future internet

MDPI

Article

Multi-Constraint and Multi-Policy Path Hopping Active Defense Method Based on SDN

Bing Zhang [1,2,3], Hui Li [1,2,3], Shuai Zhang [2,4,5], Jing Sun [1,2,3], Ning Wei [2,3,5,*], Wenhong Xu [1,2,3] and Huan Wang [1,2,3]

[1] School of Computer Science and Technology, Guangxi University of Science and Technology, Liuzhou 545006, China; 20210701009@stdmail.gxust.edu.cn (B.Z.); lihui@gxust.edu.cn (H.L.); Sunjing_gxust@gxust.edu.cn (J.S.); 20230702029@stdmail.gxust.edu.cn (W.X.); wanghuan@gxust.edu.cn (H.W.)
[2] Liuzhou Key Laboratory of Big Data Intelligent Processing and Security, Liuzhou 545006, China; 221077427@stdmail.gxust.edu.cn
[3] Cybersecurity Monitoring Center for Guangxi Education System, Liuzhou 545006, China
[4] School of Science, Guangxi University of Science and Technology, Liuzhou 545006, China
[5] School of Automotive and Information Engineering, Guangxi Eco-Engineering Vocational and Technical College, Liuzhou 545004, China
* Correspondence: 100000894@gxust.edu.cn

Abstract: Path hopping serves as an active defense mechanism in network security, yet it encounters challenges like a restricted path switching space, the recurrent use of similar paths and vital nodes, a singular triggering mechanism for path switching, and fixed hopping intervals. This paper introduces an active defense method employing multiple constraints and strategies for path hopping. A depth-first search (DFS) traversal is utilized to compute all possible paths between nodes, thereby broadening the path switching space while simplifying path generation complexity. Subsequently, constraints are imposed on residual bandwidth, selection periods, path similitude, and critical nodes to reduce the likelihood of reusing similar paths and crucial nodes. Moreover, two path switching strategies are formulated based on the weights of residual bandwidth and critical nodes, along with the calculation of path switching periods. This facilitates adaptive switching of path hopping paths and intervals, contingent on the network's residual bandwidth threshold, in response to diverse attack scenarios. Simulation outcomes illustrate that this method, while maintaining normal communication performance, expands the path switching space effectively, safeguards against eavesdropping and link-flooding attacks, enhances path switching diversity and unpredictability, and fortifies the network's resilience against malicious attacks.

Keywords: active defense; path hopping; multi-constraint; multi-strategy

Citation: Zhang, B.; Li, H.; Zhang, S.; Sun, J.; Wei, N.; Xu, W.; Wang, H. Multi-Constraint and Multi-Policy Path Hopping Active Defense Method Based on SDN. *Future Internet* **2024**, *16*, 143. https://doi.org/10.3390/fi16040143

Academic Editor: Georgios Kambourakis

Received: 14 March 2024
Revised: 15 April 2024
Accepted: 17 April 2024
Published: 22 April 2024

1. Introduction

The rapid advancement of computer networks has revolutionized daily life but has also unveiled a myriad of network security challenges. Traditional network architectures inherently harbor security vulnerabilities during design and implementation. Exploiting the ubiquity of network systems, attackers continuously innovate attack strategies to amplify the impact of existing vulnerabilities and uncover novel threats. Defenders, constrained by a time lag, rely on past experiences and existing tools, struggling to thwart all attacker assaults and identify every potential vulnerability, resulting in an information asymmetry between attackers and defenders. Moreover, the static composition and configuration of conventional networks offer a conducive environment for malicious activities. The unchanging system properties grant attackers ample time for pre-attack preparations such as reconnaissance and target identification. As network usage is prolonged, attackers have more opportunities to exploit system vulnerabilities, intensify preparations, and elevate the success rate and impact of their attacks. Post-attack, installing undetectable back doors for

sustained control becomes feasible due to the system's static nature, rendering defenders reactive over time.

Conventional defense mechanisms typically rely on installing firewalls, IDSs (intrusion detection systems), IPSs (intrusion prevention systems), and other security devices at network perimeters to block attackers and reinforce existing system technologies and protocols, while this enhances system security to some extent, it necessitates significant human and material resources. In contrast, attackers, leveraging diverse attack methods and readily available tools, incur minimal time costs. By exploiting system vulnerabilities and rapidly propagating through susceptible nodes, attackers can dismantle entire systems and reap substantial gains at minimal expense, establishing a cost disparity between attackers and defenders. In essence, the deterministic nature of traditional network structures, static configurations, and passive defense approaches aimed at bolstering security through blocking are increasingly inadequate against evolving and sophisticated attack methodologies. This perpetual passivity places defenders at a disadvantage in the ongoing battle between attackers and defenders. Consequently, there is a pressing need for an effective active defense strategy to counter these challenges.

To alleviate the challenges faced by network security and change the asymmetric situation of attack and defense, the US military has proposed the moving target defense technology (MTD) [1]. Unlike traditional defense measures that aim to enhance the overall security of the system and eliminate all security threats, MTD defends during the attacker's pre-attack preparation period and does not pursue the establishment of a perfect defense system. Its main idea is to constantly change the form features of the target system by constructing and implementing defense strategies with element diversity, dynamic structure, and compositional uncertainty. This increases the diversity, dynamism, randomness, and unpredictability of system resource properties; limits the opportunities for vulnerabilities to be continuously exposed and exploited; increases attack difficulty and attack cost; improves system resilience; and enhances active defense capabilities [2]. Path hopping, as a typical MTD technology, aims to continuously and dynamically change the transmission path during data transmission, enhance the uncertainty of the attack target, resist attackers' long-term continuous listening to the transmission path, increase the difficulty and cost of attacker detection, and enhance the defense capability and defense benefits of the network system. The traditional network architecture, with its static and deterministic characteristics, is not well suited for the large-scale deployment of MTD. The emergence of software-defined networking (SDN) has effectively addressed the shortcomings of traditional network architectures, providing a solid platform for research into the deployment of MTD strategies.

In current research, the generation of path hopping space mostly uses the SMT (satisfiability modulo theories) constraint solving method. This method has a relatively high time complexity when dealing with large-scale and high-complexity networks, and it rises exponentially with the increase in nodes, so this method cannot be used in large-scale networks. Secondly, there is a problem in current research that the constraints for generating path hopping space are too strict or insufficient. Being too strict will shrink the path hopping space, and some paths will be reused multiple times in a short time. Insufficient constraints may lead to the unavailability of hopping paths or the repeated use of key nodes in the short term. Finally, the current research on the triggering method of path hopping is relatively single, lacking mechanisms to adaptively adjust hopping strategies and hopping cycles for different attacks. These will reduce the diversity, unpredictability, and effectiveness of path hopping.

This paper proposes an SDN-based multi-constraint and multi-policy path hopping active defense method (SDADM), with the following main contributions:

- SDADM utilizes depth-first search (DFS) traversal to compute all paths between communication host nodes, expanding the path hopping space while reducing the complexity of path generation. Furthermore, constraints are applied to residual band-

width, selection periods, similar paths, and critical nodes to decrease the likelihood of repetitive use of similar paths and critical nodes.

- SDADM designs two path hopping strategies and calculates path hopping periods based on the weights of residual bandwidth and critical nodes. It achieves adaptive switching of path hopping and hopping periods under different attack scenarios using the residual bandwidth of paths in the network as a threshold.
- Simulation experiments using Mininet demonstrate that SDADM can expand the path hopping space while ensuring normal communication performance. It effectively defends against eavesdropping and link-flooding attacks, enhances the diversity and unpredictability of path hopping, and strengthens the network's ability to defend against malicious attacks.

The content of this paper is arranged as follows. Section 2 introduces the related work of this study. Section 3 introduces the overall architecture of SDADM. Section 4 introduces the design concept and execution process of the path hopping strategy algorithm. Section 5 shows the experimental verification results and analysis of SDADM. Section 6 concludes this paper.

2. Related Works

Path hopping is a method of dynamically adjusting network traffic paths and is one of the key technologies of MTD. It mainly resists eavesdropping by attackers and increases the difficulty and cost of attackers' detection by constantly changing the communication path between the two parties during communication, thereby improving the security of the network.

