the host profile information updated.

E. LINK DISCOVERY SERVICE

Link Discovery Service (LDS) plays a pivotal role in maintaining an accurate and up-to-date view of the network topology. This service is particularly crucial in environments where network dynamics are pronounced, such as in data centres or telecommunication networks. The controller utilizes the OFDP to form the logical view of the network. The OFDP involves periodic transmissions of the Link Layer Discovery Protocol (LLDP) messages from the controller to OpenFlow switches.

The process of link discovery involves the SDN controller actively gathering information about links among switches. It distinguishes between links within OpenFlow switches as internal links, and those spanning Non-OpenFlow switches as external links. More specifically, it identifies internal links through the LLDP and external links through Broadcast Domain Discovery Protocol (BDDP) packets. The BDDP protocol is a special solution that is programmed in open-source SDN controllers including OpenDayLight and Floodlight to explore and discover the external links in a hybrid OpenFlow network [45].

Fig. 5 exhibits Floodlight's specification for LLDP packet format, detailing a frame header with Dst MAC, Src MAC, and Ether Type components. During internal link discovery, the controller configures the destination MAC

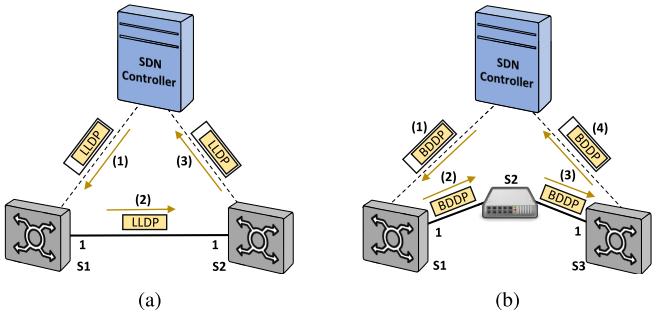

FIGURE 6. Illustration of the link discovery service. (a) OFDP for internal links, (b) OFDP for external links.

address of LLDP packets to a fixed multicast address (01:80:C2:00:00:0E). Simultaneously, it assigns the source MAC address (Src MAC) to the switch port's hardware address, and sets the ether type to '0x88cc'. On the other hand, BDDP packets follow a frame structure akin to LLDP, differing primarily in the destination MAC address (FF:FF:FF:FF:FF:FF) and ether type ('0x8942'). The controller harnesses the broadcast capability of BDDP packets to facilitate transmission across Non-OpenFlow switches. Within LLDP's payload, the Link Layer Discovery Protocol Data Unit (LLDPDU) comprises multiple Type-Length-Value (TLV) structures. The controller utilizes mandatory TLVs like Chassis ID, Port ID, and Time-To-Live (TTL) to identify links. Optional TLVs, such as Controller ID, augment LLDP's functionality, while end TLVs mark the conclusion of LLDP packets. The illustration in Fig. 6 showcases the process of identifying both internal and external links within the SDN network.

1) TWO OPENFLOW SWITCHES

In OFDP, the controller sends periodic LLDP messages to the switches. The Cisco Discovery Protocol (CDP) [46] is a proprietary alternative but less widely used. Fig. 6(a) presents a network with an SDN controller and two OpenFlow switches, namely, s_1 and s_2. The controller issues the LLDP packets, and each switch provides a suitable response to the received packet. Each LLDP response packet includes the DPID of the switch with a Port ID. For example, the controller sends LLDP packets to s_1 (see Fig. 6(a)). By receiving LLDP packets via a *Packet-Out* message, s_1 distributes it to all interfaces except the one connected to the controller. When the destination switch s_2 receives the LLDP from interface port 1, i.e., p_1, the switch encapsulates it as a `PACKET-IN` message and sends it to the controller. Upon receiving LLDP, the controller realizes an internal link between switch s_1 and switch s_2 [47].

2) OPENFLOW SWITCH AND NON-OPENFLOW SWITCH

Moving from conventional network architecture to SDN introduces hybrid SDN deployment, where legacy or Non-OpenFlow switches coexist with OpenFlow switches in the network [48]. The controller of the hybrid SDN network cannot directly monitor and control the legacy section. Therefore, LLDP protocol packets are transmitted between switches to share the switch and its adjacents' knowledge in the legacy network. Whenever a Non-OpenFlow switch catches an LLDP packet from its adjacent switch, it gathers knowledge and discards the LLDP packet. Therefore, the LLDP packet is able to wend only one hop in the link layer. Accordingly, the SDN controller uses BDDP to explore and discover the external link. Fig. 6(b) displays an external link connecting switch s_1 and switch s_3 via a legacy switch, s_4. To start, the controller initiates internal link discovery by sending an LLDP packet to switch s_1. However, when the Non-OpenFlow switch s_2 receives this packet, it disregards it due to the TTL value reaching 0. Following this, the controller dispatches a BDDP packet containing the broadcast destination MAC address to $s1$. Subsequently, the packet traverses to switch s_3, which in turn encapsulates the BDDP packet within the Packet-In message and forwards it to the controller. By analyzing these Packet-In messages, the controller identifies the presence of an external link between s_1 and s_3 [49].

F. TOPOLOGY UPDATE PROCESS

The process of determining the topology instance involves several key stages. Firstly, it begins by identifying clusters within the network. This entails categorizing ports connected to external links or multiple internal links as broadcast domain ports, while switches connected via non-broadcast domain ports form distinct clusters. Within each cluster, the controller proceeds with the second step by systematically analyzing switch-linked connections. It identifies intra-cluster links and categorizes the remaining links, such as external connections, as inter-cluster links. Moving on, the third step involves the identification of archipelagos. The controller merges multiple clusters interconnected by inter-cluster links into separate and isolated entities. Next, the fourth stage includes calculating multiple paths between switches within a given archipelago. Typically, the controller computes three paths, selecting the shortest path as the broadcast tree for that specific archipelago. Finally, the fifth step focuses on determining broadcast ports for each archipelago. These designated ports include switch link ports connected to other switches within the broadcast tree, as well as switch edge ports linked to terminal devices. This meticulous approach optimizes topology navigation and enhances the overall management of the network's structure.

G. SECURITY IN TOPOLOGY DISCOVERY SERVICE

SDN architecture suffers from several security weaknesses, increasing the risk of attacks due to malicious switches or hosts. First, a host can easily direct a packet toward the controller by triggering a table-miss entry on a switch and initiating a `PACKET-IN` message. This feature provides a malicious host with the opportunity to flood the switch with packets not matching flow rules, thereby resulting in a denial of service (DoS) attack [50] on the switch and controller.

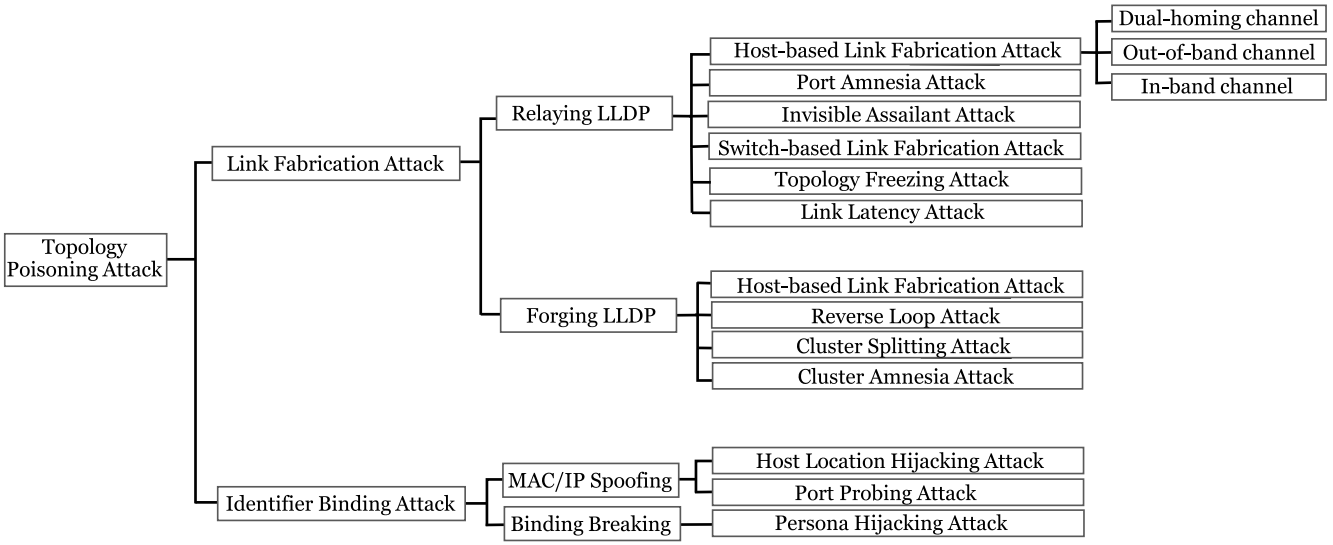

FIGURE 7. Taxonomy of Topology Poisoning Attack in SDN.

Consequently, it can cause network crashes and outages. Second, the controller sends traffic routing instructions as flow rules toward switches. It opens up the possibility for an untrusted switch to re-route or hijack the traffic by modifying the installed flow rules. It also allows a compromised host to leak sensitive information about flow rules. Third, malicious hosts and switches can inject forged or spoofed messages into the network while the controller has no built-in detection mechanism to raise security alarms.

TPA targets corrupting the perception of the SDN controller on the connected devices (e.g., switches or hosts) to the network and inter-switch link connections. It could significantly impact the network's forwarding policy by changing the traffic path. The reasons are mainly attributed to several security vulnerabilities in HTS and LDS modules in the controller. The lack of LLDP verification in the LDS module triggers a new group of attacks, called LFA. In addition, HTS suffers from IBA which allows the adversary to overtake the victim's identifiers. Fig. 7 illustrates our proposed taxonomy of existing TPA threats, categorized based on the attack aim of poisoning the controller view.

V. LINK FABRICATION ATTACK AND SECURITY COUNTERMEASURES

LFA is an example of the TPA in which the adversary intends to add a fabricated link between two switches. The traffic routed over the forged link can be intercepted, manipulated, or dropped, thereby degrading the performance of the network. It can even lead to a black hole if the traffic traverses over a forged link directed toward the malicious host. While different variants of LFA have been proposed in the literature, their strategies to poison the perception of the controller are categorized into two main parts, namely, LLDP Relay-Based LFA and LLDP Forgery-Based LFA.

In LLDP Relay-Based LFA, the fabricated link is created by relaying LLDP between two switches. Depending on the physical distance between two switches, the adversary can opt for an appropriate relaying channel. If the host can be directly connected to two switches, a dual-homed host topology [51] is used which implements a bridging technique to forward LLDP (see Fig. 8(a)). When switches have more distance, the adversary provides an out-of-band communication channel, e.g., a direct ad-hoc link or wireless connection between two hosts (see Fig. 8(b)). The third possibility is utilizing an in-band connection between two compromised hosts [52]. To this end, the adversary uses internal network infrastructure, e,g., wireless access points or routers, which could create a fabricated link between two switches far from each other in a different part of the network (see Fig. 8(d)). In LLDP Forgery-Based LFA, adversaries tamper with LLDP frames, specifically by modifying TLVs within these packets. By manipulating TLVs, attackers can create fake links between switches, deceiving the SDN controller about the actual network topology (see Fig. 8(e)).

While both LLDP Relay-Based LFA and LLDP Forgery-Based LFA share the goal of misleading the SDN controller, identifying relay-based attacks presents a notable challenge. These attacks operate without altering the LLDP packet itself, maintaining a stealthy profile that avoids triggering any anomalies in the LLDP process. In contrast, forgery-based attacks entail modifications to TLVs, potentially altering the packet structure and thereby offering a slightly more feasible detection pathway. In this section, we delve into an exhaustive exploration of various forms of LFA, elucidating their mechanisms, potential risks, and associated countermeasures.

A. HOST-BASED LINK FABRICATION ATTACK (RELAYING LLDP)

When a switch, i.e., s_1 (see Fig. 8(b)) receives an LLDP packet from the controller, it broadcasts the packet through

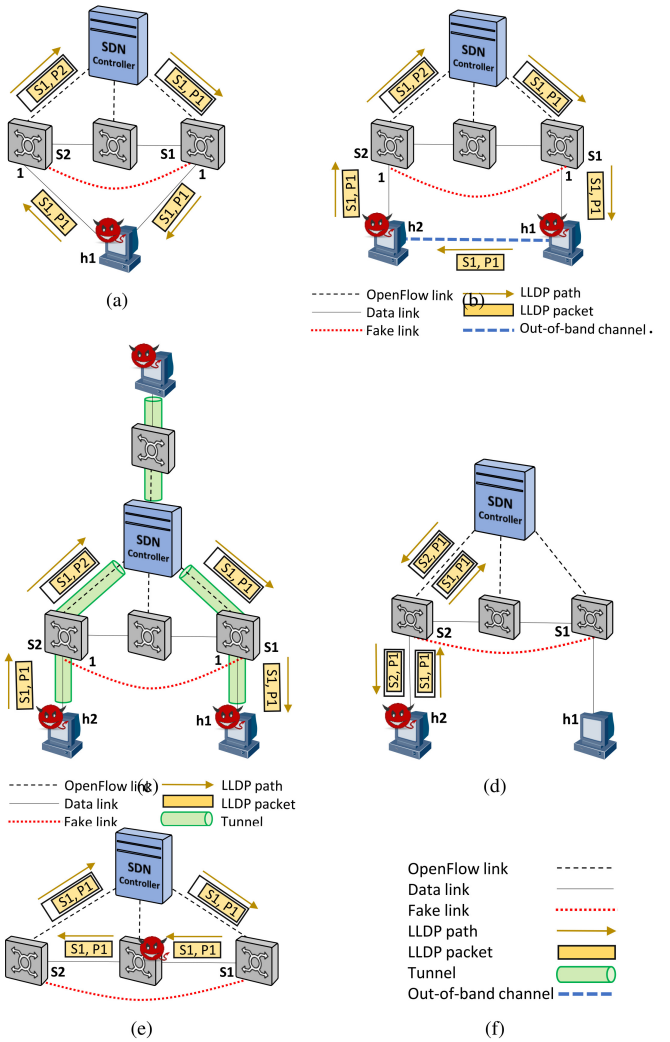

FIGURE 8. Different attack strategies in launching LFA: (a). Bridging LLDP using dual-homed compromised host, (b). Relaying LLDP using an out-of-band channel, (c). Relaying LLDP using an in-band channel, (d). Injecting a fake LLDP, (e) Relaying LLDP using a compromised switch.

all its ports. Hence, if a host or another switch is connected to this switch, it will also receive the LLDP packet. In a normal scenario, hosts ignore the receiving LLDP packet, and no action has been triggered. However, in an attack scenario, the adversary who controls two hosts, i.e., h_1 and h_2, does not ignore the LLDP packet, and h_1 relays the packet toward the second switch, i.e., s_2 by using an out-of-band channel. The adversary could alternatively utilize an in-band channel (see Fig. 8(d)) or duel-home host (see Fig. 8(a)) to relay the received LLDP. When the second switch receives the LLDP packet, it forwards the packet toward the controller, where the LDS module is tricked into believing that there is a link between these two switches, which are not really connected [12].

Countermeasure: The problem of relay-based LFA lies in that during the link discovery procedure, hosts could listen to LLDP packets through its port which is connected to a switch. To restrict the LLDP propagation path only to the switch device, a real-time port verification solution is proposed in TOPOGUARD [12] framework. In this solution, a port classification technique is used and each switch port is marked based on first-seen traffic. A port is labeled as SWITCH if the LLDP is forwarded while it is tagged as HOST if host traffic is generated. If no device is connected to the port, it is nominated as ANY. By using this strategy, the controller rejects the LLDP packet received from the ingress port tagged with the HOST label. The main drawback of TOPOGUARD is that the adversary can compromise a host and pretend its label as a SWITCH by forwarding LLDP messages.

The work in [53] also proposes a threshold-based defence for this attack using statistical analysis of link latency distribution. When a new link is added, it is initially monitored for a specific duration, called the vetted period. During the interval, the controller prevents normal traffic to traverse over the newly added link. The vetted period lasts till a specific number of link latency samples are collected. By finishing the vetted period, collected link latency values are compared against a predefined threshold. The threshold is a baseline distribution of normal latency values. However, while it offers a method to monitor and detect fake links, its effectiveness heavily relies on predefined thresholds and baseline latency values. Adversaries might find ways to manipulate latency or create scenarios where the threshold isn't breached, thus evading detection, and potentially compromising the system's security.

To protect controllers from topology poisoning attacks, SPHINX [14] develops a security framework as a middle layer between the SDN controller and the data plane. It intercepts and analyses four OpenFlow control messages, including `FLOW-MOD`, `PACKET-IN`, `STATS-REPLY` and `FEATURES-REPLY`, to learn network behaviour and metadata. Specifically, when the switch sends a `PACKET-IN` message for the controller, SPHINX extracts all flow-specific topology metadata, i.e., host IP-MAC and MAC-Port bindings. When the controller originates a `FLOW-MOD` toward the switches, flow path information, and rerouting updates are extracted. Flow statistics data, such as the number of sent or received packets, and switch configuration information, such as all active ports, are also extracted on receipt of `STATS-REPLY` and `FEATURES-REPLY` messages on the controller, respectively.

SPHINX uses the extracted metadata from `PACKET-IN` and `FEATURES-REPLY` to build and update an incremental flow graph that models network topology, where edges represent the metadata and switches are nodes. While `FLOW-MOD` and `STATS-REPLY` messages enable SPHINX to construct the data plane forwarding part of the graph. Then, SPHINX provides a policy verifier engine for validation of flow graphs against collected metadata over time and pre-defined security policies specified by the administrator (e.g., access control) which are expressed in policy language. It continuously monitors the flow graphs for permissible changes and raises alerts if it identifies deviant behaviour. SPHINX validates

topological metadata gleaned from malicious PACKET-IN and FEATURES-REPLY messages against a set of policies, such as permitting a port at a switch to have only one single neighbour and allowing bidirectional links between two switches. In case of any violation and deviant behaviour, SPHINX flags a security incident.