In terms of resisting network attacks, to resist eavesdropping and DoS attacks, Dolev et al. [3] proposed a multi-path hopping scheme based on the $n - k$ threshold to resist eavesdropping attacks. This scheme prevents attackers from obtaining a large amount of data in long-term monitoring by limiting the maximum data flow passing through the same path. Duan et al. [4] proposed a random route mutation method (RRM) to resist DoS attacks and eavesdropping. This method transforms the problem of generating path hopping space into a constraint satisfaction problem; sets constraints in terms of capacity, overlap, and QoS; and generates paths using the SMT constraint solving method. The authors also proposed the implementation of this method in SDN network and traditional network architectures. However, this method uses pure random path selection, which may result in repeated use of paths. Jafarian et al. [5] used game theory and SMT to select the hopping path according to the current state of the communication network and the attack methods used by the attacker, thereby improving the effectiveness of resisting attacks and effectively enhancing the security of static network communication. Zhao et al. [6] proposed a double hopping communication (DHC) scheme based on SDN to resist eavesdropping attacks by expanding the detection space. This scheme coordinates the hopping of multiple attributes such as IP address, communication path, and port number based on the SDN network architecture, providing diversity and uncertainty of network attributes and increasing the difficulty and cost to attackers. However, this method does not consider the situation of key nodes, leading to the repeated appearance of key nodes in multiple hopping cycles. Liu et al. [7] proposed a path random hopping method that is automatically triggered by setting a traffic threshold to change the fixed hopping cycle defect in traditional hopping design. An improved ant colony algorithm was also designed to calculate the optimal hopping path. This method effectively resists the detection of attackers and enhances the unpredictability of system attributes while reducing consumption. Zhang et al. [8] proposed a technology of coordinated hopping of transmission paths and end addresses to resist global eavesdropping attacks. This technology constrains the overlap and capacity of paths. Through SMT constraint solving, it realizes multi-path and multi-attribute coordinated hopping according to a specific hopping cycle, increases the complexity of network attributes, and effectively improves its defense capability, but the complexity is high in the process of path solving. Based on this, Chen et al. [9] designed a software-defined intranet dynamic defense

system (SIDD) with the aim of increasing the difficulty of attacks by attackers. This scheme realizes the coordinated hopping of IP addresses and paths based on zero trust and isolates dynamic design ideas, achieving comprehensive protection of the network. Zkik et al. [10] designed a new architecture to resist eavesdropping attacks by assigning weights to each path and using this as the basis for transmission rate and selection probability. This method has, to some extent, increased the dispersion of data packets and effectively avoided the acquisition of effective key information by attackers during long-term monitoring. Wang Shaolei [11] proposed a new expanded route randomization (ERR) technology based on SDN. This technology uses an improved Floyd–Warshell algorithm to calculate all shortest paths between two points. Furthermore, in the process of executing path hopping, the IP address port number and path of the routing interface hop in coordination, which expands the hopping space to a certain extent and increases the complexity of hopping. However, this method only calculates the shortest path between two points and uses a pure random path selection method, does not consider factors such as key nodes, narrows the path hopping space, and reduces the unpredictability of hopping.

In terms of path hopping selection, existing research mostly has randomness and insufficient constraints, making it difficult to obtain larger benefits. For this reason, Lei Cheng et al. [12] proposed a network moving target defense technique based on optimal forwarding path migration (OFPM), which aims to maximize the defense benefits and solve transient problems that occur when paths are hopping. This technique combines SMT to constrain the capacity, delay, and accessibility of the transmission path; establishes a safety capacity matrix based on the maximum flow–minimum cut idea; and selects the hopping cycle and hopping path based on this. This method not only improves defense benefits but also ensures the effectiveness of path selection. Zhou et al. [13] proposed a spatio-temporal random optimization method for network routing mutation based on multi-objective decision (SSO-RM). This method combines the maximum ability of the network to resist attacks and the ability of the attacker to learn, constructs the problem constraints into a random model, and analyzes the effectiveness of the attack from time and space. Tan et al. [14] divided a large complex network into local networks, then used the Floyd–Warshall algorithm to calculate the path hopping space within each region, and finally combined them into the overall hopping space, effectively improving the defense benefits of route hopping. All of these have not fully considered key nodes in the path and the similarity of the path. Zhang BoFeng [15] constrained path delay, similarity, usage timing, and key nodes, calculated path weights based on key nodes, and used this as the basis for hopping path selection, enhancing the diversity and computational difficulty of path hopping to achieve adaptive adjustment of path hopping. However, this method is for eavesdropping attacks and does not consider the situation where the network is subjected to flooding attacks. Zhang et al. [16] proposed an adaptive routing hopping technology based on deep learning. This technology generates a hopping space through SMT constraint solving according to the traffic of the intrusion detection system, the remaining credit of the node, and the quality of service, and iteratively selects the optimal hopping path by self-learning attack characteristics. Xu et al. [17] proposed a method to select the optimal hopping path by Q-learning attack strategy. This method calculates the optimal hopping strategy through the Kolmogorov model, achieving adaptive adjustment of the hopping cycle and learning rate. Hu Ruiqin [2] proposed an adaptive routing hopping technology of a path state matrix. This method constrains link bandwidth, transmission delay, and node overlap, calculates all non-repetitive paths between communication terminals through backtracking, and then builds a state matrix according to the remaining bandwidth and forwarding quantity of the link and calculates the path weight as the basis for path selection; however, this method limits the path hopping space. Li Chaoyang [18] proposed a random routing hopping method that calculates path weights based on node degree centrality, betweenness centrality, and closeness centrality, and adaptively adjusts path selection probability according to network congestion. This method can effectively avoid the repeated

use of key nodes and can adaptively select hopping paths according to the bandwidth of the path, which has a certain effect in resisting flooding attacks.

In summary, researchers have conducted extensive research on path switching, but there are still some shortcomings. Currently, in studies on generating path switching spaces, most methods use SMT constraint solving, which has a high computational time complexity for large-scale and highly complex networks. The time complexity increases exponentially with the addition of nodes, making this method impractical for large-scale networks. Furthermore, existing research either imposes overly stringent constraints on generating path switching spaces or lacks sufficient constraints. Overly stringent constraints can result in a reduced path switching space, leading to some paths being repeatedly used in a short period. Insufficient constraints may render path switching paths unusable or cause key nodes to be repeatedly used in the short term. Lastly, current research lacks a variety of triggering mechanisms for path switching, which limits adaptive adjustments of switching strategies and switching periods for different attacks. This limitation reduces the diversity, unpredictability, and effectiveness of path switching. Building upon these challenges, this paper introduces SDADM. Firstly, it employs DFS to explore all paths between communicating ends. Subsequently, it applies constraints on the remaining path bandwidth, key nodes, path similarity, and selection period to select alternative paths. Finally, it adjusts the selection of switching paths adaptively based on the remaining path bandwidth and key node weights as the probability of path selection.

3. The Overall Structure of SDADM

As shown in Figure 1, the control layer interacts with the forwarding layer through the southbound interface to obtain network topology, link status, and other information from the forwarding layer, thereby achieving centralized control of the forwarding equipment. At the same time, it issues flow tables to the forwarding devices through the southbound interface, making the forwarding devices strictly forward data according to the flow tables.