Gao and Xu [54] have put forward an innovative approach to addressing security issues in the process of discovering network topologies in SDNs. Initially, they examine current principles and threat models related to topology attacks. In their proposed research, they introduced a new approach to counteract topology poisoning attacks. They establish a legal condition detection mechanism for host migration to combat host hijacking attacks. Next, they develop LLDP source check and integrity check methods to counter link fabrication attacks. They also introduce a relay-type link fabrication attack detection approach based on entropy calculation for constructing LLDP frames. The defence mechanisms discussed in this work are integrated within the SDN controller itself. They operate autonomously from the controller's other functional modules. Lastly, they create an SDN simulation environment using Mininet and Floodlight controllers. The findings confirm the efficacy of their solution against common topology attacks and demonstrate its ability to offer thorough and comprehensive topology security protection.

B. PORT AMNESIA ATTACK

Port amnesia attack [13] can bypass TOPOGUARD defence by exploiting the port classification module. In this attack, when the compromised host, i.e., h_1 (see Fig. 8(b)), receives the LLDP packet, it relays the packet toward the peer compromised host, i.e., h_2. In h_2, before sending LLDP toward the switch s_2, the adversary disconnects and then connects the h_2 connection to the switch. This action triggers a Port-down message toward the controller, causing the resetting label of the port from HOST to ANY. When the controller receives the LLDP from the port labelled with ANY, verifies LLDP path legitimacy and announces the new link. In the case of using an in-band channel (see Fig. 8(d)) to relay the LLDP packet, both compromised hosts, i.e., h_1 and h_2, need frequent context-switching between HOST and SWITCH labels which imposes an extra complexity on the attack. This is because compromised hosts need to have a HOST label when they forward data-plane traffic while their label must set SWITCH in case of using the fabricated link.

Countermeasure: TOPOGUARD+ [13] mitigates the risk of port amnesia attack by adding two modules to TOPOGUARD, namely Link Latency Inspector (LLI) and Control Message Monitor (CMM). The former module detects the attack if the adversary uses the out-of-band channels between compromised hosts, while the latter protects the network against in-band LFA. The strategy of LLI to detect a fake link is the extra latency that the out-of-band channel imposes on LLDP propagation duration. TOPOGUARD+ periodically issues probe packets to measure the Round Trip Time (RTT)

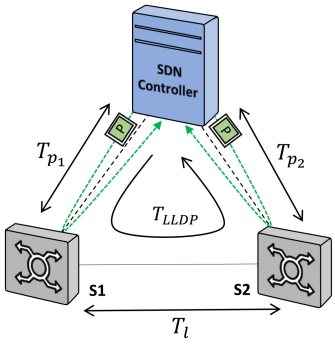

FIGURE 9. RTT monitoring in TOPOGUARD+.

between the controller and switches. Upon receiving the packet, each switch provides a suitable response to that packet. Assume that we have two switches, namely, s_1 and s_2, that are connected to a controller. Also, suppose that T_{p_1} and T_{p_2} are the corresponding link latency of the probe packets sent to s_1 and s_2. The *LLI* computes the inter-switch link latency T_l using eq. (1).

$$T_l = T_{LLDP} - T_{p_1} - T_{p_2}, \tag{1}$$

where T_{LLDP} indicates the propagation delay of the LLDP packet. To calculate the T_{LLDP}, the controller adds a timestamp to the issued LLDP packet toward the switches and takes the difference when receiving it. The RTT monitoring in TOPOGUARD+ is visually illustrated in Fig. 9, showcasing LLI's method for evaluating inter-switch link latency.

Moreover, LLI stores the values of inter-switch latency based on received LLDP packets and calculates a latency threshold as shown in eq. (2).

$$T_h = q_3 + 3 * (q_3 - q_1), \tag{2}$$

where q_1 and q_3 indicate the lower and upper quartiles of stored latency values, respectively. By comparing latency T_l and threshold T_h, LLI verifies the validity of the link and raises a security alarm in case of suspicious delay (i.e., $T_l > T_h$).

The LLI module's limitations include its vulnerability to high false alarms, particularly when normal links are removed during network peak hours. Additionally, the LLI module could be bypassed if the adversary gradually increases the latency of the threshold, i.e., T_h. For this purpose, the adversary overloads one switch in the network during a long period to slightly grow the latency of the LLDP packet in each propagation round [20]. Eventually, this approach leads to a high value for the threshold which is greater than the latency of the out-of-band channel. However, it needs hours of preparation.

The CMM module aims to monitor abnormal port connections and disconnections during LLDP packet forwarding to detect in-band port amnesia attacks. In in-band port amnesia, the adversary frequently resets the adversary port to forward both LLDP traffic and host-generated traffic without raising labelling violations. This means that before sending host

traffic, the port label should be set as HOST, while during sending the LLDP packet, it should be set as SWITCH. Hence, the CMM module monitors the frequency of port down and up during the LLDP packet forwarding process and in case of any abnormal port connection and disconnection, it generates security alerts. Despite its focus on monitoring port status changes during LLDP forwarding, the CMM module may encounter difficulty in detecting sophisticated attacks. Adversaries cleverly manipulate these alterations, staying within expected frequency thresholds to avoid triggering immediate alarms.

Deng et al. [55] conducted a thorough examination of the match field within the OpenFlow protocol and exposed vulnerabilities between applications and the data plane. They identified commonly used sensitive match fields by applications to construct functions. Through manipulation of these fields, they proposed a sensitive field manipulation attack aimed at overwhelming network bottlenecks. The attacker begins by gathering operational information about the application through the transmission of probing packets and test streams containing various sensitive fields. By exploiting timing disparities among probing packets, the attacker floods the bottleneck with the most efficient test streams, thereby preventing applications from receiving responses from the data plane, and ultimately causing core service crashes.

Additionally, the authors explored threats to SDN architecture by impeding the delivery of LLDP packets using such attacks. In certain scenarios, this can corrupt network topology, deplete controller computing resources, and disrupt core services. To validate the feasibility of the sensitive field manipulation attack, they conducted evaluations in a physical setting. The findings demonstrated that the attack surpasses previous methods in effectiveness and leads to extensive consequences. To identify and counter such attacks, the authors initially examined current defence systems and their efficacy against such manipulative actions. They discovered that existing systems are ineffective in thwarting attacks that exploit sensitive fields to overwhelm network bottlenecks.

Given that attackers base their actions on probing outcomes, the authors developed SFieldDefender to identify malicious probes and test flows. This lightweight extension of the existing controller employs a combination of machine learning model training and a multi-policy mechanism for attack mitigation. Data is collected based on constructed features and used to train multiple models. The most efficient model for attack detection, determined by latency and precision, is then selected. Additionally, various defence strategies are devised for different types of abnormal traffic. A prototype of SFieldDefender is implemented within the Floodlight controller and evaluated for its effectiveness. Experimental results demonstrate that SFieldDefender adeptly detects sensitive field manipulation attacks and safeguards network services.

The paper proposed by Deng et al. [56] reveals a significant vulnerability found in widely used open-source SDN controllers, which can be exploited by malicious or compromised applications to launch different types of attacks. The vulnerability stems from the testing features commonly integrated into SDN controllers. These controllers often employ testing applications to mimic the behaviour of regular applications, ensuring consistency with the OpenFlow protocol, the standard southbound protocol regularly updated with new features. To ensure correctness during development, controllers offer a special internal application interface (API) to generate various crucial SDN messages (such as `Packet-In` messages) without requiring permissions.

Unfortunately, this mechanism inadvertently allows sophisticated attackers to create a backdoor for applications to send spoofed malicious packets. Since SDN applications typically process data plane messages sequentially, the spoofed message can be forwarded to all subsequent benign applications. What's more concerning is the absence of an integrity-checking mechanism, meaning the falsified messages could corrupt the entire network topology and disrupt various services. Upon examining five mainstream SDN controllers (like Floodlight, ONOS, OpenDaylight, POX, and Ryu), it was discovered that all of them are vulnerable to this newly disclosed exploit. They then proceed to investigate the threats posed to various network services by exploiting the previously mentioned vulnerability.

They introduced three types of attacks aimed at disrupting the topology discovery and updating processes across different layers. Specifically, they illustrate that by leveraging the backdoor vulnerability to falsify network messages, attackers can not only seize control of network communication among hosts but also obstruct the operations of other applications by fabricating host location data. Moreover, they demonstrate that adversaries can clandestinely manipulate or invent connections. Additionally, they introduce two novel attacks, namely the archipelago division attack and the topology freezing attack, designed to hinder the establishment of a comprehensive global topology view. They carry out experiments on real SDN testbeds with multiple switches to assess the effectiveness of these attacks. Their results reveal that a malicious application can intercept webpage traffic between hosts, trigger information leaks, and launch various denial-of-service attacks against both SDN applications and hosts. Regrettably, current defence mechanisms from both the data and application planes prove ineffective against their attacks.

C. INVISIBLE ASSAILANT ATTACK

Kong and colleagues [57] introduced the Invisible Assailant Attack (IAA), a sophisticated tactic capable of infiltrating networks by injecting fake links, even when 12 different defence mechanisms are active simultaneously. IAA comprises 14 intricately planned phases employing various strategies to disguise attack traffic as regular network activity. By executing these phases methodically, attackers evade existing defences step by step. To counter this threat, they proposed the Route Path Verification (RPV) mechanism,

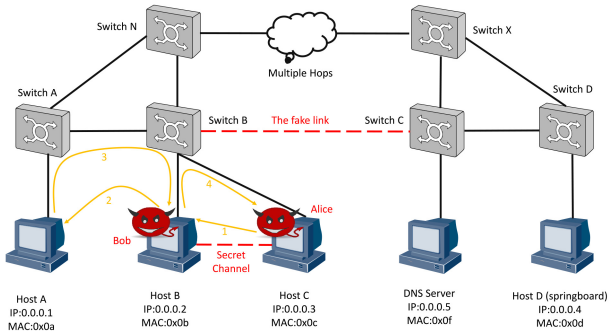

FIGURE 10. The depiction of IAA. It's important to note that each switch maintains a separate connection with the controller. For simplicity, these connections are not shown. This figure is the precise arrangement of the network. The arrows 1, 2, 3, and 4 depict the communication flow between host C and A.

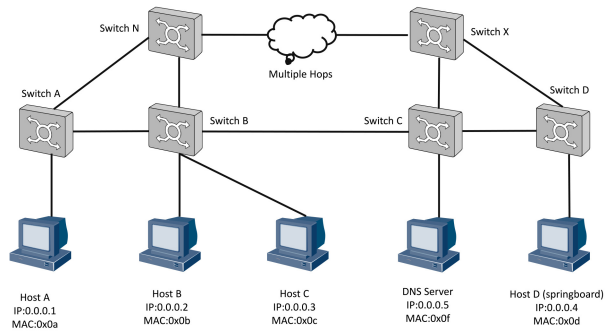

FIGURE 11. The depiction of IAA. This figure shows the poisoned perspective of the controller. It's important to recognize that every switch is individually connected to the controller, although these connections are not shown for simplicity.

which coordinates multiple defence strategies to identify counterfeit links. Experimental results demonstrate that RPV effectively detects IAA with minimal overhead, completing detection within 1 millisecond and consuming only a small amount of storage per flow [57].

IAA takes advantage of weaknesses in the link discovery service to inject a fake link, a tactic similar to previous research. However, what sets IAA apart is its ability to camouflage its attack packets within the regular network traffic, thus circumventing defensive measures. Fig. 10 shows the real network topology, whereas Fig. 11 shows the distorted view of the network that IAA aims for the controller to perceive, known as the poisoned view. In Fig. 10, the attack strategies employed by IAA are illustrated, involving two compromised hosts. In this scenario, one attacker (referred to as Alice, who has commandeered *Host C*) inserts the fake link, while the other attacker (referred to as Bob, who has compromised *Host B*) aids in maintaining the deception by disguising the packets generated by Alice to avoid detection.

In Fig. 10, Alice creates a false connection between *Switch B* and *Switch C* by passing LLDP packets between them (relaying LLDP), a technique previously employed in other attacks. However, when *Host C* communicates with other hosts, it inevitably sends packets indicating the first hop. If these packets are intercepted by the controller on an internal port, defence mechanisms will trigger alerts. Attackers struggle to predict which packets might reveal the false link, making it challenging for them to adjust their strategies. This is why existing attacks struggle to maintain false links for long periods. For instance, even though ICMP queries from compromised hosts can uncover the fake link, attackers are hesitant to discard them since ICMP queries are commonly used in normal network operations. Additionally, ARP packets and other standard network traffic can also expose the fake link, making it exceedingly difficult for attackers to anticipate the detection methods employed. However, existing attack methods fail to address this challenge, resulting in easily detectable fake links. So, the challenge is figuring out how to sustain the false connection once it's been injected. Bob tackles this by setting up *Host B* to act as a middleman, passing on messages between *Host C* and other parties. This means *Host C* only interacts with *Host B*. As illustrated in Fig. 10, in this setup, Bob's communication with other hosts won't raise any red flags. However, when Bob communicates with Alice, it's crucial to camouflage it as regular network activity to avoid detection.

To establish covert communication between Alice and Bob, the IAA process involves four key steps. Initially, before initiating the attack, the attackers must predefine specific parameters of the secret communication channel. This allows them to differentiate their communication packets from regular network traffic. Next, once the fake link is injected successfully, Alice must identify a standard host as a "springboard." This host's packets will be unwittingly used to mask the communication between attackers. Subsequently, Alice notifies Bob to verify whether the identified host meets the criteria of a suitable springboard. Finally, Alice and Bob can utilize the unwitting assistance of the springboard to test if they can conceal their communication within normal packet transmissions. Throughout this process, IAA must also employ additional measures to counter various defence strategies. For further details, please refer to paper [57].

Countermeasure: Kong and colleagues [57] developed a new security measure called Route Path Verification (RPV), which detects potential attackers by ensuring the integrity of route paths. Experimental findings suggest that this mechanism is proficient at identifying IAA.

RPV is a new approach developed from an interesting discovery: whenever a regular packet is sent to the controller due to Table-Miss, the controller can effectively track down the original sender of that packet. Let's illustrate this with an example: if a switch (let's call it *switch n*) notifies the controller about a new flow (let's call it *f*), the controller can backtrack to its immediate previous hop (*switch n - 1*) by analyzing the in-port on *switch n*. Then, the controller can verify if flow *f* indeed arrived at *switch n - 1* by querying the stored flow rules. If the relevant rule is found, the controller proceeds to trace back to *switch n - 2* based on the in-port. This iterative process continues until the controller reaches the original source host of flow *f*. However, it's important

to note that this method doesn't work for malicious packets injected by IAA.

Kong et al. [57] describe this quality as the *integrity of route paths*. From this insight, they developed RPV, a security measure aimed at detecting attacks by ensuring the integrity of route paths. RPV works by gathering topology and flow data from the network and using it to reconstruct the route path for each flow through the controller. When a new packet arrives at the controller due to Table-Miss, RPV checks the route path's integrity to validate the packet. If the packet passes the check, RPV logs it along with the current switch to continue building the route path. If a host exits the network or moves to a new switch, RPV removes all route paths originating from that host.

To improve RPV's effectiveness, two key scenarios must be addressed. Firstly, when a host migrates, it can prompt the controller to discard previous route paths linked to that host. If malicious hosts trigger fake migrations, they could evade route path verification. To counter this, RPV verifies hosts upon network entry. Only authenticated hosts can initiate route path construction. Thus, the authentication of hosts and route path verification form a crucial verification chain. The chain's security hinges on host authentication, which has been proven trustworthy. Hence, ensuring host authentication guarantees the security of the entire verification process.

Another scenario involves Bob continually sending probing packets to *Host D* in order to keep related flow rules active in switches, while Alice exploits these packets to camouflage communication between attackers. Implementing IAA can notably decrease the likelihood of triggering `Packet-In` messages. Since probing packets aid not only IAA but also various other attacks, RPV opts not to issue flow rules for probing packets upon their reporting to the controller. Consequently, the disguised packets produced by IAA will unavoidably be reported to the controller, ultimately exposing the attackers. RPV specifically targets four common types of probing packets: ICMP echo requests, TCP SYN packets, ICMP timestamp requests, and ARP requests. It's worth noting that ceasing the generation of flow rules for these probing packets doesn't impair network performance, as they are typically "single-use" in normal circumstances [57].

D. SWITCH-BASED LINK FABRICATION ATTACK (RELAYING LLDP)

It is assumed that the adversary has control of a switch, i.e., s_3 (see Fig. 8(f)). When the compromised switch receives the LLDP packet from switch s_1, instead of returning the packet to the controller, it sends the packet to the s_2. As s_2 has no awareness of the ongoing attack, send back the LLDP toward the controller, which results in adding a fake link between s_1 and s_2 [58].

Countermeasure: Stealthy Probing-Based Verification (SPV) defence [59] is implemented in the application layer to identify the fabricated links launched by the compromised host(s) or switch(es). Particularly, SPV utilizes the probing technique to verify the legitimacy of an inter-switch link. For this purpose, SPV monitors LDS activities to track new updates on the topology. Upon a new link is created in the network, SPV is notified by the controller to initiate the collection of some information about the newly added link, and statistic data and flows of associated switches. Based on collected data, SPV generates a probing packet. Specifically, some techniques are used to create the probe packet stealthily to prevent the adversary from distinguishing the probe packet from the normal host-generated traffic. Moreover, SPV utilizes a unique hash function value and a timestamp per probe packet to authenticate it and drop any forged ones. SPV also creates a flow to be installed on the source switch to direct the probe packet toward the destination switch. Upon sending the probe packet toward the source switch, SPV sets a timer and waits for the response. When the switch receives the probe packet, based on the installed flow, it forwards the probe toward the destination switch. In the normal scenario, the destination switch sends the probe packet toward the controller. However, in an attack scenario, the adversary which has control of the destination switch, forwards the probe packet toward the next switch and sends the prob toward the controller. Finally, when the controller receives the probe packet, it checks the source switch and destination switch ID to be matched with the newly added link information. In case of any mismatch, the link is considered a fabricated link.

Within the SPV mechanism, one of the notable challenges arises from the intricacy involved in crafting probe packets. These packets are meticulously designed to imitate normal host-generated traffic, employing specific techniques to conceal their identity from potential adversaries. Creating these packets requires precision and stealth to prevent any distinguishing factors that might reveal their probing nature, demanding considerable effort and technical sophistication to ensure successful deception. Another limitation of SPV lies in the complexity of key sharing required between administrators and the SPV component when encrypted communication TLS is employed for securing data plane traffic. If SPV were integrated directly into the SDN controller, this key-sharing requirement would become redundant.