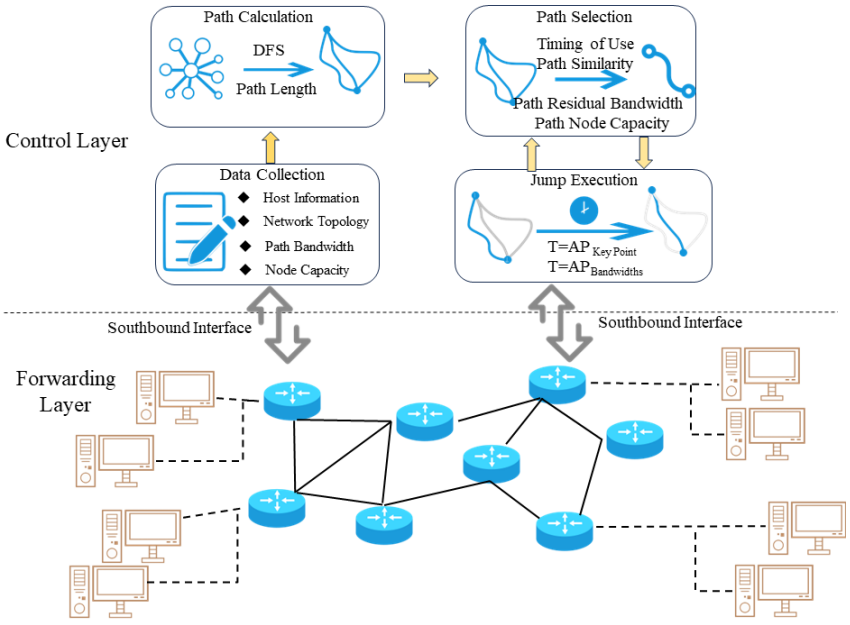

Figure 1. SDADM architecture diagram.

This paper introduces SDADM, leveraging the capabilities of SDN, including features such as separation of control and data plane, centralized control, and programmability. This method mainly includes four modules: data collection, path calculation, path selection, and hopping execution. The data collection module obtains relevant network and communication host data from the forwarding layer and sends it to the path calculation module, which calculates all paths satisfying the constraint conditions between the communication

hosts through DFS traversal, generating a path hopping space. The path calculation module sends the calculated communication paths to the path selection module, which calculates alternative paths satisfying the constraint conditions based on the currently used path, forming an alternative path space. The hopping execution module selects a path from the alternative path space adaptively based on the current network status, performs hopping, and generates corresponding flow table entries. It pre-issues flow table entries to the nodes of the hopping path in a "reverse addition, forward deletion" manner. The design of its functional modules is as follows:

(1) Data collection module: This module is responsible for collecting network status information required for path switching. For example, it gathers information related to communicating hosts, network topology, remaining link bandwidth, node capacity, etc., for further data processing by other modules. Node capacity refers to the available flow table entries in communicating nodes, which can be obtained by sending an OFPPortStatsRequest message to switches to request statistical information and then calculating it. When a host initiates communication and there are no matching flow table entries in the network, the communicating node encapsulates the host's information in a Packet-In message and sends it to the controller. This module analyzes the Packet-In message to extract source and destination addresses and other relevant information. SDN controllers support link layer discovery protocol (LLDP) and broadcast domain discovery protocol (BDDP) for obtaining network topology and link status information [19]. The SDN controller sends LLDP packets to switches using Packet-Out messages, and upon receiving these packets, switches forward them to neighboring devices and send Packet-In messages to the controller. The controller analyzes these messages, saves link discovery records in the link discovery table, and retrieves global network topology and link status information. Controllers typically provide API interfaces for accessing network topology information. In this paper, the Ryu controller is used, and this module utilizes functions like "get_switch", "get_links", and "get_host" from the "ryu.topology.api" module to obtain lists of switches, links, and hosts, thereby constructing a global network topology including hosts.

(2) Path calculation module: This module is mainly used to generate a path hopping space. Based on the network topology and the source–destination host information provided by the data collection module, all paths between the communication hosts are calculated using the DFS method, and paths with a length greater than the constraint conditions are deleted to generate a path hopping space $Space^{S \to D}$.

(3) Path selection module: This module is mainly used to select hopping paths that meet the constraint conditions from the generated path hopping space as an alternative path space. To expand the path hopping space and reduce the complexity of exploring paths, no constraints were applied during the generation of the path hopping space. If used directly, it will cause problems such as repeated use of paths, repeated occurrence of key nodes, and unavailability of paths, reducing the diversity and unpredictability of path hopping. Therefore, this module, based on the path in communication, constrains the paths in the hopping space in terms of similarity, usage time, remaining path bandwidth, and remaining path node capacity. Constraining the similarity and usage time of the path is to avoid the same or similar paths being used repeatedly over multiple hopping cycles, and to enhance the unpredictability of path hopping. Constraining the remaining path bandwidth and the remaining path node capacity is to ensure the availability of the path and to avoid problems such as data loss due to the bandwidth or node capacity not meeting the demand. The weights of the paths in the alternative space are calculated separately from the remaining path bandwidth and key path nodes as the probability of path selection. At the same time, because the current communication path is different, its similar paths and usage time are also different, and the remaining path bandwidth and node capacity are constantly changing, so this hopping space is also dynamically changing.

(4) Hopping execution module: Based on the security status of the network, it adaptively switches between two hopping strategies weighted by key nodes and remaining path bandwidth, dynamically changing the hopping cycle. It strives to select the path that is most divergent from the current path, satisfies the current network status, and has not been used recently.

4. Design of Multi-Constraint Multi-Strategy Hopping Algorithm

4.1. Generation of Path Hopping Space

To maximize the path switching space within limited transmission paths, enhance the diversity and unpredictability of path switching, and reduce the time complexity of path generation, it is necessary to explore as many communication paths between two points as possible. Therefore, in this section, this paper utilizes DFS traversal to explore all paths between two points. Most existing research employs constraint satisfaction methods for solving, such as the approach used by Huriqin [2], where constraints are imposed on link capacity and overlap in the path switching space generation, significantly limiting the potential path switching space. Link capacities are dynamic, and constraints on overlap restrict the diversity of path switching. Constraints related to paths can be addressed during the path selection stage to maintain path switching diversity and unpredictability, aiming to expand the switching space as much as possible. However, while expanding the switching space, it is essential to ensure communication quality requirements. Excessive delays during data transmission can impact user experience and degrade communication quality. Additionally, insufficient remaining node capacity in a path may result in new traffic being unable to match corresponding flow table rules, leading to forwarding failures, packet loss, or service unavailability. Therefore, constraints on transmission delays and remaining node capacities should be considered during the path switching space generation process. Insufficient remaining node capacity in a path can result in new traffic being unable to match corresponding flow table rules, leading to forwarding failures, packet loss, or service unavailability. Therefore, it is necessary to impose constraints on both transmission delays and remaining node capacities during the generation of path switching spaces. The constraint formula is as follows:

$$L^{i}_{S\to D} \le L \tag{1}$$

Since the transmission delay is proportional to the path length, the delay constraint can be expressed in the form of Equation (1), where $L^{i}_{S\to D}$ represents the length of path i, and L represents the maximum allowable path length. Not only does constraining the path length effectively reduce the delay but also, in complex network topologies with the possibility of loops, not constraining the path length may lead to cyclic searches in DFS, increasing the time complexity and performance overhead generated by the discontinuous spatial search.

The formula for constraining the remaining node capacity of a path is shown in Equation (2), where NC_k represents the remaining capacity of node k, and NC represents the node capacity required for data forwarding.

$$NC_k \ge NC \tag{2}$$

During the generation of discontinuous spatial jumps in the path, when the current node expands to its child node, if the child node satisfies the above two constraint conditions, the child node is included in the current path, and the expansion continues from that node as the root node to its child nodes. When the path length reaches L-1, the algorithm checks if the child node of the current node is the destination node; if it is not, the algorithm backtracks to the previous node to avoid unnecessary searches. If the child node of the current node does not meet constraint condition Equation (2), the algorithm backtracks to search other child nodes and removes the current node. During this process, the algorithm also checks if the node is the destination node, and if it is, the search ends. To prevent looping, nodes are marked during traversal. In terms of time complexity, this method imposes

constraints on two factors during traversal using the SMT constraint solving method with a time complexity of $O(n^2)$ [20], where n is the number of nodes. With an increase in the number of nodes and constraints, the time complexity of this method grows exponentially, making it unsuitable for large-scale network applications. The time complexity of DFS traversal is related to the number of nodes and edges, typically $O(n+e)$, where n is the number of nodes and e is the number of edges. In the worst-case scenario, it can be $O(n^2)$. This study improves upon DFS by implementing pruning and node visiting operations, significantly reducing ineffective visits such as looping and further decreasing the time complexity, which is much lower than that of SMT.

4.2. Related Constraints and Weight Calculation

This paper aims to expand the space of path jumps and enhance the diversity of path jumps by using DFS to compute all communication paths between two points without constraining the similarity of paths or key nodes. Consequently, in the space of jumps, it is inevitable that multiple similar paths or paths containing the same nodes will be repeated across multiple jump cycles. These nodes often relay a large amount of data, and when attackers monitor these nodes for extended periods, despite path jumping during the process, there remains a risk of significant data theft, reducing the unpredictability and defensive efficacy of path jumping. Furthermore, in the process of path jumping, the availability of paths should also be considered. While the previous sections focused on path generation, during jumping, it is essential to ensure that the remaining bandwidth of the path meets the data transmission requirements to avoid issues like link congestion and packet loss. To further enhance the diversity, unpredictability, and availability of path jumping, this study imposes constraints on the selected jumping paths in terms of similarity, selection cycles, remaining path bandwidth, and key nodes. This allows the network to adaptively select jumping strategies and paths based on the current network state. Most existing research on path jumping is based on fixed jump cycles, which reduces the diversity and unpredictability of jumps. This study achieves adaptive path jumping strategies while dynamically varying the jump cycles to enhance the diversity and unpredictability of path jumps. The related constraints and weight calculation equations are as follows.

Remaining Path Bandwidth Constraint

When path hopping is performed, in order to ensure the availability of the path and avoid congestion and packet loss, it is necessary to constrain the remaining bandwidth of the path. The equation is as follows:

$$B_{ik} \geq B \tag{3}$$

$$B_{ik} = MIN(b_k(set(link_i^{S \to D}))) \tag{4}$$

The B_{ik} in Equation (3) represents the remaining bandwidth of path i at the k-th hopping cycle, and B represents the bandwidth required for data forwarding. Equation (4) is the calculation method of B_{ik}, where $set(link_i^{S \to D})$ represents all link sets in path i except those directly connected to the source and destination nodes, and b_k represents the remaining bandwidth of each link at the k-th hopping cycle. It can be seen from the equation that the constraint rule of the remaining path bandwidth is to select the minimum value of all link bandwidths on the path at the k-th hopping cycle.

4.3. Path Similarity Constraint

Path similarity is a metric used to compare the similarity between two transmission paths. It can be used to compare path choices between two nodes in a network, and the calculation of transmission path similarity can be based on the proportion of the same part of the path or the same nodes. When executing path hopping, if the choice of the hopping path has a high similarity to the current transmission path, it will reduce the unpredictability of path hopping. To avoid this situation, when selecting a hopping path, this paper uses the Jaccard similarity coefficient to constrain the path, which is a measure used to compare the

similarity between two sets. It measures the ratio between the intersection and the union of two sets. The equation is as follows:

$$J(i,j) = \frac{|i \cap j|}{|i \cup j|} \tag{5}$$

in which i, j are only 0, 1 of the n-dimensional vector according to the path hopping space to build a node matrix L_{m*n} of all paths, m represents all paths of the hopping space, n represents all nodes except the source and destination nodes contained in these paths, nodes in the path are counted as 1, and nodes not in the path are counted as 0. Assuming that M_{01} represents the number of nodes that do not belong to path i but belong to path j, M_{10} represents the number of nodes that belong to path i but do not belong to path j, M_{11} represents the number of nodes that belong to both path i and path j, then Equation (5) can be expressed as

$$J(i,j) = \frac{M_{11}}{M_{10} + M_{01} + M_{11}} \tag{6}$$

According to Equation (6), calculate the similarity coefficient between paths, and create a path similarity matrix J_{m*m} to record the calculation results. From the equation, it can be seen that the similarity coefficient is between 0 and 1, and the smaller the value, the lower the similarity between the two paths and the fewer the number of overlapping nodes. When choosing a path, a threshold can be set for the similarity coefficient as shown in Equation (7), and paths that satisfy the constraint conditions are preferentially selected as alternative hopping paths.

$$J(i,j) \leq \alpha \tag{7}$$

In Equation (7), α serves as the threshold for the path similarity coefficient, used to determine the similarity between alternative paths and the current path. Typically, α is set to 0.5; setting it too high will expand the range of similarity between alternative paths and the current path, leading to a higher similarity between them. Conversely, setting it too low will narrow down the selection range for alternative paths, resulting in fewer alternative paths and reduced diversity in jumps and ultimately affecting the effectiveness of defense strategies.

4.3.1. Selection Cycle Constraint

The path similarity constraint enhances the unpredictability of path hopping but at the same time results in these paths with smaller similarities being repeatedly used over multiple hopping cycles after the constraint. If the attacker listens for a long time, it is easy to crack the hopping rule. Therefore, when selecting a path, it is necessary to constrain the use cycle of the path as shown in Equation (8) and preferentially select those paths that have not been reused or that have never been used within a certain hopping cycle.

$$T_{now} - T_{irecent} \geq \beta \; or \; T_{recent} = 0 \tag{8}$$

where T_{now} represents the current number of cycles for which path jumping is being executed, and T_{recent} represents the number of cycles since alternative path i was last used. The value β represents the threshold for the selection cycle of alternative paths and is used in conjunction with the path similarity constraint. If only the path similarity constraint is used independently, it may result in lower-speed paths being repeatedly used in a short period. On the other hand, if only the selection cycle constraint is used independently, the similarity of alternative paths is not effectively ensured, and there may be a situation where jumping paths are cyclically used within a certain range. The value of β is typically set based on the size of the jump space. A too-high β value will narrow down the selection range of alternative paths, leading to the cyclic use of paths in the overall jump space and reducing the unpredictability of jumps. Conversely, a too-low β value will result in paths being reused too quickly, reducing the effectiveness of defense against attacks.

4.3.2. Hopping Path Weight Calculation

Different network attacks require different network attributes. For eavesdropping attacks, it is generally required that the path can meet the continuity of data forwarding in time and that the scale of forwarding data is large enough in space so that the attacker can analyze and reorganize the data, recover the original information, and achieve the purpose of stealing user information. DDoS (distributed denial of service) and link-flooding attacks including sending a large amount of data to the destination node to consume link bandwidth so that the path has no remaining bandwidth to provide services for normal communication. Due to the finiteness of the hopping path, the intersection between paths is inevitable, so there will be many key nodes, and the amount of data forwarded by these nodes is much larger than that of general nodes. Once attacked, it will cause a large amount of data leakage; even if path hopping is performed, there is still the possibility that the hopping path will pass through this key node. At the same time, when the network is under flooding attack or when the amount of data transmitted is relatively large, the remaining bandwidth of the path becomes very important. Based on this, when selecting a path, the key node and the remaining bandwidth of the path need to be considered. The weight of the key node refers to the calculation of the number of intersecting paths of each node (except the source node and the destination node) in the path, and the path weight is calculated based on the maximum number of node intersections. The remaining bandwidth weight of the path refers to the calculation of the remaining bandwidth of each link in the path, and the path weight is calculated based on the minimum remaining bandwidth of the link. The equations are as shown in Equations (9) and (10):

$$W_1(path_i^{S\to D}) = \frac{\frac{1}{MAX(N^{S\to D}(set(path_i^{S\to D})))}}{\sum_{path_i^{S\to D}\in Space^{S\to D}} \frac{1}{MAX(N^{S\to D}(set(path_i^{S\to D})))}} \tag{9}$$

$$W_2(path_i^{S\to D}) = \frac{MIN(b_k(set(link_i)))}{\sum_{path_i^{S\to D}\in Space^{S\to D}} MIN(b_k(set(link_i^{S\to D})))} \tag{10}$$

In Equation (9), $set(path_i^{S\to D})$ represents the set of nodes after removing the source node and the destination node from path i, $N^{S\to D}$ represents the number of cross paths passing through the node, and $MAX(N^{S\to D}(set(path_i^{S\to D})))$ represents the maximum number of cross paths for nodes other than the source and destination nodes in path i. It can be seen from Equation (9) that the more paths that intersect with the key nodes in the path, the smaller the weight of the path and the lower the probability of being selected. It can be seen from Equation (10) that the larger the remaining bandwidth of the path, the greater the weight of the path and the higher the probability of being selected.