E. TOPOLOGY FREEZING ATTACK

The adversary, initially, creates two fabricated links where the links originated from the same source switch and port while they have different destination switches and ports [20]. When the controller identifies the multi-link port, it removes the port and the associated links from the network topology graph. The unusual behaviour is that after this moment, the topology graph is not updated, even in the case of adding or removing new links. In order words, the adversary froze the network topology which significantly impacts topology-based applications such as routing and load balancing. No practical and detective solution has been proposed for this attack yet.

F. LINK LATENCY ATTACK

Soltani et al.'s proposed threat model [60] assumes that an attacker compromises one or more hosts. The attacker can use an out-of-band communication channel, present in a wired or wireless connection between two compromised hosts, to relay LLDP messages. This attack, known as the Link Latency Attack (LLA), involves the adversary creating a fake link between two switches by relaying LLDP messages through the out-of-band channel. The attacker leverages end hosts to inject unwanted traffic, such as ARP, to increase the packet processing time of the switches. As a result, the switches' response time to controller packets, like probe packets, is delayed. Exploiting this delay, the adversary can relay LLDP messages among switches and insert a fabricated link between them. This traffic manipulation negatively impacts network performance, leading to a detour in traffic and resulting in a lower quality of service (QoS) or quality of experience (QoE).

Countermeasure: Soltani et al. [60] propose a system known as Real-time Link Verification (RLV), which utilizes machine learning (ML) techniques for detecting Link Latency Attacks (LLA) and Link Fabrication Attacks (LFA) in SDN. The following provides a detailed overview of the RLV architecture, ML model configuration, and implementation process. The system's workflow unfolds as follows: The SDN controller generates Link Layer Discovery Protocol (LLDP) and probe packets at regular intervals. Subsequently, these packets are transmitted to the data plane switches. Upon receiving the LLDP packets, the switches respond to each one and send the responses back to the controller. The controller gathers these LLDP response packets, extracts the necessary metrics, and vectorizes them in batch format. Batching latency values from switches helps verify their validity along with associated delays through RLV, effectively reducing communication overhead. RLV assesses each vector based on the ML classification model and conveys its decision to the controller. Depending on the outcome, the controller either discards the LLDP packet or updates the topology database. Classification results and new LLDP data are stored in a dataset managed using InfluxDB. Human analysts play a vital role in monitoring the dataset's accuracy. RLV is deployed on a separate server with high processing capacity to minimize the time needed for link verification and model regeneration. However, this deployment raises security concerns, including spoofing and information leakage risks. To address these risks, a mutual authentication mechanism is implemented between the controller and the RLV server, verified by a Remote Access Dial-In User Service (RADIUS) server using a certificate-based approach. Additionally, the communication channel between the controller and RLV server is encrypted using SSL/TLS for enhanced security [60].

G. LINK FABRICATION ATTACK (FORGING LLDP)

When a compromised host, i.e., h_2 (see Fig. 8(e)) receives the LLDP packet from the connected switch s_2, it captures and manipulates the field of switch DPID and Port ID inside the packet. By sending the modified LLDP toward the switch s_2, the controller announces a non-existing inter-switch link between s_1 and s_2. The reason is that neither LLDP integration nor authentication is guaranteed during the OFDP process.

Countermeasure: TopoGuard [12] provides this assurance by adding a key-Hash Message Authentication Code (HMAC) [61] as a signed Type, Length, Value (TLV) inside each LLDP packet. TLVs are structured data elements used within the LLDP protocol to convey specific information. In this case, the TLV contains a HMAC is generated using a key, ensuring the integrity and authenticity of the transmitted data within the LLDP packet. The HMAC is computed using eq. (3).

$$HMAC(K, m) = h((K \oplus opad)|h((K \oplus ipad)|m)) \quad (3)$$

where m represents TLV fields in LLDP, i.e., switch DPID and PortID. Parameters of h and K indicate hash function and secret key, respectively. Sign | and $\oplus$ represent concatenation and XOR function. Also *opad* and *ipad* are considered constant values [61]. Particularly, TopoGuard utilizes a static secret key and SHA-256 hash function. Another approach is to select a random dynamic secret key, i.e., Ki, j, where i is the LLDP packet number and j indicates the topology discovery round. By using the dynamic secret key, each LLDP has a unique HMAC value. Hence, the adversary fails to use one instance of LLDP to create other fake LLDPs. Upon receiving the LLDP, the controller verifies the genuineness of the packet by checking the HMAC TLV.

The weakness of TopoGuard lies in its vulnerability to replay attacks due to the absence of a mechanism ensuring the freshness of its defence strategy. The cryptographic protection mechanism employed by TopoGuard doesn't incorporate a robust system to ensure the uniqueness or timeliness of the generated HMAC tags. As a result, when the system fails to generate new and unique cryptographic values for each transmission or session, it opens up the possibility for adversaries to execute replay attacks. Without this element of freshness, attackers can intercept previously valid HMAC tags and retransmit them, exploiting the system's inability to distinguish between the original and replayed data. This limitation compromises the effectiveness of TopoGuard in preventing adversaries from successfully sending forged LLDP packets, as replay attacks can bypass its defence mechanism. Moreover, the implementation of HMAC within each LLDP packet introduces computational overhead due to cryptographic operations, potentially impacting network performance. Moreover, managing keys, especially if they are dynamic, adds complexity to the system, increasing the chance of mismanagement or errors.

The approach introduced in [62] aimed to ensure integrity protection with freshness by implementing a strategy that involves updating the cryptographic key within each LLDP round. However, a critical constraint of this methodology is

its reliance on the controller to meticulously manage and monitor the keys utilized in every individual LLDP round.

H. REVERSE LOOP ATTACK

In OFDP, the controller utilizes the `Link-Type` field in the LLDP packet to identify if there is a reversed link between two switches, i.e., S_1 and S_2 (see Fig. 6(a)). To this end, the controller sets the field to $0x01$ value and sends it toward switch S_1. Upon receiving the LLDP from S_2, the controller set `Link-Type` to a $0x02$ and resend it back to switch S_2. In addition, if the latency value of the received LLDP changes, the controller updates the construction of the network topology graph which is a highly resource-consuming task.

The adversary leverages the mentioned two features to launch a reverse loop attack [20]. To this aim, when switch S_2 receives the LLDP packet with `Link-Type` to $0x02$, the adversary manipulates two fields of the packet. First, he changes the `Link-Type` value to $0x01$. Second, he modifies the timestamp field to slightly increase the LLDP latency value. Then, the adversary resends the packet to the controller. These malicious activities cause repeated LLDP send and receive between the controller and switch S_2, and also a significant workload on the controller. The risk of a reverse loop attack could be mitigated by ensuring the integrity of LLDP packets.

Countermeasure: Securing the integrity of LLDP packets is vital to thwart adversaries from transmitting counterfeit LLDP data. As highlighted in Section V-G, TopoGuard aimed to address this concern but lacked freshness in its approach, leaving room for potential replay attacks. To effectively counter such threats, the regular updating of cryptographic keys is imperative. The work in [62] proposed a solution aiming to preserve integrity with freshness by updating the cryptographic key in every LLDP round. However, this approach necessitates the controller to meticulously track the keys utilized in each LLDP round, constituting a notable limitation.

Addressing the shortcomings of prior methods, recent work [20] introduces an innovative approach. This method involves computing the HMAC tag over the DPID, port, and time-stamp, employing a singular cryptographic key rather than employing different keys in every LLDP round. Leveraging the existing time-stamp field within LLDP packets, this modification not only safeguards against adversaries tampering with the DPID or port to execute link fabrication attacks but also thwarts alterations to the time-stamp field within LLDP packets. This enhancement significantly fortifies the defence against potential attacks aimed at undermining LLDP packet integrity.

I. CLUSTER SPLITTING ATTACK

The LDS dynamically recalculates the network's structure when it detects changes in the topology. Then, the controller categorizes switches into clusters based on the presence of broadcast domain ports between them. These clusters can then merge into larger entities called archipelagos when connected by inter-cluster links. The LDS utilizes this clustering to divide the network into distinct parts, allowing it to avoid computing paths between switches located in separate parts, thereby saving storage and enhancing routing efficiency. By recognizing broadcast domain ports, the SDN controller identifies network segments where external links or multiple internal links are attached to a switch port. Unfortunately, the current controller lacks a mechanism to verify the legitimacy of a broadcast domain port. This gap poses a vulnerability as an adversary could fabricate an additional link for a switch port, creating a false broadcast domain port. Such manipulation could disrupt the subsequent steps in computing a topology instance. A potential vulnerability lies in the identification of clusters, making it susceptible to a cluster-splitting attack [63] during the initial step of computing the topology instance. This attack is designed to fragment switches belonging to a single cluster into multiple clusters. The adversary initiates the attack by creating a fake link for a specific switch port through the injection of LLDP packets. The controller, upon detecting multiple internal links, erroneously categorizes the port as a broadcast domain port. Consequently, during the re-computation of the topology instance, the LDS is misled into splitting the cluster. Furthermore, the LDS disregards the links associated with the counterfeit broadcast domain port. Eventually, the divided clusters amalgamate into different archipelagos due to the absence of inter-cluster links. This results in a substantial disruption of network communication, rendering hosts within the same physical cluster unable to communicate with each other.

Countermeasure: The countermeasure employed against Cluster Splitting Attack is the LldpChecker [63], a comprehensive solution designed to mitigate both Cluster Splitting Attack and another distinct threat, denoted as Cluster Amnesia Attack. To facilitate a structured discussion, the details of LldpChecker will be elucidated subsequent to the exploration of Cluster Amnesia Attack (see Section V-J), providing a comprehensive understanding of the strategic measures implemented to address the security vulnerabilities posed by Cluster Splitting Attack and Cluster Amnesia Attack.

J. CLUSTER AMNESIA ATTACK

To enhance the efficiency of route computation, the LDS employs the concept of archipelagos for network topology management. Clusters are segregated into distinct archipelagos if no inter-cluster links exist; otherwise, they coalesce into a single archipelago based on these links, typically external. The LDS systematically examines each external link, establishing or consolidating archipelagos depending on the islands connected by each link. However, the controller lacks a mechanism to authenticate external links, rendering it susceptible to adversarial manipulation. Adversaries could fabricate external links for switches, disrupting the archipelago merging process. Moreover, compromised

archipelago information adversely impacts route computation and the identification of broadcast domain ports [63].

Adversaries can exploit a vulnerability to execute a cluster amnesia attack when the LDS is in the process of identifying intra-links and archipelagos. This attack involves the LDS neglecting certain clusters during archipelago identification if there are no external links among them. The adversary initiates the attack by generating a fictitious external link within a cluster through the injection of a BDDP packet. Subsequently, the controller merges the cluster with itself into an archipelago and disregards clusters lacking external links. Typically, after the controller divides the cluster, there are no external links present. Consequently, the LDS fails to account for scenarios where the source and destination clusters of an archipelago are identical, leading to a breakdown in logic during the cluster merging process. The overlooked clusters remain unaffiliated with any archipelago, resulting in communication failure and topology update issues.

Countermeasure: The work in [63] introduced an effective and lightweight countermeasure for the cluster splitting attack and cluster amnesia attack, known as the LLDPCHECKER. This countermeasure operates in two distinct stages: LLDP verification and link validation.

Stage 1: LLDP verification: To thwart these attacks that hinge on counterfeit LLDP packets, a robust defence strategy involves authenticating LLDP packets. The controller generates a distinct LLDP packet for each active port on the switch. The Src MAC field in these LLDP packets is configured to match the MAC address of the respective switch port. Although the controller does not verify the MAC address during LLDP packet processing, adversaries exploit this vulnerability to craft a broadcast domain port and disrupt the network topology. However, without management authority, attackers cannot obtain MAC addresses of other legitimate switch ports. Additionally, the Src MAC address for each port is unique. Leveraging these insights, the integrity of LLDP packets can be ensured by validating the Src MAC field. It's worth noting that the OpenFlow switch in the Mininet simulation environment and certain hardware OpenFlow switches support this feature, consistent with prior research. To validate LLDP packet integrity, a mapping table is established, binding each switch port to its corresponding MAC address.

When the network service initiates, LLDPCHECKER creates an entry for the switch port and its associated MAC address in the mapping table. Upon the shutdown of a switch port, a `Port-Status` message is transmitted to the controller. Upon receipt, LLDPCHECKER removes the relevant entry from the mapping table. This approach enables LLDPCHECKER to dynamically maintain real-time information about the relationship between LLDP and switch ports. Building on this foundation, LLDPCHECKER extracts the datapath ID and port number from LLDP packets and retrieves the MAC address from the mapping table based on this information. Subsequently, LLDPCHECKER compares these values with the Src MAC field of the LLDP packets to validate consistency.

This process enables LLDPCHECKER to identify malicious LLDP packets and prevent the creation of counterfeit broadcast domain ports. During the link discovery process, OpenFlow switches encapsulate LLDP packets into `Packet-In` messages and transmit them to the controller. LLDPCHECKER in the controller processes these messages according to Algorithm 1. It initially extracts the remote switch port from the LLDP packets. Legitimacy is confirmed if the source MAC address retrieved from the PortMAC mapping table does not align with the MAC address contained in `Packet-In` messages. Otherwise, such messages are deemed illicit and prevented from entering subsequent applications.

Stage 2: link validation: Indeed, certain OpenFlow switches, share the same Src MAC address across all ports, creating a vulnerability where fake LLDP packets can bypass the initial defence mechanism. To address this, a more thorough validation stage is introduced before the link discovery service identifies these links. In this phase, LLDPCHECKER categorizes all ports into three groups based on the connected device type: SWITCH, HOST, and NULL, indicating connections to switches, hosts, and no devices, respectively. When the controller receives a `Packet-In` message containing an LLDP packet, LLDPCHECKER takes different actions based on the device type of the remote switch port. If the port is labelled as HOST, it discards the LLDP packet from the host since a port connected to a host cannot establish an external or internal link. For a remote switch port labelled as SWITCH, LLDPCHECKER retrieves the links associated with the port from the link library. It then verifies whether the newly discovered links align with the source and destination switch ports of these stored links. If not, the LLDP packet is flagged as illegitimate. LLDPCHECKER does not outright block LLDP packets on ports labelled as HOST, as research [13] demonstrates that prohibiting LLDP packets sent by a HOST-marked switch port doesn't thwart topology poisoning attacks. Attackers can reset the port's device type from HOST to NULL temporarily by disabling the network interface. However, the study finds that changes in attaching port status occur when network links are added or removed. In contrast, injecting or relaying LLDP packets enables adversaries to conduct a topology attack without altering port status. Therefore, link verification is performed through the remote switch port. If `Packet-In` messages with LLDP packets pass the first stage, the second phase of the inspection is initiated. The Port-Links mapping table stores the link ports of all switches and their corresponding links. LLDPCHECKER compares the source and destination switch ports of newly discovered links with these stored links. If inconsistencies are found, the message is flagged as illegal. This way, LLDPCHECKER can dynamically detect fake links in real-time.

Shrivastava and Kataoka [49] designed a new relay-based link fabrication attack, Multi-hop Link (MHL) fabrication in

order to inject a fake Multi-hop Link in the controller's topology view. In the presented Multi-hop Link fabrication attack, an adversary needs the concise exploration of packets to drop the LLDP messages while relaying the BDDP messages. In the second step, they illustrate the deployment of the proposed multi-hop link fabrication in order to elude basic behaviour-based defence methods. The adversary imitates the legacy switches and host traffic to impersonate the multi-hop link's actions in the represented attacking strategies over the basic MHL's fabrication attack. Then, they proposed a new defence system for both basic and extended MHL fabrication attacks. Shrivastava et al. proposed a defence and prevention platform named "Hybrid-Shield" which is presented with the Floodlight open-source controller. This proposed hybrid shield can be comfortably deployed to many different types of SDN controllers since it uses simple and fundamental SDN functionalities. And finally, they conducted experiments to evaluate and show the effectiveness of their proposed Hybrid-Shield. The results show that Hybrid-Shield provides quick detection and high accuracy of MHL verification [49].

Yuan et al. [64] introduced a method for formally verifying potential attacks in SDN controllers automatically. They delve into the commonalities among SDN controllers, thoroughly examining the OpenFlow protocol, mainstream controller implementations, and vulnerabilities within SDN systems. Leveraging this understanding, they abstract core components, their functions, and critical communications within SDN systems. Additionally, they identified key properties that, if compromised, could lead to catastrophic outcomes, which are often exploited by attackers. The authors then construct a formal description of SDN systems, delineating the behaviours of various entities within the system, including malicious and benign components, and their interactions. This description serves as the system model for analyzing potential attacks against SDN controllers through formal verification using model-checking techniques. Subsequently, they assessed counterexamples generated during the analysis, filtering out those with practical exploit potential. Finally, they validated the feasibility of these attack paths by applying them to real-world SDN systems.

VI. IDENTIFIER BINDING ATTACK AND SECURITY COUNTERMEASURES

Several attacks have been designed against identifier bindings in SDN which allows the adversary to overtake the victim's identifiers, called Identifier Binding Attack (IBA). Three critical attacks, namely Host Location Hijacking Attack, Port Probing Attack, and Persona Hijacking Attack break bindings between identifiers in different layers and assign them to the adversary host. In this way, the adversary deceives the controller into believing that the malicious host is the legitimate owner of the victims' identifiers. This section provides a comprehensive analysis of diverse IBA, explaining their operational mechanisms and corresponding mitigation strategies.

A. HOST LOCATION HIJACKING ATTACK

This attack allows the adversary to take over the MAC address from the victim host which leads to hijacking its location information [12]. To this aim, the adversary first finds the victim host's MAC address and assigns it to its Ethernet source address. The malicious host then sends spoofed data plane packets (e.g., Internet Control Message Protocol (ICMP), Hypertext Transfer Protocol (HTTP), Domain Name System (DNS), etc.) to trigger `PACKET-IN` messages toward the controller. As described in Section IV, HTS continuously monitors `PACKET-IN` message to track host mobility and updates the host profile based on host motions. Upon receiving the packet, HTS indicates a mismatch between extracted information (MAC address, switch DPID and port number) from the `PACKET-IN` messages and the corresponding entry of the host profile. The HTS supposes that the victim has migrated to the new location and updates the new information inside the host profile which is the location of the adversary. By launching the attack, all traffic for the victim host is hijacked toward the adversary host. The attack lasts until the victim initiates any traffic toward the controller which HTS can correct the host profile entry to the victim location. To persist the attack over a large timescale, the adversary could launch it on server targets which typically work in passive mode (i.e., listen to the received traffic on a specific port).

Countermeasure: The main reason behind the attack is that the host identifiers are not authenticated and verified. The first approach to mitigate the risk of the attack is to apply cryptographic methods and utilize public and private keys to authenticate the host in a migration event. However, the technique could impose considerable computation overhead for `PACKET-IN` message processing and require host implementations. Another approach is presented in the TopoGuard [12] in which HTS checks the legitimacy of host migration by verifying the following two conditions; (1) during the migration process, a `Port-Down` message must be received from the source port (a precondition of the host migration), (2) host reachability test must be failed to the source port (postcondition of the host migration) after finishing the migration.

In defence against this attack, Sphinx constructs a flow graph that preserves IP/MAC associations for all hosts. In addition, it stores a list of permitted switch ports that could be assigned to hosts. Upon receiving an ARP spoofed message, it validates against security policy rules and mentions authorized binding. Any violation is raised as a security alarm.

B. PORT PROBING ATTACK

The main weakness of the TopoGuard arises when the victim host is in transit status between source and destination switches. The adversary takes advantage of this weakness to mount a port probing attack. In this attack, the adversary periodically monitors and probes the victim port's liveness by using port probing tools such as ICMP, Transmission

Control Protocol (TCP), and APR. Upon detecting that the victim host is offline or starting the migration process, HTS receives a `Port-Down` message from the source port which satisfies the precondition of host migration. In addition, no location address is bound to the victim host which meets the requirement of postcondition. At this moment when the migration process is still not finished by the victim host, a malicious host could complete the migration process by spoofing the victim's identifier and sending a `PACKET-IN` message to the controller.

Countermeasure: Port probing exploits a critical vulnerability tied to host migration—a race condition where the first claiming end-host is recognized as the target by the controller. This attack exploits the absence of authentication around network identifiers like MAC and IP Addresses, capitalizing on the ability to spoof these identifiers and their associations. Conventional network access controls, exemplified by IEEE 802.1x [44], rely on certificates or cryptographic credentials to validate device authorization before allowing traffic through the network port. Unfortunately, 802.1x lacks the cryptographic binding of network identifiers (e.g., MAC address) to user credentials, making it inadequate in preventing port probing attacks.

In response to this limitation, Secure Binder [65] extends the protective scope of 802.1x across the entire identifier hierarchy. This comprehensive approach effectively mitigates port probing attacks by disallowing attackers from falsely assuming the victim device's identity without triggering alerts. Further insights into Secure Binder will be detailed in Section VI-D.

C. PERSONA HIJACKING ATTACK

In this attack, the adversary breaks two identifier bindings, including MAC Address to Network Location, and also IP Address to MAC address. It is supposed that a Dynamic Host Configuration Protocol (DHCP) server has been configured to assign and release IP addresses to/from the network node. The attack operates in two phases. In the first phase, which is referred to as IP takeover, the adversary leverages the DHCP server to take over the IP address and hostname of the victim. To this end, the adversary initiates a fake `DHCP-RELEASE` message and requests for releasing the victim's IP address. This breaks the MAC-IP binding of the victim host. Then, the adversary requests DHCP for an IP address. To bind the released IP address (i.e., victim's IP address) to the MAC address of the malicious host, the adversary launches a flooding attack against the DHCP server. To this end, the adversary sends a huge number of `DHCP-DISCOVER` messages to request a new IP address until the server offers the IP address of the victim.

The second phase is addressed as the Flow Poisoning Attack. Before re-assigning the released IP address to the adversary, the DHCP server initiates an ARP probing message to verify that the IP address is not currently used by other nodes. Flow Poisoning Attack aims to blackhole the victim's claim in response to the ARP request. First, the adversary breaks the binding between the MAC address and network location in the DHCP server. To this aim, the adversary sends a spoofed message from the DHCP server toward the victim. Due to flow table miss, the `PACKET-IN` message is directed toward the controller, leading to updating the location of the DHCP server in HTS and adding a new flow rule on the corresponding switch. As a result, all traffic toward the DHCP is directed toward the adversary location.

Hence, when the DHCP server sends ARP probing to verify the IP availability, the controller discovers the location information of the legitimate DHCP server and applies a new flow rule on the switch. The critical point is that due to a delay in rule consistency in the switch and the controller, the old flow rule is not removed from the switch. Hence, the ARP reply from the victim is matched with the old rule and consequently is directed toward the fake DHCP which is the adversary location. As a result, because the real DHCP server receives no claim for the released IP address, it allocates the victim's IP address to the adversary. The attack age depends on the frequency of flow rule updating. In an extended case, it will last till the victim's DHCP lease expires.

Countermeasure: Several network weaknesses cause personal hijacking attacks. First, some protocols, such as ARP, use an insecure broadcast domain and send its request with no authentication. This empowers the adversary to listen to the broadcast traffic and even respond to the request. Second, identifier bindings are changed without consistency checking and also considering their implications on other services. Secure Binder [65] uses SDN functionality in separating data and the control layer to provide a defense against port probing and personal hijacking attacks.

To protect the IP address to MAC address binding, Secure Binder sends all identifier-binding broadcast traffic toward the controller, preventing the adversary from listening and responding to a broadcast request. To this aim, Secure Binder designs a binding mediator module in the controller to separate the identifier binding control traffic from normal data-plane traffic. Once a switch receives the explicit identifier binding traffic (such as ARP, DHCP, and 802.1x), sends it to the controller. The mediator validates receiving binding messages by using a binding store database where all authenticated bindings in all layers are stored. If receiving traffic aims to change an existing binding, before updating the database, the mediator checks the old binding to make sure that it is no longer reachable.

For protecting MAC address to network location binding, i.e., switch DPID and port, Secure Binder develops a port control module in the controller to perform dynamic filtering on source address. The module aims to prevent the adversary from sending any spoofed messages toward the controller. To this end, the module leverages the flow-table pipeline feature to reserve the first table, i.e., Table 0, to install egress filtering rules, separating the binding traffic from the normal. Additionally, network ports are categorized into four states, namely, Unknown, Edge, Internal, and Quarantined. The initial value of a port is Unknown. The

port state will turn to Internal if it receives any LLDP message. In this case, incoming traffic will go to Table 1 for forwarding. The port is set to Edge when it receives host-generated traffic. In Edge port, based on the rule installed in Table 0, packets with validating source addresses are sent to Table 1 for forwarding, and other ones are sent to the controller to log and drop. Any abnormal behaviour changes the status of a port to Quarantined. Secure Binder also leverages IEEE 802.1x [44] protocol to guarantee that a unique mapping exists between a host and the corresponding MAC address. IEEE 802.1x is a port-based access control standard that initiates a cryptography authentication check when a host joins the network. Secure Binder extends the IEEE 802.1x protocol to validate a host's MAC address in the authentication process. To this end, it provides a database located in the RADIUS authentication center which contains records of mapping between the host's certificate (or password) and MAC address. If the incoming MAC address and certificate match with existing mapping in the database, the host is successfully authenticated. Then, Secure Binder binds the MAC address to the associated port and inserts a flow rule on the switch to allow the traffic with the MAC address to be forwarded to the table with a higher ID.

Secure Binder faces potential overhead and latency issues due to routing all identifier-binding broadcast traffic to the controller. This approach could significantly increase network latency and create additional overhead, particularly problematic in expansive networks. The heightened traffic load imposed on the controller might strain its processing capabilities, potentially impeding the overall network performance, especially in large-scale environments. Furthermore, while Secure Binder robustly combats external threats, its limitations in countering insider attacks—where authenticated users exploit their access privileges maliciously—pose a concern. Binder's vulnerability to insider attacks underscores the need for a zero-trust mindset, where all users and devices—even those already authenticated—should be continually scrutinized for potential threats or misuse of privileges.

D. ARP SPOOFING ATTACK

ARP spoofing is a critical tactic utilized by cyber adversaries to launch DoS or MITM attacks, severely compromising system performance and data integrity. Despite efforts to thwart ARP spoofing using SDN-based intrusion detection systems (IDSs), many existing approaches are ineffective due to static thresholds and the absence of real-time information during detection. To address these issues, a new deception-based IDS is proposed in [67] to efficiently identify attackers in SDN networks. This method tricks attackers into gathering real-time data, enhancing detection capabilities. Simulation results in the Mininet simulator demonstrate that the proposed approach significantly outperforms existing methods in mitigating ARP spoofing attacks. They suggest a fresh approach to IDS that revolves around deception. This technique tricks attackers into revealing real-time data such as attack frequency and timing, thereby enhancing the detection system. Their key contributions include introducing a novel dynamic detection threshold derived from data gathered from decoys and presenting an IDS that incorporates deception and adapts to fluctuations in attack frequency during detection.

In the following, we'll explore ARP spoofing attack scenarios. The attacker begins by conducting network reconnaissance before executing ARP spoofing attacks. Network reconnaissance involves gathering information, such as IP and MAC addresses, services, etc., about a system using tools like Nmap and Xprobe2. This gathered data helps the attacker identify potential targets. Following reconnaissance, the attacker proceeds to launch ARP spoofing attacks to corrupt victims' ARP caches. In this scenario, adversary X focuses on host T and corrupts its ARP cache by pretending to be host Y. As a result, any traffic from T that was meant for Y is rerouted to X. A host is categorized as a non-attacker if it's recognized by a single IP-MAC pair within the network.

Here are the attacker's traits: 1. An attacker can send an ARP packet to acquire multiple IP addresses for a single MAC address or multiple MAC addresses for a single IP address within the network. 2. The MAC address in the header of an ARP packet sent by an attacker typically differs from the MAC address at the data link layer. 3. Attackers often send out more ARP requests than the number of ARP responses they receive.

Countermeasure: Motivated by the challenges posed by ARP spoofing and the necessity for reliable solutions, the study [68] presents a thorough examination in their paper, offering innovative strategies for safeguarding networks within the SDN framework. Their research makes the following significant contributions. They introduce a novel and efficient ARP spoofing detection and mitigation scheme based on deep learning, enhancing the network's ability to detect and thwart such attacks effectively. Their system displays improved network throughput and CPU utilization, even amidst ARP spoofing attacks, while also identifying potentially risky situations through detailed packet analysis. The proposed system undergoes thorough assessment across various network sizes and traffic conditions, offering valuable insights into its effectiveness and scalability.

The fundamental approach includes four key components: First, their system gathers data on network traffic from across the entire SDN network, including information on network devices and their connections. This holistic network perspective forms the basis for identifying any unusual or suspicious behaviour and potential avenues of attack. To effectively detect and counter ARP spoofing attacks, they propose a state-of-the-art detection system that utilizes advanced methods, including a DNN (Deep Neural Network) model. This model is trained to recognize patterns indicative of ARP spoofing, enabling real-time monitoring of MAC addresses associated with each network device. Additionally, the system cross-references ARP responses with a known

TABLE 3. Summary of LFA and IBA attacks, description, attack root cause.

Ref	Attack	Description	Root Cause	Cause	Exploited Module
[12]	Host-based Link Fabrication (Relaying LLDP)	The adversary relays LLDP packet between two switches using compromised host(s) to create a fabricated link.	Relaying LLDP	Participation of host(s) in LLDP propagation process	LDS & OFDP
[13]	Port Amnesia	The adversary resets a host-connected port to change its classification	Relaying LLDP	Dynamic changes in port usage	Port classification
[57]	Invisible Assailant Attack	Injects fake links into networks, hiding attack traffic within regular data flow, evading detection through planned phases, exploiting link discovery service vulnerabilities.	Relaying LLDP	Failure to verify LLDP route paths.	LDS & OFDP
[59]	Switch-based Link Fabrication (Relaying LLDP)	The adversary relays LLDP packet between two switches using compromised switch(s) to create a fabricated link.	Relaying LLDP	Lack of LLDP propagation path verification	LDS & OFDP
[20]	Topology Freezing	The adversary imposes run-time error in topology update process by adding two fabricated links	Relaying LLDP	Weak communication of topology update and routing modules	Topology update
[12]	Link Fabrication (Forging LLDP)	The adversary sends a fake LLDP toward a switch to create a fabricated link.	Forging LLDP	Lack of verification on the integrity of LLDP	LDS & OFDP
[20]	Reverse Loop	The adversary manipulates *Link-Type* field in LLDP to exhaust the controller's resources in an infinite loop	Forging LLDP	Lack of LLDP authentication and integration	LDS & OFDP
[66]	Link Latency	The adversary sends huge ARP traffic toward switches to increase the probe packet processing time and keep the relayed LLDP latency in the valid range.	Relaying LLDP	Ineffective outlier detection method	Latency calculation
[63]	Cluster splitting attack	The attack aims to fragment a single cluster of switches into multiple clusters	Forging LLDP	Lack of mechanism to ensure the legitimacy of a broadcast domain port	LDS & OFDP
[63]	Cluster amnesia attack	The attacker creates a false external link for a switch, disrupting the merging process of archipelagos.	Forging BDDP	Lack of mechanism to validate external links	LDS & OFDP
[12]	Host Location Hijacking	The adversary takes over the MAC address from the victim host and assign it to the malicious host	MAC/IP Spoofing	Insecure broadcast domain	HTS
[13]	Port Probing	Upon disconnecting the victim host from the network, the adversary completes the host migration event.	MAC/IP Spoofing	Lack of host authentication in migration event	HTS
[65]	Persona Hijacking	The adversary creates a fake DHCP server and poison the flow rule to hijack the victim host traffic	Binding Breaking	Flow rule inconsistency in the switch and the controller	HTS

database of MAC addresses and IP pairings to spot any inconsistencies. When an ARP spoofing attack is detected, the system can respond immediately by isolating the offending device or notifying network administrators. Furthermore, their system promptly alerts network administrators to detect threats, empowering them to make swift and well-informed decisions. It conducts thorough traffic analysis, distinguishing between normal network activity and potentially malicious behaviour. Importantly, all network traffic data, whether from attack incidents or normal operations, is stored in CSV format, facilitating in-depth analysis and assisting in the development of future threat mitigation strategies.

VII. EXPLORING SECURITY COUNTERMEASURES AND TRADE-OFFS

This section offers a detailed exploration of security countermeasures and trade-offs relevant to security countermeasures against LFA and IBA. We commence with Section VII-A, delving into the intricacies of Performance Evaluation, and assessing the efficacy of the security measures. Subsequently, in Section VII-B, we navigate the complexities of balancing security requirements with operational efficiency in the context of secure topology discovery for SDN environments.

Table 3 presents the summary of existing LFA and IBA attacks and attack root cause analysis. This table is pivotal as it provides a comprehensive overview, enabling a quick and structured understanding of different attack scenarios and their underlying causes within the context of the Topology Poisoning Attack. It's crucial to note that all LFA and IBA scenarios operate under the assumption that the adversary has compromised at least one host or switch. However, the specific location of the attacker can significantly influence the likelihood of a successful attack. In instances where the assumption is that the attacker has compromised a host and is situated within that host, the risk factor increases. This is because even a relatively weak attacker with limited capabilities could initiate the attack, thereby elevating the overall likelihood of successful exploitation. It's important to recognize that this scenario raises the probability of an attack occurrence. On the other hand, when the assumption is that the attacker has compromised a switch, the feasibility of the attack becomes more constrained. Initiating such an attack from a compromised switch would typically require a more sophisticated and potent adversary. Consequently, this implies a lower likelihood of occurrence compared to scenarios involving compromised hosts. However, despite the lower likelihood, the impact and severity of the attack could be equally significant.

In examining the landscape of threats posed by LFA and IBA, it's essential to comprehensively understand each attack type and its associated countermeasures. Table 4 provides an exhaustive analysis, offering a detailed breakdown of various attacks, corresponding countermeasures, and their respective advantages and disadvantages. This comprehensive comparison facilitates a structured comprehension of the strengths and limitations of each countermeasure in mitigating LFA and IBA attacks within network infrastructures.

A. PERFORMANCE EVALUATION

The performance evaluation of main countermeasures for LFA, outlined in Table 5, highlights varying effectiveness, scalability, and compatibility across different defence methods.

1) EFFECTIVENESS

The effectiveness of the countermeasures is primarily demonstrated by simulating various network topology

TABLE 4. Security comparison on presented countermeasures against topology poisoning attack.

Attack	Countermeasure	Advantage	Disadvantage	Ref.
Host-based Link Fabrication (Relaying LLDP)	• TopoGuard. Port classification technique based on first-seen traffic	✓ Restrict hosts in LLDP propagation	× Ineffective in a dynamic environment × Vulnerable to Port Amnesia attack	[12]
	• Comparing the statistical distribution of link latency values against a baseline threshold	✓ Statistical analysis	× High false alarm in peak hours × Relies on single threshold	[53]
Port Amnesia	• TopoGuard+. Comparing link latency interval with a threshold (LLI) • TopoGuard+. Monitoring the frequency of port down and up during LLDP propagation (CMM).	✓ Cover both in-band and out-of-band attacks	× High false alarm in peak hours × Relies on a single threshold × Overhead of port monitoring	[13]
Link Latency	• **MLLG**. Classifying LLDP latency value using machine learning techniques	✓ High attack detection rate ✓ Dynamic latency threshold	× Overhead of training model × Offline evaluation × Lack of scalability for large SDN	[66]
	• **RLV**.Scaled and real-time classification on link latency values using weighted supervised ML	✓ High performance in large scaled SDN ✓ Robustness	× Overhead of learning model × Retraining model for different latencies	[60]
Switch-based Link Fabrication (Relaying LLDP)	• **SPV**. Verifying link legitimacy by generating and sending a probe packet per new link toward the source switch	✓ Detect both known and unknown LFA	× Communication overhead × Key Sharing Complexity	[59]
Topology Freezing	• Checking the availability of links before sending real traffic.	✓ Intuitive countermeasure	× No attack detection system	[20]
Link Fabrication (forging LLDP)	• TopoGuard. HMAC over DPID and Port with a static key • TopoGuard. HMAC over DPID and Port with a dynamic secret key	✓ Verify LLDP integration ✓ Verify LLDP authorization	× Overhead of HMAC × Overhead of LLDP verification × Vulnerability to replay attacks	[12]
	• LLDP integrity protection with updated freshness in every LLDP round	✓ Prevent replay attack	× Key tracking complexity	[62]
Reverse Loop	• HMAC over DPID, Port and time stamp using a single key	✓ Eliminates key tracking burden	× Reliance on LLDP timestamp	[20]
Cluster splitting attack	• LLDPChecker: LLDP verification module maintains a real-time mapping table to detect false broadcast domain ports.	✓ Realtime detection of false broadcast domain port	× Overhead of LLDP verification	[63]
Cluster amnesia attack	• LLDPChecker: Link validation module categorizes ports by device type and verifies links	✓ Efficient port categorization ✓ Detection of irregular LLDP	× Overhead of link validation	[63]
Host Location Hijacking	• Applying cryptographic methods for host authentication • TopoGuard. Verifying pre & post-condition of host migration	✓ Protect the legitimacy of host migration	× Computation overhead in packet-in processing × Require implementation on host(s)	[12]
Persona Hijacking and Port Probing	• **SecureBinder.** Separating identifier binding traffic from normal traffic (Mediator module) • **SecureBinder.** Port classification (egress filtering) and extending IEEE 802.1x	✓ Protect the identifier binding in all layers ✓ Enforce access control	× Potential Overhead and Latency × Limited protection against insider attacks	[65]

TABLE 5. Performance evaluation for main countermeasures for LFA.