4.4. Hopping Strategy Execution Algorithm

This paper proposes a multi-constraint and multi-strategy path hopping mechanism, which uses the proportion of the remaining bandwidth of the link as the threshold. When the remaining bandwidth of the link is greater than the threshold $(B_{i,j} \geq X)$, it means that the network has not suffered from flooding or DDos attacks, so the hopping strategy is mainly to resist eavesdropping, and the next hopping path is selected based on $W_1(path_i^{S\to D})$ for the next cycle. At the same time, in order to achieve dynamic changes in the hopping cycle, the hopping cycle is combined with the path weight, as shown in Equation (11), where A is a constant, and T is the cycle of executing hopping. When the remaining bandwidth of the link is less than the threshold $(B_{i,j} \leq X)$, it means that the current link is transmitting a large amount of data or is under attack. The purpose of selecting the hopping path based on the path weight above is to avoid key nodes, enhance the diversity and unpredictability of hopping, and effectively resist eavesdropping attacks, but it cannot effectively resist attacks such as flooding. Therefore, this paper designs a multi-strategy hopping mechanism. When the remaining bandwidth of the link is less

than the threshold, in order to effectively resist link-flooding or DDos attacks, the hopping path is selected based on the remaining bandwidth of the link. The greater the remaining bandwidth, the higher the probability of selection, as shown in Equation (12). Because the weight and remaining bandwidth of each path are different, the hopping cycles are also different, and they can switch between the two path selection modes adaptively according to the current network status. Compared with the random selection and fixed hopping cycle of existing research, this method enhances the diversity and unpredictability of hopping and improves the defense capability of the network. The hopping strategy execution algorithm is shown as Algorithm 1.

$$T = A * W_1(path_i^{S \to D}) \tag{11}$$

$$T = A * W_2(path_i^{S \to D}) \tag{12}$$

Algorithm 1 Hopping Strategy Execution Algorithm

Input: $\text{Space}^{S \to D}$, $W_1(path_i^{S \to D})$, $W_2(path_i^{S \to D})$, NC_{ik}, B_{ik}, J_{m*m}, L_{m*n}
Output: $path_i^{S \to D}$

function HoppingStrategy($S, D, W_1, W_2, NC_{ik}, B_{ik}, J, L$)
  Set A, X
  Initial $alternate\_paths = []$, $T_{now} = 0$, $weight1 = weight2 = 0$
  while True **do**
    Update NC_{ik}, B_{ik}
    for $path_i$ in $\text{Space}^{S \to D}$ **do**
      if $J(i,j) \leq a$ and $(T_{now} - T_{recent} \geq \beta$ or $T_{recent} == 0)$ and $B_{ik} \geq B$ **then**
        $alternate\_paths.append(path_i)$
      end if
      Update $W_1(path_i^{S \to D})$, $W_2(path_i^{S \to D})$
      Sleep (T)
    end for
  end while
  if $B_{i,j} \geq X$ **then**
    $T = A * W_1(path_i^{S \to D})$
    $num = random(0,1)$
    for i in $(0, 1, 2, \ldots, len(alternate\_paths))$ **do**
      $weight1 = weight2 + W_1(path_i^{S \to D})$
      if $weight2 < num \leq weight1$ **then**
        $T_{irecent} = T_{now}$
        $path^{S \to D} = path_i^{S \to D}$ **return** $path^{S \to D}$
      else
        $weight2 = weight1$
      end if
    end for
  else
    mark this link is false
    if this link is false **then**
      $T = A * W_2(path_i^{S \to D})$
    end if
  end if
end function

This algorithm first sets the coefficient A, the threshold X, and $B_{i,j}$ as the remaining bandwidth of the current communication path. Select paths that meet the path similarity constraint, selection cycle constraint, path remaining bandwidth constraint, and path remaining node capacity constraint from the path hopping space and store them in the alternative path space. Because the remaining bandwidth and node capacity of paths in the

network are changing in real time and the current communication path used is different, its path similarity constraint is also different, and the alternative path hopping space is also different, so it is periodically updated, as shown in steps 2 to 13. Finally, it determines whether the remaining bandwidth of the current communication path exceeds the threshold. If it exceeds the threshold, it indicates that the network is under attack or the amount of data transmitted is large. At the same time, mark this path as false, and monitor it. As long as this path is false, the weight of the remaining bandwidth of the path is used as the basis for the final hopping path selection. To enhance the unpredictability of path hopping, generate a random number between 0 and 1, accumulate the weights of the paths in the alternative path space, and when the random number is less than or equal to the accumulated weight, select this path as the hopping path and update the use time of this path, as shown in steps 15 to 25. If it does not exceed the threshold, it indicates that the network is communicating normally. At this time, the main focus is on resisting eavesdropping, and the hopping path is selected based on the weight of the key nodes. Repeat the above steps, as shown in steps 26 to 31. This algorithm adaptively switches different object weight hopping strategies according to different network attacks, enhancing the defense while reducing the possibility of similar paths and multi-path intersecting nodes reappearing in multiple cycles. In addition, because of the different weights, the hopping cycle of the path is also dynamically changing, which further enhances the unpredictability of path hopping and increases the difficulty of the attacker's attack.

4.5. Path Hopping Execution Process

As shown in the SDN network topology in Figure 2, H1 and H2 are communication hosts. During the communication process, path hopping is executed, which requires the controller to update the flow table on the switch. This process will cause the problem of inconsistent flow table updates, resulting in data loss. Therefore, this paper adopts the flow table pre-distribution strategy and the flow table update strategy of "reverse addition, sequential deletion". When updating the flow table, the flow table entries are first delivered to the last switch. When deleting, start deleting from the first switch, and the priority of the newly added flow table entries should be higher than the old ones. Assuming that H1 and H2 are communicating, the steps of their hopping execution are as follows:

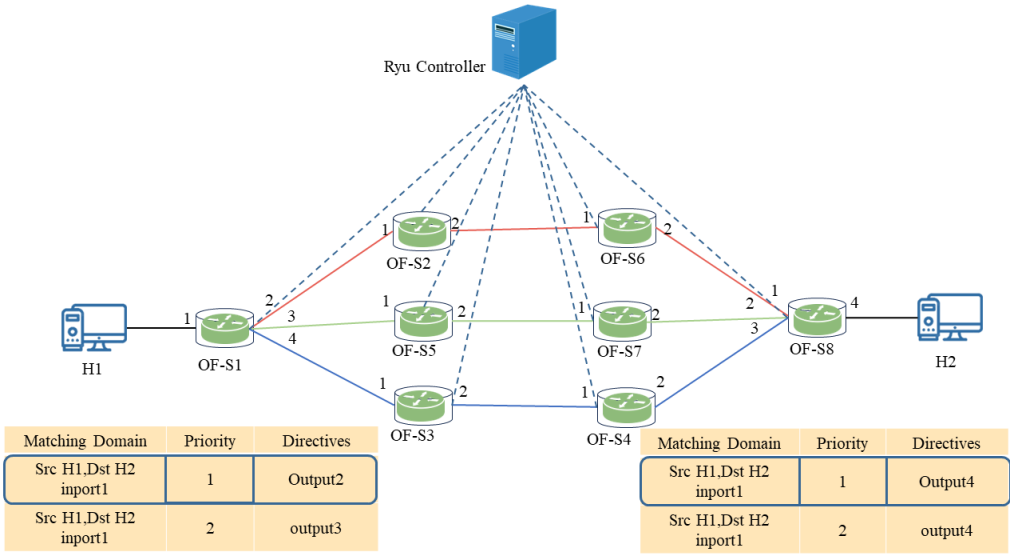

Figure 2. Process of SDADM path hopping.

(1) H1 initiates communication and sends the packet to the switch OF-S1. At this time, there is no corresponding flow table entry on the switch, and the communication information will be encapsulated in the packet_in packet and sent to the controller.

(2) After the controller receives the packet_in packet, the controller calculates all communication paths between H1 and H2 according to the SDADM algorithm, selects a path based on the weight, assumed to be (OF-S1, OF-S2, OF-S6, OF-S8), and adopts the reverse addition method to issue the forward and reverse flow tables, that is, the flow table is issued forward from the switch OF-S8.