Countermeasure	Effectiveness	Performance Overhead	Scalability	Compatibility
TopoGuard [12] against LFA (Relaying LLDP)	(+) Detects violation of Device Type of particular ports. (-) Bypassed by Port Amnesia Attack	Port Manager processing time for LLDP packets (avg. 0.02 ms) and host-generated packets (avg. 0.032 ms).	Limited scalability for network size due to controller overhead for LLDP processing.	Primarily programmed for Floodlight; adaptation for other controllers requires additional programming.
TopoGuard+ [13] against Port Amnesia	(+) Detects all fake links through comparing measured link latencies and computed threshold (-) Bypassed by Link Latency Attack	LLDP construction through the addition of timestamp TLV (avg. 0.134 ms) & LLDP processing for link latencies inspection (avg. 0.299 ms)	Limited scalability for network size due to controller overhead for LLDP construction and LLDP processing	Primarily programmed for Floodlight; adaptation for other controllers requires additional programming.
MLLG [66]	(+) Detects all fake links using trained ML model (Supervised learning)	LLDP construction through the addition of timestamp TLV & ML model processing on LLDP	Limited scalability for network size due to high false alarm	ML-based approach compatible with all SDN controllers.
RLV [60]	(+) Detects all fake links using trained ML model (weighted supervised learning)	LLDP construction through the addition of timestamp TLV & ML model processing on LLDP	Efficient scalability across various network sizes	ML-based approach compatible with all SDN controllers.
LLDPChecker [63]	(+) Detects all fake links through two stages of LLDP and Link validation	LLDP validation (avg. 0.0004 ms) & link validation the second stage (avg. 0.005 ms)	Efficient scalability across various network sizes and topologies	Primarily programmed for Floodlight; adaptation for other controllers requires additional programming.
SPV [59]	(+) Detects all fake links through probe messages	Overhead includes generating probes, installing new flows, and link verification process.	Efficient scalability across various network sizes and topologies	Compatible with all SDN controllers
TopoGuard [12] against LFA (Forging LLDP)	(+) Detect all fake link using HMAC mechanism.	Overhead in HMAC computation. Average overhead is 0.431ms (80.4% of overall LLDP construction time).	Limited scalability for network size due to controller overhead for LLDP processing.	Primarily programmed for Floodlight; adaptation for other controllers requires additional programming.

poisoning attacks in the environment, such as the Floodlight controller. The reactions of the SDN controller equipped with the countermeasure are then observed by monitoring its console output and security logs. This method allows for a thorough assessment of how well the countermeasures detect and respond to the

simulated attacks, providing valuable insights into their efficacy.

For each countermeasure, the effectiveness is evaluated based on its ability to detect and mitigate fake links introduced by various attacks. TopoGuard, for instance, positively detects violations of a device type of particular ports, showcasing its effectiveness in identifying abnormal link behaviour. However, it is also noted that TopoGuard can be bypassed by the Port Amnesia Attack, indicating a limitation in its effectiveness against certain types of attacks. Similarly, TopoGuard+ demonstrates effectiveness by detecting all fake links through a comparison of measured link latencies and computed thresholds. However, it is vulnerable to being bypassed by the Link Latency Attack, which affects its overall effectiveness. MLLG and RLV both leverage machine learning models to detect fake links effectively, showcasing their robustness in identifying abnormalities in network behaviour. On the other hand, LldpChecker exhibits effectiveness by detecting all fake links through two stages of LLDP and Link validation, demonstrating a comprehensive approach to link verification.

2) PERFORMANCE OVERHEAD

The performance overhead across most countermeasures against LFA (Relay LLDP) primarily arises from two main factors. Firstly, the extension of the LLDP protocol with a timestamp TLV introduces additional delays in the construction of LLDP packets. Secondly, there are additional security inspections conducted on LLDP packets, contributing further to the overhead. Unlike TOPOGUARD+, MLLG, and RLV, the LLDPCHECKER does not extend the LLDP protocol, resulting in no extra delay in LLDP packet construction. Consequently, it exhibits less performance overhead compared to the aforementioned countermeasures. The noteworthy aspect is that while each of these countermeasures adds overhead to the SDN controller, it remains negligible, with no impact on data plane flows. Additionally, the overhead of LLDP packet construction with the implementation strategy of computing the HMAC value is considerable, accounting for 80.4% of the overall LLDP construction time [12].

In addition, other countermeasures like SPV, which utilize a probe message mechanism, incur additional overhead in generating probes and installing new flows on switches. To alleviate this, SPV employs a multi-threading approach. In single-threading mode, the verification time for the largest dataset with 96 data-plane links is 26.1 seconds. However, leveraging the multi-threading mode significantly improves the verification time, reducing it to 10.6 seconds.

3) SCALABILITY

In the context of network security countermeasures, scalability is crucial for ensuring that the protection mechanisms can effectively adapt to larger networks or increased traffic without significant degradation in performance or efficiency. A scalable countermeasure should be able to accommodate the growing complexity and size of networks while remaining effective in detecting and mitigating security threats.

For each countermeasure mentioned, scalability manifests differently based on various factors such as processing overhead, false alarm rates, and compatibility. Countermeasures like RLV and LLDPCHECKER demonstrate efficient scalability across different network sizes due to their optimized processing methods and ability to adapt to various topologies. On the other hand, countermeasures like TOPOGUARD and TOPOGUARD+ show limited scalability due to their reliance on controller overhead for LLDP processing and construction, which can become a bottleneck as network size increases. Compatibility also plays a role, with ML-based approaches like MLLG and RLV being compatible with all SDN controllers, offering scalability without being tied to specific platforms. Overall, a balance between effectiveness, performance overhead, and adaptability is essential for ensuring scalability in network security countermeasures.

4) COMPATIBILITY

Compatibility refers to the ability of each countermeasure to seamlessly integrate with various SDN controllers without requiring significant modifications or additional programming efforts. Countermeasures that exhibit high compatibility can be deployed across different controller platforms with ease, enhancing their versatility and applicability in diverse network environments. For example, countermeasures like MLLG, RLV, and SPV are designed to be compatible with all SDN controllers, allowing them to be deployed without compatibility concerns or platform limitations. This compatibility ensures that these countermeasures can be readily integrated into existing network infrastructures, regardless of the specific SDN controller being used, thereby maximizing their utility and effectiveness in safeguarding against LFA attacks.

Conversely, countermeasures with limited compatibility, such as TOPOGUARD and TOPOGUARD+, are primarily programmed for specific SDN controllers like Floodlight. While effective within their designated environments, these countermeasures may require additional programming or adaptation efforts to function seamlessly with other SDN controller platforms. This lack of compatibility can introduce challenges in deploying these countermeasures in heterogeneous network environments where multiple SDN controllers are utilized. Thus, compatibility considerations play a crucial role in determining the ease of deployment and interoperability of network security countermeasures, ultimately influencing their effectiveness in mitigating LFA attacks and other security threats in SDN deployments.

B. NAVIGATING TRADE-OFFS IN SECURE TOPOLOGY DISCOVERY FOR SDN ENVIRONMENTS

Each countermeasure offers unique advantages and disadvantages in mitigating topology poisoning attacks. Careful consideration of these factors is crucial when selecting and implementing countermeasures to ensure comprehensive

protection for SDN environments. In SDN ensuring the security of topology discovery techniques involves navigating a complex landscape of trade-offs and challenges.

TopoGuard, employing a port classification technique based on first-seen traffic, effectively restricts hosts in LLDP propagation, which is advantageous for maintaining network integrity. However, its effectiveness wanes in dynamic environments due to its inability to adapt quickly to changing network configurations. Additionally, vulnerability to Port Amnesia attacks introduces a significant trade-off as it may lead to false positives or negatives, compromising detection accuracy and reliability. On the other hand, statistical analysis, as employed by another countermeasure, offers a more analytical approach but suffers from a high false alarm rate during peak hours and reliance on a single threshold, highlighting a trade-off between precision and performance.

Countermeasures like TopoGuard+ cover both in-band and out-of-band attacks, providing comprehensive protection. However, this broader coverage comes at the expense of increased false alarms during peak hours and overhead associated with port monitoring, indicating a trade-off between coverage and operational efficiency. The reliance on a single threshold further exacerbates this trade-off, as it may not adequately account for variations in network conditions.

MLLG's machine learning-based approach offers a high attack detection rate and dynamic latency threshold, enhancing adaptability and accuracy in detecting topology poisoning attempts. However, the overhead involved in training the model and offline evaluation introduces scalability challenges, posing a trade-off between detection accuracy and resource consumption. Similarly, RLV's performance and robustness in large-scale SDNs are commendable but come at the cost of increased complexity due to model learning and retraining requirements, highlighting a trade-off between performance and computational overhead. MLLG is suitable for environments with moderate to high traffic variability and dynamic network conditions, such as large enterprise networks. While RLV is ideal for large-scale SDN environments with complex network architectures, such as cloud service provider networks or telecommunications networks. For example, a cloud service provider managing a vast infrastructure of virtualized resources could leverage RLV to continuously monitor link latency and ensure the reliability and performance of their services.

SPV's method of verifying link legitimacy provides comprehensive security coverage by detecting both known and unknown LFAs. However, the communication overhead and complexity in key sharing introduce trade-offs in terms of network performance and management overhead, balancing security with operational efficiency. SPV is well-suited for environments where network integrity and security are paramount, such as critical infrastructure networks.

Countermeasures like HMAC over DPID and Port with either a static or dynamic secret key aim to verify LLDP integration and authorization. However, they suffer from overhead in HMAC and LLDP verification, as well as vulnerability to replay attacks. Additionally, LLDP integrity protection with updated freshness in every LLDP round helps prevent replay attacks but introduces key tracking complexity. Countermeasures like LLDPChecker aim to detect false broadcast domain ports and irregular LLDP, respectively. While they offer real-time detection, they come with the overhead of LLDP verification. The LLDPChecker is suitable for environments where real-time detection of malicious activities is critical, such as data centres or cloud computing environments. For example, a data centre hosting critical applications and services may deploy cluster splitting and cluster amnesia attack detection mechanisms to promptly identify and mitigate any anomalies in the network.

Cryptographic methods for host authentication and verification of pre and post-conditions of host migration help protect the legitimacy of host migration. However, they may introduce computation overhead in packet-in processing and require implementation on hosts. The solution is suited for environments where securing host migrations and ensuring the integrity of host authentication are paramount such as Cloud Computing Infrastructure and Data Centers.

SecureBinder separates identifier-binding traffic from normal traffic and enforces access control. However, it may introduce potential overhead and latency and offer limited protection against insider attacks. SecureBinder is well-suited for environments where strict access control and data confidentiality are essential, such as corporate networks.

One fundamental trade-off arises between scalability and effectiveness. While robust security measures are essential for safeguarding against topology-based attacks, such measures often introduce additional computational and communication overhead, potentially impacting the scalability of the network. Balancing the need for enhanced security with the imperative of efficient network operation is therefore crucial, especially in large-scale SDN environments where the sheer volume of network traffic can pose significant challenges. Countermeasures like TopoGuard, which employs a port classification technique based on first-seen traffic, effectively restrict hosts in LLDP propagation, maintaining network integrity. However, its effectiveness wanes in dynamic environments due to its inability to adapt quickly to changing network configurations.

Moreover, the trade-off between overhead and detection accuracy presents a central challenge. Enhancing the accuracy of topology discovery techniques typically requires allocating more resources, leading to increased overhead. This dilemma underscores the need to optimize the balance between detection accuracy and resource consumption to ensure that security measures do not unduly burden network performance. Additionally, in dynamic environments characterized by frequent topology changes, maintaining stability while accommodating adaptability poses a significant challenge. Security solutions must be agile enough to dynamically adjust to evolving network conditions without sacrificing stability or reliability. Countermeasures like MLLG's machine learning-based approach offer a

high attack detection rate and dynamic latency threshold, enhancing adaptability and accuracy in detecting topology poisoning attempts. However, the overhead involved in training the model and offline evaluation introduces scalability challenges, posing a trade-off between detection accuracy and resource consumption.

Furthermore, the complexity of security measures often presents a trade-off with manageability. Introducing sophisticated security mechanisms can increase the complexity of network management and configuration, potentially overwhelming network administrators. Simplifying the management of security measures is therefore essential to ensure that they remain manageable and maintainable, even as the network grows in complexity. Additionally, allocating resources for security purposes may divert them from other critical network functions, creating a trade-off between resource consumption and security assurance. Striking the right balance between enhancing security assurance and preserving network efficiency is thus a key challenge in SDN environments. Countermeasures like SPV's method of verifying link legitimacy provide comprehensive security coverage by detecting both known and unknown LFAs. However, the communication overhead and complexity in key sharing introduce trade-offs in terms of network performance and management overhead, balancing security with operational efficiency.

Lastly, the pursuit of adaptability in security measures must be carefully managed to avoid inadvertently exposing the network to new vulnerabilities. Implementing adaptive security measures may introduce new attack vectors or weaknesses if not properly monitored and updated. Therefore, a continuous cycle of monitoring, evaluation, and adaptation is essential to address emerging threats and vulnerabilities effectively. By addressing these trade-offs and challenges through a holistic approach to secure topology discovery, SDN environments can achieve a high level of protection against topology-based attacks while maintaining efficient network operation.

VIII. TOPOLOGY DISCOVERY PROTOCOL SECURITY ENHANCEMENT

In the realm of safeguarding topology discovery services, extensive efforts have been undertaken. Sections V and VI have introduced defence mechanisms, and now, researchers are delving into enhancing link discovery security through OFDP modifications. However, while these endeavours shore up certain vulnerabilities and refine topology accuracy, they possess inherent limitations. Despite fortifying the correctness of the topology view, these solutions often overlook threats embedded in other construction phases. Moreover, their integration extends the LLDP, leading to an inevitable surge in network overhead. As we move ahead, related works addressing these challenges will be explored in the upcoming subsections.

A. SECURED DISCOVERY PROTOCOL

Ochoa-Aday et al. [69] introduced a new protocol designed to uncover layer 2 infrastructures within extensive SDN setups. Their proposed eTDP (enhanced Topology Discovery Protocol) efficiently delegates discovery functions among switches that support this protocol, creating a hierarchical distribution. Unlike existing methods, this solution allows for the automatic detection of network components without relying on prior IP configurations or the controller's prior knowledge of the network. By employing this mechanism, the SDN controller can seamlessly unveil the network's topology, constructing a comprehensive view without facing scalability challenges and utilizing the most efficient control paths to each switch. In experimental simulations using real-world topologies, the researchers demonstrated that eTDP offers an effective means of discovering network topology, achieving discovery times of under 0.08 milliseconds in the examined networks. The results indicate that increasing the number of SDN controllers does not impact the overall number of packets generated per switch. Additionally, eTDP outperforms OpenFlow-based approaches, showcasing significant improvements, especially when compared to the current OFDP (OpenFlow Discovery Protocol).

Rojas et al. [70] proposed, implemented and evaluated the Tree Exploration Discovery Protocol (TEDP), which is an improved and advanced algorithm for topology service without any additional messages in comparison with LLDP. The proposed TEDP algorithm initiates the topology discovery service at a single node with the help of flooding a probe framework in order to discover the SDN network and then, gather information. In contrast, other traditional algorithms poll each device and collect the replies afterwards, as in LLDP. Furthermore, they implemented their algorithm in two various ways: a simple SDN network in which the implementation inhabits the TEDP-S controller and a hybrid network in which the service is implemented in a shared status between the network switches and the SDN controller. The results of their proposed algorithm seem very promising since the number of control messages is decreased significantly and also topology discovery service is improved in order to provide latency-based paths more efficiently [70].

To accomplish the objective of minimizing the number of LLDP `PACKET-OUT` messages to one per switch, Pakzad et al. leverage two key features. Initially, when an OpenFlow switch establishes a connection with the controller, it conveys information about its ports, Port IDs, and associated MAC addresses through an OpenFlow OFPT `FEATURES REQUEST` message. This creates a direct mapping of MAC addresses to Port IDs for each switch in the controller. Additionally, they capitalize on the OpenFlow switches' capability to modify packet headers, commonly used for tasks like updating TTL fields or implementing Network Address Translation. Their proposed SDN topology discovery method, labelled OFDPv2, introduces specific changes to the existing version (OFDP):

1) The controller's behaviour is altered to restrict the number of LLDP `PACKET-OUT` messages sent to each switch to one. The Port ID TLV field in the LLDP payload is set to 0 and disregarded.
2) A new rule is implemented on each switch, directing that every LLDP packet received from the controller should be forwarded to all available ports. The source MAC address of the corresponding Ethernet frame is then set to the address of the port through which it is transmitted.
3) The `PACKET-IN` event handler on the controller, responsible for processing incoming LLDP packets, undergoes modification. Instead of parsing the Port ID TLV in the LLDP payload, the handler now examines the source MAC address of the Ethernet header. It looks up the corresponding Port ID in the controller's database, which maintains a one-to-one mapping of MAC addresses and switch Port IDs [71].