(3) When the hopping time T is reached, the controller has re-selected the hopping path according to the SDADM algorithm, assumed to be (OF-S1, OF-S5, OF-S7, OF-S8). The controller issues the flow table in reverse order to the nodes on the path, and the match priority of the flow table is higher than the priority of the path (OF-S1, OF-S2, OF-S6, OF-S8). During the process of issuing the flow table in reverse order, when the flow table is not installed in the intersection node with the previous path (such as OF-S1), the communication data are transmitted according to the original path. When the flow table is issued to OF-S1, the data will be transmitted according to the new path.

When generating the flow table, the survival time hard_timeout and idle_timeout of the flow table will be set. When the time of the flow table exceeds hard_timeout, the flow table entry will be forcibly deleted, generally set to 2 RTT (round-trip time), to ensure that the data on the original communication path can be transmitted normally during path hopping, reducing the packet loss rate. When the flow table entry is not used within the idle_timeout time, it will be assumed that communication between the hosts has stopped, and the controller will end the path hopping and delete the flow table entry, usually set to a round-trip cycle. These two settings ensure normal communication while also reducing the consumption of node capacity and preventing overflow of flow table entries.

5. Experiment and Result Analysis

Resisting eavesdropping attacks and link-flooding attacks are the two main functions of path hopping. An eavesdropping attack is a method of obtaining sensitive data or confidential information by listening to and stealing information during the communication process. Attackers usually collect enough packets in time and space from one or more nodes in the network, analyze them, and reorganize them to obtain key data. A link-flooding attack refers to the attacker sending a large number of fake or deceptive packets to the network link, causing link overload or resource exhaustion, thus affecting the normal function and performance of the network. This paper designs a multi-strategy path hopping active defense method considered from two angles: key nodes and path bandwidth. The weight of key nodes is used as the selection probability, combined with path similarity constraints, to enhance the unpredictability of path hopping to resist eavesdropping attacks. The path bandwidth weight is used as the selection probability to enhance the selection probability of high-bandwidth paths to resist link-flooding attacks.

To verify the effectiveness and performance overhead of the SDADM designed in this paper, this chapter uses Mininet simulation software [21] and creates a network topology as shown in Figure 3 using the Ryu controller as the controller for the entire network. The topology consists of 1 Ryu controller and 12 OpenFlow switches. For the simplicity of the network topology, only two communication hosts, H1 and H2, are labeled in Figure 3, and each switch also connects two communication hosts and communicates, providing a certain amount of data flow for the network to simulate a real network. The configuration information of the experimental environment is shown in Table 1.

Table 1. Experimental environment configuration parameters.

Environment Configuration	Parameters
OS	Ubuntu 16.04.7 LTS
CPU	Intel(R) Xeon(R) E5-26700
RAM	Samsung 16 GByte
Ryu	Ryu 4.34
Mininet	Mininet 2.3.1b1

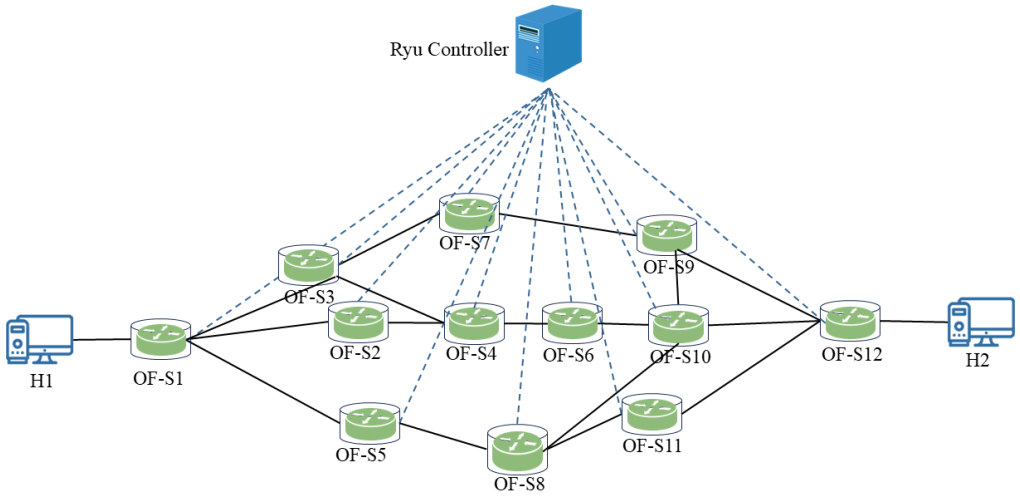

Figure 3. Network topology.

5.1. SDADM Effectiveness Experiment

5.1.1. Diversity of Path Hopping

In the generation of the existing path hopping space, there is a phenomenon that the constraints are too strict, which limits the scale of the path hopping space, reduces the diversity of path hopping, and causes the hopping path to be reused in multiple hopping periods. This section proves the superiority of this scheme in enhancing the diversity of path hopping through the analysis and comparison of the three schemes of a traditional network without path hopping, PSM-ARM (adaptive route mutation based on path state matrix) and the SDADM designed in this paper. The experiment sets α to 0.5, β to 5, A to 10, X to 20%, and sets all the bandwidths in the path to 100 Mb/s. H1 and H2 communicate at a rate of 10Mb/s, the other hosts in the network communicate normally, and the path hopping situations of the three communication modes in multiple hopping periods are compared without considering attacks and bandwidth. The experimental results are shown in Table 2.

Table 2. Experimental results of path hopping diversity.

Hopping Strategy	Hopping Period	Communication Path	Whether the Path Changes	Whether Reused
No path hopping	1	(OF-S1, OF-S3, OF-S7, OF-S9, OF-S12)	No	Yes
PSM-ARM	1	(OF-S1, OF-S4, OF-S8, OF-S11, OF-S12)	Yes	No
	2	(OF-S1, OF-S3, OF-S7, OF-S9, OF-S12)	Yes	No
	3	(OF-S1, OF-S4, OF-S8, OF-S11, OF-S12)	Yes	Yes
	4	(OF-S1, OF-S2, OF-S4, OF-S6, OF-S10, OF-S12)	Yes	No
	5	(OF-S1, OF-S3, OF-S7, OF-S9, OF-S12)	Yes	Yes
SDADM	1	(OF-S1, OF-S5, OF-S8, OF-S11, OF-S12)	Yes	No
	2	(OF-S1, OF-S2, OF-S4, OF-S3, OF-S7, OF-S9, OF-S12)	Yes	No
	3	(OF-S1, OF-S5, OF-S8, OF-S10, OF-S12)	Yes	No
	4	(OF-S1, OF-S3, OF-S7, OF-S9, OF-S12)	Yes	No
	5	(OF-S1, OF-S3, OF-S4, OF-S6, OF-S10, OF-S12)	Yes	No

As can be seen from Table 2, in the traditional mode without path hopping, the communication path is fixed, which is very susceptible to attacks, and attackers can listen to complete data at any node on the path. PSM-ARM pursues the path without any repeated points. Although it effectively avoids key nodes, it reduces the scale of path hopping and the diversity of path hopping. Furthermore, in the process of path selection, there is no constraint on the time of use, so it is very easy to repeat the use of paths in multiple cycle

hops. SDADM calculates all paths between two points, and through the constraints of similarity and usage time, it enhances the diversity and unpredictability of hopping while expanding the path hopping space. Moreover, as the network scale and L value increase, the difference in the number of paths between SDADM and PSM-ARM will be greater, which greatly expands the path hopping space and enhances the diversity of path hopping.

5.1.2. Effectiveness of Resisting Eavesdropping Attacks

Eavesdropping attacks primarily involve acquiring communication data within the network, reassembling and analyzing it, restoring the original data, and preparing for the next step of the attack. Therefore, the amount of data stolen and the success rate of the attack are directly proportional. This section primarily analyzes the effectiveness of SDADM in resisting eavesdropping attacks by comparing the proportions of data packets detected under normal communication conditions between four different communication mechanisms: DHC [22], DPW [18], traditional no-path-hopping, and SDADM. In the simulation experiment parameter settings, the bandwidth of all links is set to 200 Mb/s. The host H1 communicates with H2 at a rate of 20 Mb/s for 5 min, and the rest of the devices communicate normally. In SDADM, the coefficient A is set to 10, α is 0.5, β is set to 5, L is set to 10, and X is set to 20%. During the communication process, all nodes in the simulation topology are monitored through Wireshark, and the amount of data monitored at each node is compared and analyzed to assess the effectiveness of SDADM in resisting eavesdropping attacks. The experimental results are shown in Figure 4.