The primary concept behind sOFTD proposed by Azzouni et al. [47] involves shifting a portion of the discovery process from the controller to the switch while making minimal adjustments to the OpenFlow switch design. The key design features of sOFTD include:

- sOFTD introduces a port-liveness-detection mechanism (BFD) to the switch.
- It utilizes OpenFlow FAST-FAILOVER groups (which are optional in OpenFlow 1.1+) to monitor switch ports for connection updates.
- sOFTD allows the switch to notify the controller about changes in port connectivity.
- The switch is equipped with a rule ("drop lldp") to discard every LLDP packet, preventing LLDP flood attacks.
- The controller sends encrypted LLDP packets only when there is a switch-port connectivity update, and these packets are directed solely to the relevant switches.
- LLDP packets are accompanied by rules (with a hard timeout of 1 second) to redirect them back to the controller. These rules take precedence over the "drop lldp" rules.

Since sOFTD avoids sending periodic discovery packets, initial results indicate significantly improved performance compared to OFDP [47].

Azzouni et al. [72] built upon their earlier research addressing limitations in OFDP. They introduced a new topology discovery protocol for OpenFlow, known as sOFTDP, which involves minimal modifications to the OpenFlow switch design. Notably, their approach is inherently more secure than previous solutions applied to conventional OFDP. Additionally, through proof-of-concept experiments, they demonstrated that their proposal significantly outperforms both OFDP and OFDPv2 [71] by several orders of magnitude. The fundamental concept behind sOFTDP involves decentralizing a portion of the discovery process from the controller to the switch. With subtle adjustments to the OpenFlow switch design, sOFTDP empowers the switch to independently identify link events and communicate them to the controller. The controller is then equipped with the necessary logic to manage these switch notifications. The essential elements of sOFTDP's design include the utilization of Bidirectional Forwarding Detection (BFD) as a mechanism for port liveness detection, asynchronous notifications, a topology memory, FAST-FAILOVER groups, "drop lldp" rules, and hashed LLDP content [72].

Nehra and colleagues [73] introduced SLDP, a pioneering link discovery protocol tailored for SDN networks. SLDP stands out with its three-tiered security levels and lightweight design, achieved through a novel link discovery packet structure. This protocol proves to be more efficient by minimizing the generation and transmission of SLDP packets. The design of SLDP encompasses key elements such as the SLDP packet structure, system architecture, and event sequence. Unlike traditional LLDP packets, SLDP eliminates unnecessary Type-Length-Values (TLVs) like time to live (TTL) and EndTLV, which are irrelevant in SDN. SLDP introduces a fixed-length positional packet structure, optimizing the use of bits and simplifying the packet format. To bolster security, SLDP employs a token-based prevention approach to thwart poison, replay, and flooding attacks. It even introduces additional security levels, such as poison detection with mitigation and flood detection with mitigation, to address low-probability attacks. In the implementation, the controller initially dispatches SLDP packets to every port of a switch, but as iterations progress, certain ports are designated as non-eligible. This strategic decision reduces the number of SLDP packets generated and sent by the controller. The researchers implemented SLDP in the Mininet environment with the RYU controller, showcasing a significantly faster process of link discovery packet creation and verification compared to RYU's original OFDP (OpenFlow Discovery Protocol) implementation. SLDP's topology discovery outperforms OFDP due to its quick and targeted transmission of lightweight packets, resulting in lower CPU and bandwidth resource utilization [73].

Jia and colleagues introduce a novel probe frame structure for Lightweight Automatic Discovery Protocol (LADP), enhancing both traffic efficiency and security. The suggested LADP discovery method gathers link information while concurrently addressing injection, replay, and flooding attacks in the discovery process. The LADP approach involves four stages: Initialization; the second stage, where an LADP frame is created and sent to the data plane by the control plane; the third stage, where the LADP frame explores the network; and finally, the controller computes the topology information [74].

Chang et al. [75] introduced the inaugural delegation function designed for topology discovery in SDN. This method effectively reduces the delay associated with creating and sending LLDP frames from the controller. The aim is to expedite the detection of changes in network topology and

enhance its accuracy. Through experimental findings, they demonstrate that their approach not only attains excellent scalability and responsiveness for topology discovery but also does not impose additional CPU usage on switches. The suggested approach for discovering network topology is applied in both controllers and switches. Initially, LLDP-enabled OpenFlow switches are deployed, capable of generating LLDP frames and sending them to adjacent switches within one hop. Each switch identifies directly connected switches by receiving these LLDP frames. When a new neighbouring switch is linked or an existing one is disconnected, switches communicate these changes in topology to the controller. Within the controller, a topology manager module is established to collect reports of topology changes from switches and construct an updated representation of the network topology. This suggested mechanism seamlessly integrates with the original OFDP, enabling the controller to implement it selectively on chosen switches for various purposes. For instance, a controller might opt to allow remote switches to independently generate LLDP frames to promptly detect changes in network topology. Alternatively, the controller can delegate the synthesis of LLDP frames to all switches before executing critical or intricate tasks. This approach prevents the controller from consistently generating LLDP frames for all switch ports, significantly cutting down on both messaging and computational burdens. Moreover, the traffic load associated with maintaining network topology is substantially reduced, as switches only report updates to the controller. Consequently, the controller can deliver quicker response times for its applications [75].

Hauser et al. [76] proposed a novel scheme called P4-MACsec, which automatically protects links between switches with MACsec in P4-SDN. Their proposed scheme shows a P4 data plane implementation for MACsec including encryption and decryption. Their proposed new two-tier control plane contains local controllers operating on all of the P4 switches that are linked to a centralised controller. This proposed control plane is responsible for leading P4 switches. At the next stage, Hauser et al. proposed a modern secure scheme for link discovery by using automatic deployment of MACsec link protection and encrypted LLDP packets. The proposed P4-MACsec removes the prior settings efforts for MACsec [76].

B. SECURED DISCOVERY PROTOCOL IN HYBRID NETWORKS

Alvarez-Horcajo and colleagues [77] introduced the Hybrid Domain Discovery Protocol (HDDP), a groundbreaking protocol designed to efficiently uncover the complete network topology in hybrid SDN domains, where both SDN and non-SDN devices coexist. This protocol relies on an exploration mechanism initiated by the control plane, which uses a controlled flooding approach to reach all devices and gather essential information from the hybrid network. Unlike OFDP, their approach excels by discovering the entire hybrid topology in a network domain comprising SDN and non-SDN devices with a reduced number of exchanged packets [77]. HDDP's standout feature is its capability to integrate non-SDN devices and their bidirectional links into a comprehensive discovery of a hybrid SDN network. Given that non-SDN devices lack direct links to SDN controllers, they depend on SDN devices to relay their information. This connection may not be direct but indirect through other non-SDN devices. HDDP operates as a distributed protocol, relying on the exchange of HDDP Requests and Reply control messages to reveal the network topology. The structure of HDDP control messages is based on the minimum length of Ethernet networks, chosen as the infrastructure layer for HDDP implementation, although adaptability to other infrastructures like wireless scenarios is possible. The controller instructs its connected SDN devices to broadcast an HDDP Request message, initiating network exploration. Subsequently, all devices respond with corresponding HDDP Reply messages. These messages contain information crucial for the controller to attain a comprehensive view of the underlying topology, combining various data subsets extracted from these messages. The method by which HDDP triggers these HDDP Reply messages varies based on the nature of nodes, distinguishing between wired and wireless nodes [77].

Martinez-Yelmo et al. [78] proposed Enhanced Hybrid Domain Discovery Protocol (eHDDP) which is a new method for collecting information from full-hybrid SDN topologies. This proposed method is able to deal with wired/wireless links and also explore hybrid SDN topologies with SDN and non-SDN gadgets. upgraded the Hybrid Domain Discovery Protocol (HDDP) header in the control messages in order to support various kinds of wireless devices and interfaces. Moreover, they proposed an improved version of the Mininet-WiFi program which aims to work with wireless ad-Hoc networks with a huge number of gadgets that authorized to perform an assessment of eHDDP [78].

A new layer 2 link discovery scheme is suggested by Hussain et al. [79], accompanied by a fresh frame format. This enables the controller to identify legacyâ"OF and legacyâ"legacy links in an h-SDN. This not only facilitates bidirectional topology discovery in the h-SDN (except for nonterminal LSs) but also minimizes the need for extra messages. The suggested approach identifies links within the network through a solitary P_{out} message, eliminating the need for the one-way link detection typical in state-of-the-art protocols. In order to streamline the number of P_{in} messages, a lone packet sent from the OFS to the controller triggers bidirectional updates to the topology table. Experiments conducted in Mininet, based on the number of detected ports and messages required for topology retrieval within the network, demonstrate a significant improvement. The results show a 34.6%–97.9% increase in the number of detected ports and a reduction in the number of P_{in} messages by about 25%-58.9% compared to state-of-the-art protocols across diverse topologies.

The majority of existing protocols use a unidirectional process initiated by the controller to discover links, but they often miss important connections in certain network scenarios. In the proposed hybrid SDN approach, a new method is suggested to identify the global network topology efficiently. This involves circulating a single frame throughout the network, with each node appending relevant information. The OFS then sends this information to the controller, allowing for bidirectional topology optimization.

The proposed ICLF scheme reduces the number of messages needed for topology detection. Additionally, a novel frame format is introduced for this scheme. The frame formats, LLDP and ICLF, share standard fields like Preamble, Destination Address, Source Address, and Eth Type. However, they differ in the destination field, where LLDP uses a multicast address, and ICLF uses a broadcast address. In ICLF, the frame contains the Root MAC address, Source Port, and Destination Port, with the Root MAC address serving a role similar to LLDP's Chassis ID. Both LLDP and ICLF involve a relationship between source and destination addresses.

Unlike LLDP, ICLF includes both source and destination port IDs to establish topology at the controller, accounting for the lack of information on legacy port IDs. Before ICLF operates, the controller establishes proactive rules in the OFS. For P_{in} events, the OFS rule directs the packet to the controller, while P_{out} events send packets to all active switch interfaces. The P_{in} event rule in the OFS has a 5-second idle timeout, notifying the controller when the flow entry is removed. To prevent a broadcast storm, a port status variable ensures frames are only broadcasted when its value is 0; otherwise, no broadcast occurs [79].

IX. IMPORTANCE OF TOPOLOGY DISCOVERY SERVICE IN REAL-WORLD USE CASES

In this section, we have delved into the crucial role of SDN Topology Discovery Service within real-world applications, emphasizing its significance across diverse domains. Specifically, we explored its relevance within Software-Defined Industrial Networks, where the dynamic nature of industrial environments necessitates agile network management facilitated by TDS. Additionally, we scrutinized its pivotal role within Vehicular Networks, where the seamless coordination of vehicular communication demands accurate topology mapping provided by TDS. Furthermore, our discussion extended to the realm of Open-RAN 5G Networks, where the disaggregated architecture relies heavily on TDS for efficient orchestration and optimization of network resources. Through these varied contexts, the importance of TDS emerges as a fundamental enabler of network reliability, scalability, and adaptability, underscoring its indispensable role in modern network infrastructures.

A. SOFTWARE-DEFINED INDUSTRIAL NETWORKS

Connecting various machines and industrial systems through industrial networks can enhance productivity and cost efficiency by integrating information technology (IT), operation technology (OT), and communication technology (CT). Despite these benefits, traditional industrial networks face challenges in adapting to new manufacturing methods like flexible control and intelligent task scheduling. Manual offline pre-configuration is time-consuming and prone to errors, leading to scattered administration and coordination issues. SDN addresses these issues by separating control and data planes, enabling efficient networking, and offering a global network view for centralized management. SDN's unified interfaces enhance programmability and customizability, making it a valuable solution for challenges in industrial networks. Software-defined industrial networks (SDIN) have gained attention and are being applied in various industrial fields such as smart grids, vehicular networks, smart healthcare, and smart metering [80]. Wang et al. [80] studied the attacks against SDIN and their propagation due to multi-controller coordination. They also performed two types of attacks (topology forgery attack and packet-in flooding attack) in emulated distributed SDIN. Then, they illustrated their propagation among different controllers and supply more information related to the attack. Wang et al. [80] also proposed an attack mitigation platform based on reinforcement learning and deep Q-learning in order to defend the distributed SDIN against the mentioned attacks. This proposed reinforcement learning-based network can adaptively adjust the switch takeover decisions by dealing with the SDIN environment and exploring the arrival rate of requests provided by industrial devices in order to separate the attack source and endure the attack to some extent to buy time for network maintenance [80].

B. VEHICULAR NETWORK

Wang et al. [81] implemented two types of Topology Poisoning Attack (fake LLDP packet injection and LLDP packet relay) in 4 mainstream controllers in the emulated SDN-enabled vehicular edge network and then explored its influence from the RSU layer, the application layer, the controller layer and the vehicle layer. The proposed paper by Wang et al. [81] is the first study on the topic of TPA in the vehicular edge network which also introduced an attack tolerance scheme based on deep reinforcement learning algorithms in order to improve the vehicular edge network with a certain grade of self-retrieval and solve the issue that the TPA is complicated to be defended thoroughly [81].

The location hijacking attack was implemented on five mainstream SDN control planes by Wang and Liu [82] in the software-defined vehicular network emulator for the first time. They also hijack the vehicle's location in the popular Ryu SDN controller. Their attempt to perform the location hijacking attack upon the wireless terminal in SDN is done for the first time. Subsequently, they attempt to illustrate the importance of SDN security by summarizing more detail about the attacks and inspecting the attack impacts in three different layers: infrastructure layer, controller layer and vehicle layer. Wang and Liu [82] also implemented a

recovery program which was operating based on deep Q-learning (DQL) and reinforcement learning algorithms in order to deal with the location hijacking attacks upon servers that consist of rigid consequences. They assumed two states for attack recovery, there is/is not a secure data centre on the ground that collects data from different services. The service deployment of a software-defined spaceâ"air-ground integrated vehicular network can be adaptively modified with the help of exploration, and learning in the environment. The influenced service can be transformed into a high-altitude platform and there still will be access for the vehicles [82].

In addition, Wang et al. [83] have effectively compromised four commonly used controllers in simulated SDN-enabled vehicular networks by corrupting their topology views. They present two methods for executing these attacks (LLDP Packet Injection and LLDP Packet Relay) and compare the implementation details across the controllers. They segment the SDN-enabled vehicular network into four layers (Application Layer, Controller Layer, RSU Layer, Vehicle Layer) and assess the attack consequences in a hierarchical manner, aiming to raise awareness about SDN controller security. They also deliberate on countermeasures against these attacks and provide recommendations for enhancing security.

C. OPEN-RAN 5G NETWORK

O-RAN is a telecommunications network architecture that aims to standardize and virtualize the radio access network (RAN) elements in mobile communication systems. The goal of O-RAN is to promote interoperability, flexibility, and innovation in the development and deployment of RAN components. Traditionally, RAN components were provided by a single vendor, leading to closed and proprietary systems. O-RAN seeks to break down this traditional approach by defining open interfaces and standards, enabling different vendors to supply compatible and interchangeable components. This approach is expected to enhance competition, reduce costs, and accelerate the deployment of new technologies, ultimately benefiting both network operators and end-users in the mobile communication ecosystem.

There are significant similarities between SDN and, conversely, the RIC within the O-RAN framework. It is crucial to note that while SDN is a standalone paradigm, the RIC is intricately integrated within the O-RAN framework, highlighting the distinctive nature of their respective roles and architectures. The RIC in O-RAN and SDN both embrace centralized control and programmability for enhanced network management. SDN separates the control and data planes, enabling centralized control and dynamic resource allocation, with a focus on open interfaces for interoperability. Similarly, the RIC in O-RAN centralizes control in the radio access network, offering a programmable interface for optimizing radio resource allocation based on real-time conditions. Both concepts contribute to network flexibility, efficiency, and standardization, but SDN is a broader networking paradigm, while RIC is specifically designed for the radio access network within O-RAN architectures [84].

In our discussion, we'll explore the main components of O-RAN, unravelling key concepts for a clearer understanding of its architecture. O-RAN Central Unit–Control Plane (O-CU-CP) is a virtual node responsible for hosting the Radio Resource Control (RRC) and the control plane aspect of the Packet Data Convergence Protocol (PDCP). O-RAN Central Unit â" User Plane (O-CU-UP) is a virtual node designed to host the user plane component of the PDCP and the Service Data Adaptation Protocol (SDAP). O-RAN Distributed Unit (O-DU) functions as a virtual node that accommodates the Radio Link Control (RLC), MAC, and High-Physical layers based on a functional split at a lower layer. The RIC is a network function within the O-RAN architecture. It facilitates the control and optimization of RAN elements and resources through detailed data collection and actions performed over the E2 interface. The RIC may incorporate AI/ML workflows, encompassing tasks such as model training, inference, and updates. To gain a deeper understanding of O-RAN architecture, readers are encouraged to consult the relevant specifications in the references [85], [86].

The paper referenced as [87] presents a new attack called *Bearer Migration Poisoning (BMP)*. This attack aims to deceive the nRT RIC, leading to alterations in the user plane traffic path and resulting in signalling overhead. An interesting aspect of BMP is that it allows a relatively weak adversary with just two compromised hosts to execute the attack without compromising the RIC, RAN components, or applications. The attack relies on two specific procedures initiated by the RIC.

- *Bearer context migration procedure:* A bearer context refers to a set of signalling data transmitted through the E1 interface, linking the CU-CP and CU-UP. The establishment of a bearer context involves coordinating the necessary resources and information to enable the transfer of user plane services between the CU-UP, the corresponding DU, and the UE. To achieve this, the CU-CP utilizes bearer context management operations, which can be initiated by the nRT RIC platform [85]. In this process, the CU-CP sends a `BEARER CONTEXT SETUP` message to create a new bearer context between the specified CU-UP and DU. Subsequently, it signals the DU with a `F1 BEARER MODIFICATION` message to adjust the configuration of the F1 interface. Finally, the CU-CP issues a `BEARER CONTEXT RELEASE` message to discard the previous bearer context. This leads to the CU-CP transitioning the bearer context from the original CU-UP to the intended CU-UP for a specific DU. However, this functionality exposes Open RAN to an expanded range of security risks.
- *Link discovery procedure:* The RIC has the capability to oversee the network topology of RAN elements on the data plane via the E2 interface, involving the

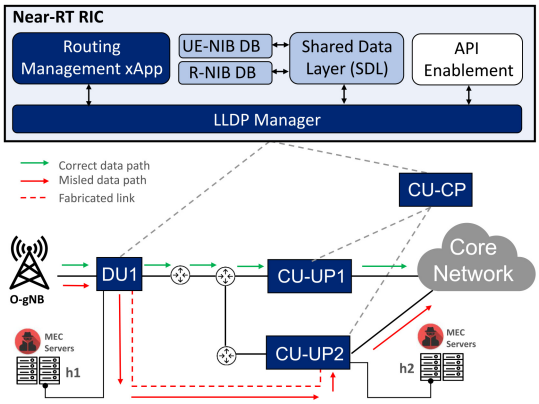

FIGURE 12. Bearer migration poisoning attack in Open RAN.

implementation of the link discovery protocol. Initially, the controller creates and sends LLDP messages to each node in the data plane. Subsequently, as each node receives the LLDP packets, it disseminates them through all of its ports. Following this, the node receiving these packets forwards the LLDP information back to the RIC. As a result, the controller can identify all the existing links among the nodes in the data plane. This systematic process is repeated at regular, predefined intervals.