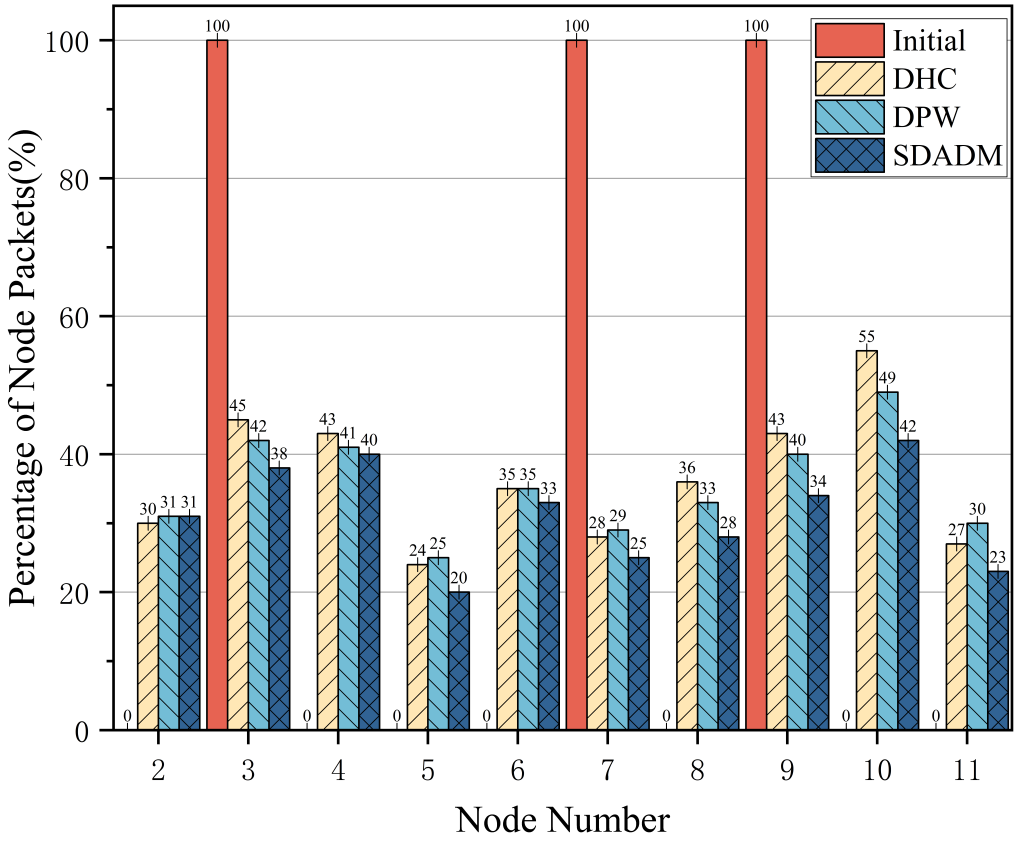

Figure 4. Comparison of effectiveness in resisting eavesdropping attacks.

In Figure 4, the x axis represents the serial number of all nodes in the topology, and the y axis represents the proportion of data packets detected at each node. As can be seen from the figure, in the case of no address hopping in traditional network communication, the communication path is fixed and always remains (OF-S1, OF-S3,OF-S7, OF-S9, OF-S12). This means that attackers can eavesdrop on complete data at any node on this path. The

figure shows that 100% of the data can be detected at OF-S3, OF-S7, and OF-S9. DHC design uses two hopping algorithms. One is random path selection, which has a certain effect on reducing the data monitoring quantity of nodes but will cause some paths to be reused and nodes to reappear, resulting in too much data being forwarded by some nodes. The weighted hopping algorithm of DHC tested in this paper calculates the weight of the path based on the number of cross paths of nodes, which serves as the probability of path selection. The more cross paths, the smaller the probability of being selected, which to some extent reduces the amount of data forwarded by some nodes. However, from the experimental results, there is still a large amount of data on some key nodes. As can be seen from Figure 4, more than 50% of the data is still being forwarded on node 10, and nodes 3, 4, and 9 are also forwarding more than 40% of the data. DPW (Random Routing Defense Method Based On Dynamic Path Weight) enhances constraints on degree centrality, betweenness centrality, and closeness centrality on the basis of DHC to further reduce the probability of key nodes being selected. As can be seen from the figure, this method further reduces the data forwarding volume of node 10 on the basis of DHC, and the data forwarding volume of nodes 3, 4, and 9 also has a slight decrease. Moreover, this method implements the automatic adjustment of path weights according to network conditions, enhancing the ability to resist different attacks. In the process of path selection for SDADM, constraints are applied to key nodes and path similarity at the same time, reducing the probability of key nodes being reused and also reducing the possibility of the same or similar paths appearing repeatedly, making the data more dispersed, effectively alleviating the problem of too much data being transmitted by some nodes. As can be seen from Figure 4, the amount of data forwarded by key nodes has been significantly reduced, and for nodes with many cross paths like node 10, the amount of data forwarded is significantly lower than the previous two methods. This method makes the data in the network more dispersed and the changes in data volume between nodes more stable. As can be seen from the figure, in the network, except for node 10, the amount of data detected by all other nodes will not exceed 40%, and the effect of resisting attacks is significantly better than DHC and DPW and is more secure.

5.1.3. Effectiveness of Resisting Link-Flooding Attacks

Link-flooding attacks aim to cause link overload, making it impossible for legitimate network traffic to pass normally, thereby affecting the normal operation of the network. In the experiment, Hping3 [23] is used to send UDP packets at a rate of 85 Mb/S to the two links OF-S3-OF-S7 and OF-S7-OF-S9 to simulate flooding attacks lasting for 10 min. The bandwidth of all links is set to 100 Mb/s, and the host H1 sends data packets to H2 at a rate of 10 Mb/s, with all other hosts suspending communication to ensure that only H1 and H2 are communicating in the network. By comparing the remaining bandwidth of the path at different communication times under the four communication mechanisms of no address hopping, DHC [22], dynamic routing random hopping [15], and SDADM, the effectiveness of SDADM in resisting flooding attacks is analyzed. The experimental results are shown in Figure 5.

In Figure 5, the x axis represents the communication time of the host, and the y axis represents the remaining bandwidth of the communication path at that moment. As can be seen from the figure, when there is no path hopping in traditional networks, the communication path remains unchanged. When facing link-flooding attacks, it cannot effectively defend. During the host's communication time, the remaining bandwidth of the link is always 5%. It is conceivable that when the attacker increases the attack flow, the bandwidth of this path will be exhausted and unable to provide normal services. DHC uses the method of weighted paths for path hopping, taking the number of cross paths of nodes as weights as the probability of path selection, which to some extent alleviates the impact of link-flooding attacks. However, this method of selecting a path based on key nodes does not consider the factor of path bandwidth and uses a random selection method. Therefore, paths containing attack links will appear in the process of multiple

hops, and the link-flooding attack has not been completely avoided. As shown in the figure, at times 2, 4, and 8, the remaining bandwidth of the path is still 5%, and there is still a risk of being attacked and unable to provide normal services. Dynamic routing random hopping is similar to DHC, with key nodes as constraints. The more cross paths of key nodes, the smaller the weight and the smaller the probability of being selected. The process of path selection still does not consider the factor of path bandwidth. However, this method introduces constraints on the Jaccard distance of the path and the timing of path use, reducing the probability of path reuse and the occurrence of key nodes to a certain extent. Therefore, compared with DHC, this method has certain advantages in path selection, reducing the probability of selecting paths containing attack links, but it still cannot completely avoid link-flooding attacks. SDADM adopts a dual-strategy path selection algorithm. When a link-flooding attack occurs, when the remaining bandwidth of the current communication path is lower than the threshold, it will automatically switch the path selection algorithm according to the remaining bandwidth weight of the path as the selection probability, effectively avoiding the path containing the attacked link. The experimental results also prove the effectiveness of this method.