By utilizing the procedure for migrating bearer context and the link discovery process, adversaries can strategically deceive the RIC, initiating malicious actions that pose a threat to the integrity of the network. In the subsequent sections, we further elaborate to provide a more detailed description of this potential threat.

Threat: The malicious actor aims to trick the RIC into believing that initiating a bearer context migration procedure is necessary. We illustrate the BMP attack using a simplified O-RAN structure, comprising a single DU, two CU-UPs, a pair of MEC servers acting as hosts, and multiple routers in the mid-haul network, as depicted in Fig. 12. It is assumed that the adversary has gained control over two MEC hosts, specifically h_1 and h_2, connected to DU_1 and CU-UP_2 respectively. In this particular attack scenario:

1) The malicious attacker at h_1 actively monitors the port associated with DU_1 (referred to as port 1 in this illustration). During the process of link discovery, the RIC sends out an LLDP packet toward DU_1. Upon receiving the LLDP packet, DU_1 broadcasts it across its ports. The adversary at h_1 intercepts this packet and redirects it to h_2 through an alternate communication channel. Subsequently, the compromised h_2 forwards the LLDP packet to $CU-UP_2$, which then relays it back to the RIC. The RIC, upon receiving the LLDP packet from $CU-UP_2$, is deceived into recording a non-existent link between DU_1 and $CU-UP_2$, as depicted by the dashed red line in Fig. 12. This maneuver is based on the link fabrication strategy outlined in [13], [66].
2) As the routing *xApp* is configured to receive updates regarding the radio topology of the network, it receives a `Topology Update Report` from the RIC, outlining the inclusion of the deceptive link. The routing algorithm, typically reliant on determining the shortest path, selects this new link as it perceives a more direct connection between DU_1 and $CU-UP_2$. Consequently, the *xApp* issues a `Path Update Request` to the RIC, requesting the application of the updated routing configuration.
3) Upon receiving the path update notification, the RIC generates a `RIC Control Request` to prompt the CU-CP to initiate modifications to the bearer context. Subsequently, upon receiving the request from the RIC, the CU-CP initiates the transition of CU-UP. This involves terminating the current bearer context with DU_1 and $CU-UP_1$ and establishing a new bearer context with DU_1 and $CU-UP_2$.

The reconfiguration of the bearer context, redirecting user traffic through the artificially established link between $CU-UP_1$ and $CU-UP_2$, has detrimental effects on the cell performance managed by the RU, as indicated by the red arrows in Fig. 12. As evidenced by empirical data and analyses presented in [87], a BMP attack can result in two significant outcomes. In the initial phase, it can significantly reduce both downlink and uplink throughput to nearly 0 Mbps, posing a severe threat to service quality and user satisfaction. This, in turn, may lead to potential customer and revenue loss for the operators. Secondly, the attack can cause a substantial increase in signalling overhead, leading to inflated network latency and the wastage of valuable radio spectrum resources. The signalling cost experiences an approximately tenfold increase due to the impact of the BMP attack.

X. CHALLENGES, OPEN ISSUES & FUTURE RESEARCH DIRECTIONS

In this section, we will delve into the limitations, challenges, open issues, and future research directions in the field of SDN security. We have categorized these aspects into three main groups: (a) The integration of AI/ML in the security of SDN topology discovery service security, (b) The integration of SDN with new technologies like cloud computing, IoT, and Open RAN, and (c) How the new digital twin technology can be utilized to enhance the security of the SDN topology discovery service.

A. ADVANCEMENTS IN MACHINE LEARNING

Given that machine learning and deep neural networks are integral to ensuring the security of SDN, we are confident that they hold the potential to address numerous outstanding issues and influence forthcoming trajectories. We have organized these challenges and future pathways into two main categories: 1) Advancements in algorithms and models, and 2) Improvements in dataset exploration.

1) ALGORITHMS AND MODELS

The future of security in SDN topology discovery services relies on advanced machine learning (ML) algorithms and anomaly detection systems. As SDN environments evolve, robust security measures are crucial for proactive threat identification and mitigation. ML algorithms, including unsupervised learning like clustering, autonomously identify patterns and detect anomalies without labelled training data. Semi-supervised learning combines labelled and unlabeled data, enhancing adaptability for recognizing both known and unknown threats. Deep learning models, such as neural networks like CNNs and RNNs, extract intricate features to understand SDN topology. Integrating these techniques into SDN security frameworks enhances anomaly detection for subtle and complex threats. Ensemble learning methodologies, combining multiple ML algorithms, ensure a robust security framework with minimized risks of false positives and negatives. In conclusion, the future of SDN security lies in integrating unsupervised and semi-supervised ML techniques, deep learning models, and ensemble methodologies for proactive defence against evolving cyber threats.

Another possible future work in the area of topology discovery service security could be the integration of Reinforcement Learning (RL) into SDN controllers that offers distinct advantages over other machine learning approaches. RL enhances adaptive capabilities, enabling dynamic policy adjustments in response to evolving threats. Unlike traditional machine learning, RL excels in learning optimal strategies through trial and error, making it well-suited for addressing complex challenges like mitigating topology poisoning attacks. Algorithms such as Q-learning or Deep Q Networks (DQN) in RL provide a unique edge by allowing agents to iteratively refine strategies, showcasing the potential for groundbreaking advancements in network security that surpass the capabilities of conventional machine learning methods [88].

Exploring the application of autoencoders for anomaly detection in the realm of security, particularly in the context of topology discovery services and the mitigation of TPAs, represents a potential future direction. Anomaly detection in SDN security encounters a significant challenge due to imbalanced datasets. In many cases, the occurrence of normal network behaviour far outweighs the instances of anomalous activities, leading to imbalanced data distributions. Tackling imbalanced datasets is a crucial aspect of effective anomaly detection. This prospective avenue of exploration is suggested in the proposed paper by Soltani et al. [60]. If the computational load of the neural network can be managed, autoencoders (AEs) have the potential to identify unsupervised anomalies [89]. AEs utilize compressed encoding from input and decoding for data reconstruction. Typically, normal input exhibits lower reconstruction errors, as it closely aligns with the training data, while abnormal input tends to result in higher reconstruction errors. However, this assumption may not hold in the LLDP dataset proposed by Soltani et al. [60], as it contains outliers and noise. As the network size increases, the complexity of the LLDP data grows, causing shared features, like TLLDP, between normal and abnormal LLDPs, leading to mixed feature values. In such cases, researchers may face challenges in properly reconstructing data from both normal and anomalous LLDPs. A potential solution is the use of denoising autoencoders, but this approach necessitates a source of clean, noise-free data for training, which is often unavailable in real-world networks. In the absence of such data, the RLV model proposed by Soltani et al. [60] might have to segregate noise and outliers from input values before training the encoder. However, this aspect is left for future research and remains unexplored. These proposed future research directions aim to provide a comprehensive roadmap for advancing the field of SDN security through the integration of machine learning, deep learning, and reinforcement learning techniques.

2) ENHANCEMENTS IN DATASET EXPLORATION

The future research direction titled Enhancing Topology Discovery in SDN for Robust ML-Based Countermeasures involves advancing the field by creating larger, real-world, and synthetic datasets specifically tailored for the topology discovery service in SDN. This strategic approach aims to bolster Machine Learning (ML) based countermeasures against topology poisoning attacks.

- *Expanding Dataset Scale:* Future research should focus on developing larger datasets with diverse network topologies, enhancing ML models' ability to distinguish normal behaviour from topology poisoning anomalies [90].
- *Incorporating Real-world Scenarios:* Future research should prioritize real-world datasets from SDN deployments, ensuring ML models are trained on diverse and complex data, enhancing the robustness and practicality of developed countermeasures [90].
- *Synthetic Data Generation Exploration:* Future research should explore creating synthetic datasets to simulate diverse network conditions and potential topology poisoning attacks. This allows researchers to systematically evaluate ML models under controlled conditions, enabling the development of countermeasures effective across various threat scenarios [90].

By prioritizing the development of larger, real-world, and synthetic datasets for topology discovery services, researchers bolster ML models with diverse training environments. This strengthens ML-based countermeasures, fortifying SDN against topology poisoning attacks in practical and challenging scenarios.

B. SYNERGIZING WITH EMERGING TECHNOLOGIES

One of the pivotal considerations in exploring limitations, facing challenges, and outlining future research directions lies in the synergy between SDN and emerging technologies. This includes the integration of SDN with transformative

innovations such as OPEN-RAN, Cloud Computing, and the IoT. These intersections are poised to be a focal point of discussion here, offering promising avenues for exploration and development in the evolving landscape of networking technologies.

1) THE INTEGRATION OF SDN AND OPEN-RAN

Future research directions in the integration of SDN and Open RAN involve exploring SDN functionality atop the Near Real-Time Radio Access Network Intelligent Controller (Near RT RIC) within the Open RAN architecture. This integration holds promise for enhancing network programmability and resource optimization. However, as SDN operates in this dynamic environment, potential challenges arise, particularly in the security domain. The interaction between the SDN Topology Discovery Service and the Open RAN may introduce novel vulnerabilities, leading to concerns such as topology poisoning attacks. Understanding and mitigating these risks will be crucial for ensuring the robustness of the integrated SDN and Open RAN system. Open RAN's disaggregated and modular architecture introduces a unique nature that necessitates a closer examination of its security aspects.

Notably, the security implications of Open RAN have yet to be comprehensively addressed within the research community, establishing a novel research area. The unique nature of Open RAN can intricately impact SDN security, amplifying the need for a focused exploration of potential vulnerabilities and risks. As the research community delves into this uncharted territory, it becomes imperative to unravel the interdependencies between Open RAN and SDN, specifically addressing security concerns arising from their integration. Understanding and addressing the security challenges introduced by Open RAN not only contribute to fortifying the Open RAN environment but also play a pivotal role in enhancing the overall security posture of the integrated SDN and Open RAN ecosystem. This underscores the significance of comprehensive research endeavours to establish robust security measures and frameworks tailored to the nuanced characteristics of Open RAN within the broader SDN context [91].

2) INTEGRATION OF SDN AND CLOUD

The integration of cloud computing environments with the SDN paradigm offers an innovative opportunity to seamlessly combine application provisioning in the cloud with the network, utilizing programmable interfaces and automation. However, challenges in evolving cloud networks include ensuring application performance during migration, enabling flexible deployment of appliances, addressing policy enforcement complexities and topology dependence, and managing security and privacy concerns. The demand is growing for a more programmable, flexible, and secure cloud infrastructure. SDN, as a novel networking paradigm, plays a key role in enhancing cloud manageability, scalability, controllability, and dynamism by transforming traditional cloud network backbones into robust service-delivery platforms. Recent developments in SDN-based cloud indicate a new type of cloud where SDN technology governs the network infrastructure, providing Networking-as-a-Service (NaaS). This evolution extends cloud computing beyond server and storage centralization to include network centralization and virtualization [92].

The integration of SDN with cloud computing indeed offers substantial benefits, but it also introduces new security challenges. One of the potential security risks involves the topology discovery service. To the best of our information, there currently isn't any security research available on this specific topic. Therefore, exploring the integration of SDN with cloud computing, particularly focusing on the security aspects of SDN topology discovery services integrated with cloud platforms, represents a significant area for future research. Identifying and addressing these research gaps is imperative to guarantee the resilience of the combined infrastructure and to proactively mitigate potential emerging risks.

3) SECURING SDN AND IOT

The exploration of future research directions in the realm of security within SDN integrated with the Internet of Things (IoT) is a complex and evolving domain, with a specific focus on the topology discovery service. In this context, SDN serves as a paradigm that separates the control plane from the data plane, providing centralized control and programmability. IoT, on the other hand, involves a myriad of interconnected devices generating vast amounts of data. The integration of SDN and IoT poses unique security challenges, necessitating a nuanced understanding of topology discovery services. Topology discovery involves the identification and mapping of network elements, and in the SDN-IoT security landscape, it becomes crucial for ensuring the integrity and confidentiality of communication. Detailed technical considerations include the development of secure and efficient algorithms for real-time topology discovery, intrusion detection and prevention, the implementation of authentication mechanisms for devices in the dynamic IoT environment, and the establishment of secure communication channels between SDN controllers and IoT devices. Addressing these intricacies requires a comprehensive approach that considers not only the diversity and scale of IoT deployments but also the dynamic nature of SDN configurations, making it an intriguing area for future research and innovation [22].

C. EMPOWERING TOPOLOGY DISCOVERY SERVICE IN DIGITAL TWINS

The incorporation of Digital Twins (DTs) into SDN architectures shows great promise in bolstering security measures. DTs serve as highly accurate digital duplicates of physical objects, offering valuable data and virtual prototypes for diverse purposes. Operators depend on DTs to improve preventive maintenance, spur innovation in business models,

hasten product development, and optimize sustainability and efficiency in real-world scenarios [93]. DTs provide a robust platform for proactive security testing, assessment, and risk mitigation. Security professionals can simulate various Topology Poisoning Attacks (TPA) scenarios within the DT environment, enabling the identification of vulnerabilities and the development of robust security measures before implementing them in the physical network. Through the integration of DTs with SDN, organizations can further enhance the security of the topology discovery service.

The SDN topology discovery service can leverage the simulation and prediction capabilities of the DT to analyze network behaviour, detect security vulnerabilities, and take proactive security measures. Centralized monitoring and management of security events within the DT environment enable organizations to gain a comprehensive view of the network and respond swiftly to security incidents affecting the topology discovery service. Furthermore, DTs offer real-time monitoring and analytics capabilities, allowing for the detection of network anomalies and security threats. By leveraging Machine Learning (ML) and analytics within the DT environment, security professionals can gain valuable insights into network behaviour, identify potential security risks, and respond promptly to mitigate them. This proactive approach to security, enabled by DTs, helps protect the integrity and confidentiality of the topology discovery service and the overall SDN infrastructure. It is imperative to conduct further investigation and research to fully explore the potential of integrating DTs into the SDN topology discovery service and to address the specific security considerations in this context.

XI. CONCLUSION

SDNs represent a revolutionary shift in network architecture, providing unparalleled flexibility and programmability. At the core of SDNs is the Topology Discovery Service, a crucial element dynamically identifying and mapping the network structure. Network topology discovery is the process by which the controller learns about (i) the network devices (e.g., switches), (ii) the links between switches and (iii) the location of the hosts within the network. Securing the Topology Discovery Service is imperative to safeguard against unauthorized access and data manipulation, preserving the integrity and reliability of SDNs. A holistic approach to topology discovery and security is vital as SDNs evolve, unlocking their full transformative potential.

In this paper, we have offered a thorough examination of the security aspects pertaining to SDN architecture and topology discovery services. Our study aims to contribute to the understanding of topology poisoning attacks in SDN by providing a detailed summary and technical root cause analysis, building upon existing research in the field. The topology discovery service is vulnerable to Topology Poisoning Attacks. The reasons are mainly attributed to several security vulnerabilities in the Host Tracking Service and Link Discovery Service. First, we delved into the intricacies of SDN architecture, followed by an exploration of topology discovery services. Subsequently, we extensively addressed the topic of topology poisoning attacks. We reviewed and summarized the various types of scenarios in the Link Fabrication Attack and Host Location Hijacking attack. In addition to our in-depth analysis, we have also established a comprehensive taxonomy for these attacks. We categorize these security threats based on various parameters, encompassing the attack's objectives, the targeted vulnerable entity, the adversary's location, and the nature of the communication channel involved. This taxonomy serves as a structured framework to better understand and address the multifaceted challenges posed by security in SDN topology discovery services.

We also discussed the existing security countermeasures along with analysing their potential security vulnerabilities. Finally, we examined the limitations, challenges, open issues, and proposed future research directions within the realm of security in SDN topology discovery services. These insights are anticipated to be invaluable for academia and industry, providing a foundation for their work and contributing to the advancement of this field.

REFERENCES

[1] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 1–12, 2007.

[2] M. Casado et al., "SANE: A protection architecture for Enterprise networks," in *Proc. USENIX Security. Symp.*, 2006, p. 50.

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

[4] (Res. Market, Dublin, Ireland). *Global Software-Defined Networking Market by Component (SDN Infrastructure, Services, Solutions), SDN Types (Open SDN, SDN via API, SDN via Overlay), Organization Size, End-User, Vertical-Forecast 2023-2030*. (2023). [Online]. Available: https://www.researchandmarkets.com/reports/5639407/global-software-defined-networking-market-by#src-pos-1

[5] (Ponemon Inst., Michigan, IN, USA). *Cost of a Data Breach Report 2023*. 2023. [Online]. Available: https://www.ibm.com/downloads/cas/E3G5JMBP

[6] (Ponemon Inst., Michigan, IN, USA). *Cost of a Data Breach Report 2020*. 2020. [Online]. Available: https://bit.ly/3l0AjR4

[7] Q. Long, Y. Chen, H. Zhang, and X. Lei, "Software defined 5G and 6G networks: A survey," *Mobile Netw. Appl.*, vol. 27, no. 5, pp. 1792–1812, 2022.

[8] B. Balasubramanian et al., "RIC: A RAN intelligent controller platform for AI-enabled cellular networks," *IEEE Internet Comput.*, vol. 25, no. 2, pp. 7–17, Apr. 2021.

[9] S. Khan, A. Gani, A. W. A. Wahab, M. Guizani, and M. K. Khan, "Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 303–324, 1st Quart., 2017.

[10] T. Arnold et al., "Beating BGP is harder than we thought," in *Proc. 18th ACM Workshop Hot Topics Netw.*, 2019, pp. 9–16.

[11] T.-H. Nguyen and M. Yoo, "Analysis of link discovery service attacks in SDN controller," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, 2017, pp. 259–261.

[12] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures," in *Proc. NDSS*, 2015, pp. 8–11.

[13] R. Skowyra et al., "Effective topology tampering attacks and defenses in software-defined networks," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw. (DSN)*, 2018, pp. 374–385.

[14] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting security attacks in software-defined networks," in *Proc. NDSS*, 2015, pp. 8–11.

[15] J. Kim et al., "Enhancing security in SDN: Systematizing attacks and defenses from a penetration perspective," *Comput. Netw.*, vol. 241, Mar. 2024, Art. no. 110203.

[16] Z. A. Bhuiyan, S. Islam, M. M. Islam, A. A. Ullah, F. Naz, and M. S. Rahman, "On the (in)security of the control plane of SDN architecture: A survey," *IEEE Access*, vol. 11, pp. 91550–91582, 2023.

[17] M. Rahouti, K. Xiong, Y. Xin, S. K. Jagatheesaperumal, M. Ayyash, and M. Shaheed, "SDN security review: Threat taxonomy, implications, and open challenges," *IEEE Access*, vol. 10, pp. 45820–45854, 2022.

[18] M. B. Jimenez, D. Fernandez, J. E. Rivadeneira, L. Bellido, and A. Cardenas, "A survey of the main security issues and solutions for the SDN architecture," *IEEE Access*, vol. 9, pp. 122016–122038, 2021.

[19] O. S. Al-Heety, Z. Zakaria, M. Ismail, M. M. Shakir, S. Alani, and H. Alsariera, "A comprehensive survey: Benefits, services, recent works, challenges, security, and use cases for SDN-VANET," *IEEE Access*, vol. 8, pp. 91028–91047, 2020.

[20] E. Marin, N. Bucciol, and M. Conti, "An in-depth look into SDN topology discovery mechanisms: Novel attacks and practical countermeasures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1101–1114.

[21] A. Abdou, P. C. van Oorschot, and T. Wan, "Comparative analysis of control plane security of SDN and conventional networks," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3542–3559, 4th Quart., 2018.

[22] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 812–837, 1st Quart., 2019.

[23] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful SDN data planes," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1701–1725, 3rd Quart., 2017.

[24] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 623–654, 1st Quart., 2015.

[25] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, 4th Quart., 2015.

[26] J. H. Cox et al., "Advancing software-defined networks: A survey," *IEEE Access*, vol. 5, pp. 25487–25526, 2017.

[27] "A Linux Foundation Collaborative Project." Open vSwitch. 2024. [Online]. Available: http://openvswitch.org

[28] N. McKeown et al., "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[29] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, "Network configuration protocol (NETCONF)," Internet Res. Task Force, RFC 6241, Jun. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6241.txt

[30] M. Smith, M. Dvorkin, Y. Laribi, V. Pandey, P. Garg, and N. Weidenbacher, "OpFlex control protocol," Internet Engineering Task Force, Internet-Draft draft-smith-opflex-00, 2014, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-smith-opflex-00

[31] W. Zhou, L. Li, M. Luo, and W. Chou, "REST API design patterns for SDN northbound API," in *Proc. 28th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, 2014, pp. 358–365.

[32] "OpenStack neutron." Accessed: 5 May 2024. [Online]. Available: https://docs.openstack.org/neutron/latest/

[33] S. Ahmad and A. H. Mir, "Scalability, consistency, reliability and security in SDN controllers: A survey of diverse SDN controllers," *J. Netw. Syst. Manage.*, vol. 29, no. 1, pp. 1–59, 2021.

[34] N. Gude et al., "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, 2008.

[35] R. Sherwood et al., "Flowvisor: A network virtualization layer," OpenFlow, Mesa, AZ, USA, Rep. TR-2009-1, 2009.

[36] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, W. Snow, and G. Parulkar, *OpenVirteX: A Network Hypervisor*, Open Netw. Summit, Newport, VIC, Australia, 2014.

[37] "An instant virtual network on your laptop (or other PC)." MININET. [Online]. Available: http://mininet.org/

[38] R. Amin, M. Reisslein, and N. Shah, "Hybrid SDN networks: A survey of existing approaches," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3259–3306, 4th Quart., 2018.

[39] S. Jain et al., "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[40] "Software-defined networking: The new norm for networks," Open Netw. Found., Palo Alto, CA, USA, White Paper, Apr. 2012.

[41] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) protocol version 1.2," Internet Eng. Task Force, RFC 5246, Aug. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5246.txt

[42] "OpenFlow switch specification, Version 1.5.1," Open Netw. Fund., Palo Alto, CA, USA, Rep. ONF TS-025, Mar. 2015.

[43] Y. Zhao, J. Yan, and H. Zou, "Study on network topology discovery in IP networks," in *Proc. 3rd IEEE Int. Conf. Broadband Netw. Multimedia Technol. (IC-BNMT)*, 2010, pp. 186–190.

[44] *802.1AB-2009 IEEE Standard for Local and Metropolitan Area Networks–Station and Media Access Control Connectivity Discovery, Corrigendum 2: Technical and Editorial Corrections*, IEEE Standard 802.1AB-2009 (Revision of IEEE Std 802.1AB-2005), 2015.

[45] L. Ochoa Aday, C. Cervelló Pastor, and A. Fernández Fernández, *Current Trends of Topology Discovery in OpenFlow-Based Software Defined Networks*, UPCommons, Castelldefels, Spain, 2015.

[46] (Cisco, San Jose, CA, USA). *Cisco Discovery Protocol Configuration Guide, Cisco IOS Release 15M&T*. 2016. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cdp/configuration/15-mt/cdp-15-mt-book/nm-cdp-discover.html

[47] A. Azzouni, N. T. M. Trang, R. Boutaba, and G. Pujolle, "Limitations of OpenFlow topology discovery protocol," in *Proc. 16th Annu. Mediterr. Ad Hoc Netw. Workshop (Med-Hoc-Net)*, 2017, pp. 1–3.

[48] X. Huang, S. Cheng, K. Cao, P. Cong, T. Wei, and S. Hu, "A survey of deployment solutions and optimization strategies for hybrid SDN networks," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1483–1507, 2nd Quart., 2018.

[49] P. Shrivastava and K. Kataoka, "Topology poisoning attacks and prevention in hybrid software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 510–523, Mar. 2022.

[50] H. S. Abdulkarem and A. Dawod, "DDoS attack detection and mitigation at SDN data plane layer," in *Proc. 2nd Global Power, Energy Commun. Conf. (GPECOM)*, 2020, pp. 322–326.

[51] J. Abley, K. Lindqvist, E. Davies, B. Black, and V. Gill, "IPv4 multihoming practices and limitations," Internet Eng. Task Force, RFC 4116, Jul. 2005.

[52] J. Hua, Z. Zhou, and S. Zhong, "Flow misleading: Worm-hole attack in software-defined networking via building in-band covert channel," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1029–1043, 2020.

[53] D. Smyth, S. McSweeney, D. O'Shea, and V. Cionca, "Detecting link fabrication attacks in software-defined networks," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2017, pp. 1–8.

[54] Y. Gao and M. Xu, "Defense against software-defined network topology poisoning attacks," *Tsinghua Sci. Technol.*, vol. 28, no. 1, pp. 39–46, 2022.

[55] S. Deng, L. Chen, and X. Gao, "Manipulating sensitive match fields to poison applications in SDN," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 2, pp. 2413–2425, Apr. 2024.

[56] S. Deng, X. Qing, X. Li, X. Gao, and X. Gao, "SDN application Backdoor: Disrupting the service via poisoning the topology," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2023, pp. 1–10.

[57] D. Kong et al., "Combination attacks and defenses on SDN topology discovery," *IEEE/ACM Trans. Netw.*, vol. 31, no. 2, pp. 904–919, Apr. 2023.

[58] M. Antikainen, T. Aura, and M. Särelä, "Spook in your network: Attacking an SDN with a compromised OpenFlow switch," in *Proc. Nordic Conf. Secure IT Syst.*, 2014, pp. 229–244.

[59] A. Alimohammadifar et al., "Stealthy probing-based verification (SPV): An active approach to defending software defined networks against topology poisoning attacks," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2018, pp. 463–484.

[60] S. Soltani, M. Shojafar, H. Mostafaei, and R. Tafazolli, "Real-time link verification in software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 3596–3611, Sep. 2023.

[61] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," Internet Eng. Task Force, RFC 2104, 1997.

[62] T. Alharbi, M. Portmann, and F. Pakzad, "The (in)security of topology discovery in software defined networks," in *Proc. IEEE 40th Conf. Local Comput. Netw. (LCN)*, 2015, pp. 502–505.
[63] S. Deng, W. Dai, X. Qing, and X. Gao, "Vulnerabilities in SDN topology discovery mechanism: Novel attacks and countermeasures," *IEEE Trans. Depend. Secure Comput.*, early access, Sep. 11, 2023, doi: 10.1109/TDSC.2023.3314111.
[64] B. Yuan et al., "Towards automated attack discovery in SDN controllers through formal verification," *IEEE Trans. Netw. Service Manag.*, early access, Apr. 10, 2024, doi: 10.1109/TNSM.2024.3386404.
[65] S. Jero, W. Koch, R. Skowyra, H. Okhravi, C. Nita-Rotaru, and D. Bigelow, "Identifier binding attacks and defenses in software-defined networks," in *Proc. 26th USENIX Security. Symp. (USENIX Secur. 17)*, 2017, pp. 415–432.
[66] S. Soltani, M. Shojafar, H. Mostafaei, Z. Pooranian, and R. Tafazolli, "Link latency attack in software-defined networks," in *Proc. 17th Int. Conf. Netw. Service Manage. (CNSM)*, 2021, pp. 187–193.
[67] F. Mvah, V. K. Tchendji, C. T. Djamegni, A. H. Anwar, D. K. Tosh, and C. Kamhoua, "Deception-based IDS against ARP spoofing attacks in software-defined networks," in *Proc. Workshop Comput., Netw. Commun. (CNC)*, 2024, pp. 188–192.
[68] V. Hnamte and J. Hussain, "Enhancing security in software-defined networks: An approach to efficient ARP spoofing attacks detection and mitigation," *Telemat. Informat. Rep.*, vol. 14, Jun. 2024, Art. no. 100129.
[69] L. Ochoa-Aday, C. Cervelló-Pastor, and A. Fernández-Fernández, "ETDP: Enhanced topology discovery protocol for software-defined networks," *IEEE Access*, vol. 7, pp. 23471–23487, 2019.
[70] E. Rojas, J. Alvarez-Horcajo, I. Martinez-Yelmo, J. A. Carral, and J. M. Arco, "TEDP: An enhanced topology discovery service for software-defined networking," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1540–1543, Aug. 2018.
[71] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in software defined networks," in *Proc. 8th Int. Conf. Signal Process. Commun. Syst. (ICSPCS)*, 2014, pp. 1–8.
[72] A. Azzouni, R. Boutaba, N. T. Mai Trang, and G. Pujolle, "sOFTDP: Secure and efficient topology discovery protocol for SDN," 2017, *arXiv:1705.04527*.
[73] A. Nehra, M. Tripathi, M. S. Gaur, R. B. Battula, and C. Lal, "SLDP: A secure and lightweight link discovery protocol for software defined networking," *Comput. Netw.*, vol. 150, pp. 102–116, Feb. 2019.
[74] Y. Jia, L. Xu, Y. Yang, and X. Zhang, "Lightweight automatic discovery protocol for OpenFlow-based software defined networking," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 312–315, Feb. 2020.
[75] Y.-C. Chang, H.-T. Lin, H.-M. Chu, and P.-C. Wang, "Efficient topology discovery for software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1375–1388, Jun. 2021.
[76] F. Hauser, M. Schmidt, M. Häberle, and M. Menth, "P4-MACsec: Dynamic topology monitoring and data layer protection with MACsec in P4-SDN," 2019, *arXiv:1904.07088*.
[77] J. Alvarez-Horcajo, E. Rojas, I. Martinez-Yelmo, M. Savi, and D. Lopez-Pajares, "HDDP: Hybrid domain discovery protocol for heterogeneous devices in SDN," *IEEE Commun. Lett.*, vol. 24, no. 8, pp. 1655–1659, Apr. 2020.
[78] I. Martinez-Yelmo, J. Alvarez-Horcajo, J. A. Carral, and D. Lopez-Pajares, "eHDDP: Enhanced hybrid domain discovery protocol for network topologies with both wired/wireless and SDN/non-SDN devices," *Comput. Netw.*, vol. 191, May 2021, Art. no. 107983.
[79] M. W. Hussain, K. H. K. Reddy, J. J. Rodrigues, and D. S. Roy, "An indirect controller-legacy switch forwarding scheme for link discovery in hybrid SDN," *IEEE Syst. J.*, vol. 15, no. 2, pp. 3142–3149, Jun. 2021.
[80] J. Wang, J. Liu, H. Guo, and B. Mao, "Deep reinforcement learning for securing software-defined industrial networks with distributed control plane," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 4275–4285, Jun. 2022.
[81] J. Wang, Y. Tan, and J. Liu, "Topology poisoning attacks and countermeasures in SDN-enabled vehicular networks," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, 2020, pp. 1–6.
[82] J. Wang and J. Liu, "Location hijacking attack in software-defined space-air-ground integrated vehicular network," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5971–5981, Apr. 2022.
[83] J. Wang, Y. Tan, J. Liu, and Y. Zhang, "Topology poisoning attack in SDN-enabled vehicular edge network," *IEEE Internet Things*, vol. 7, no. 10, pp. 9563–9574, Oct. 2020.
[84] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 1376–1411, 2nd Quart., 2023.
[85] *O-RAN Architecture Description, O-RAN Alliance, Version 11.00*, O-RAN Working Group, Washington, DC, USA, 2024.
[86] *O-RAN Cloudification and Orchestration Use Cases and Requirements for O-RAN Virtualized RAN, O-RAN Alliance, Version 09.00*, O-RAN Working Group, Washington, DC, USA, 2024.
[87] S. Soltani, M. Shojafar, A. Brighente, M. Conti, and R. Tafazolli, "Poisoning bearer context migration in O-RAN 5G network," *IEEE Wireless Commun. Lett.*, vol. 12, no. 3, pp. 401–405, Mar. 2023.
[88] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A survey of networking applications applying the software defined networking concept based on machine learning," *IEEE Access*, vol. 7, pp. 95397–95417, 2019.
[89] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2017, pp. 665–674.
[90] J. Xie et al., "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 393–430, 1st Quart., 2019.
[91] A. S. Abdalla, P. S. Upadhyaya, V. K. Shah, and V. Marojevic, "Toward next generation open radio access networks: What O-RAN can and cannot do!" *IEEE Netw.*, vol. 36, no. 6, pp. 206–213, Dec. 2022.
[92] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.
[93] L. U. Khan, W. Saad, D. Niyato, Z. Han, and C. S. Hong, "Digital-twin-enabled 6G: Vision, architectural trends, and future directions," *IEEE Commun. Mag.*, vol. 60, no. 1, pp. 74–80, Jan. 2022.

SANAZ SOLTANI received the master's degree in software engineering from the Amirkabir University of Technology (Tehran Polytechnic), Iran, 2014. She is a Ph.D. Researcher of Information and Communication Systems with the 5GIC & 6GIC Innovation Centre, University of Surrey, U.K. Before, she was a Network Specialist with Huawei and MTN telecommunication companies involved in 4G and LTE projects. Her research interests include network softwarization, Open RAN, network security, and privacy.

ALI AMANLOU recived the B.Sc. degree (Hons.) in electrical engineering, specializing in telecommunications. He is a Visiting Researcher with the 5GIC and 6GIC Innovation Centre, University of Surrey, U.K. He has authored numerous publications in peer-reviewed journals. He frequently serves as a paper reviewer for esteemed journals, such as IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON IMAGE PROCESSING, and IEEE ACCESS. His research interests encompass artificial intelligence, deep learning, 5G/6G technology, network security, and open RAN.

MOHAMMAD SHOJAFAR (Senior Member, IEEE) received the Ph.D. degree in ICT from the Sapienza University of Rome, Rome, Italy, in 2016, with an "Excellent" degree. He is a Senior Lecturer (Associate Professor) of Network Security with the 5G and 6G Innovation Centre (5G/6GIC), Institute for Communication Systems, University of Surrey, U.K. He is an Intel Innovator. Before joining 5G/6GIC, he was a Senior Researcher and a Marie Curie Fellow with the SPRITZ Security and Privacy Research Group, University of Padua, Italy. He secured around £1.9M as PI in various EU/U.K. projects, including ORAN-TWIN (funded by EPSRC/DSIT CHEDDAR Hub U.K.; 2024), D-XPERT (funded by I-UK/U.K.; 2024), 5G MoDE (funded by DSIT/U.K.; 2023), 5G ONE4HDD (funded by DSIT/U.K.; 2023), TRACE-V2X (funded by EU/MSCA-SE; 2023), AUTOTRUST (funded by ESA/EU; 2021), PRISENODE (funded by EU/MSCA-IF; 2019), and SDN-Sec (funded by Italian Government; 2018). He was also a COI of various U.K./EU projects like 6G_SMART (funded by CELTIC-NEXT; 2024), HiPER-RAN (funded by DSIT/U.K.; 2023), APTd5G Project (funded by EPSRC/UKI-FNI; 2022), ESKMARALD (funded by UK/NCSC; 2022), GAUChO, S2C, and SAMMClouds (funded by Italian Government; 2016–2018). He is an Associate Editor of IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, and *Computer Networks*. He is a Professional ACM Member, an ACM Distinguished Speaker, a Fellow of the Higher Education Academy, and a Marie Curie Alumni.

RAHIM TAFAZOLLI (Senior Member, IEEE) is the Regius Professor and a Professor of Mobile and Satellite Communications. He is the Director of ICS and the Founder and Director of the world's first 5G Innovation Centre, University of Surrey, U.K. Many governments regularly invite him for advice on mobile communications and, in particular, 5G technologies. He has given many interviews to International media in the form of television, radio interviews, and articles in the international press.