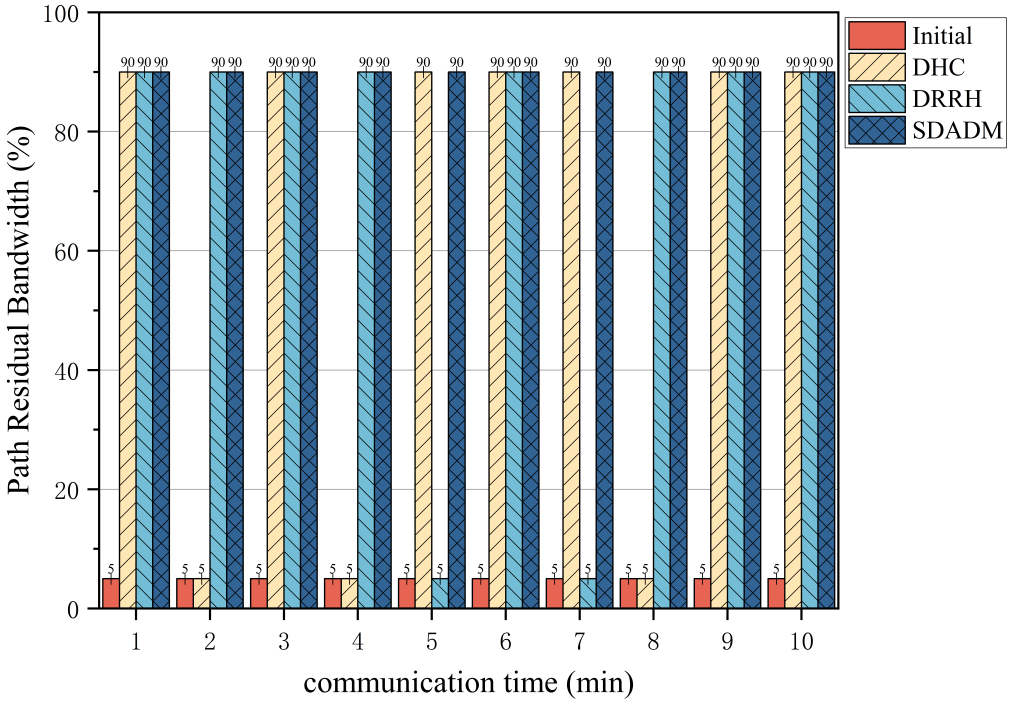

Figure 5. Comparison of effectiveness in resisting link flooding attacks.

5.2. Performance Analysis of SDADM

5.2.1. Generation Duration of Transition Space

In the process of path generation, to solve the problem of the high complexity of SMT constraint solving, this paper proposes a method based on DFS to solve the path between two points. This section analyzes the superiority of the method designed in this paper by comparing the time of generating path transition space under different network topologies. This section selects 6 different complexity network topologies from the article on the topology zoo [24]. The relevant data are shown in Table 3. The corresponding experimental results are shown in Figure 6. As can be seen from the figure, when the network topology is relatively simple, the time of the two methods is basically similar. However, as the complexity of the topology increases, the solution time of SMT grows exponentially; for the algorithm of SDADM designed in this paper, although also gradually increasing, the increase is relatively stable, and the gap between the two is getting bigger and bigger. Moreover, the SDADM algorithm only needs to calculate once at the beginning

of communication when generating the path transition space, while the SMT constraint solving algorithm, due to changes in factors such as path bandwidth and path similarity, needs to recalculate based on the constraint conditions each time the transition is made. Therefore, considering comprehensively, SDADM has more advantages.

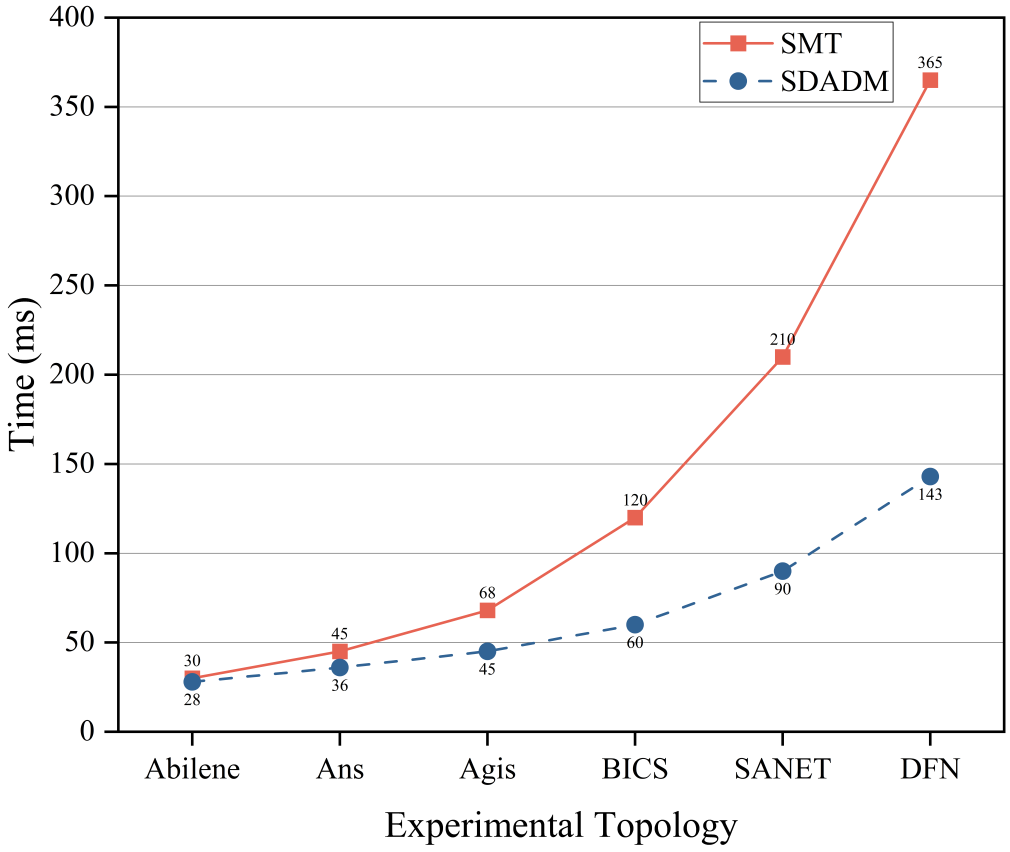

Figure 6. Comparison of path transition space generation time.

Table 3. Experimental topology data.

Topology Name	Abilene	Ans	Agis	BICS	SANET	DFN
Nodes	11	18	25	33	43	58
Links	14	25	30	48	45	87

5.2.2. Data Transmission Delay

When enhancing the path hopping space and security, it will inevitably cause some performance consumption. Among them, the transmission delay of data, as a major evaluation indicator, largely shows the user's experience. This section mainly analyzes the transmission delay of the SDADM hopping strategy by comparing the three path hopping strategies of no path hopping, PSM-ARM, and SDADM under different data transmission rates. In the parameter settings, the path bandwidth is uniformly set to 100 Mb/s, and data are transmitted between H1 and H2 at rates of 20 Mb/s, 40 Mb/s, 60 Mb/s, and 80 Mb/s, respectively, to record and analyze the transmission delay of each path hopping strategy at different transmission rates. The comparative experimental results are shown in Figure 7, where the horizontal axis is the transmission rate, and the vertical axis is the transmission delay. From Figure 7, it can be seen that in the absence of path switching, traditional data transmission follows the shortest path, resulting in lower transmission delays. As the transmission rate increases, the transmission delay also increases, albeit with a small variation, ranging between 0.09 ms and 0.21 ms. On the other hand, the data

transmission paths of PSM-ARM and SDADM are constantly changing, deviating from the traditional shortest path approach, leading to increased transmission delays compared to no path switching. PSM-ARM utilizes backtracking to select path nodes, generating completely unique transmission paths without repetitions, and incorporates dynamic periodic switching. This approach slightly improves transmission delays compared to traditional networks, with delays ranging between 0.09 ms and 0.21 ms as the transmission rate increases.

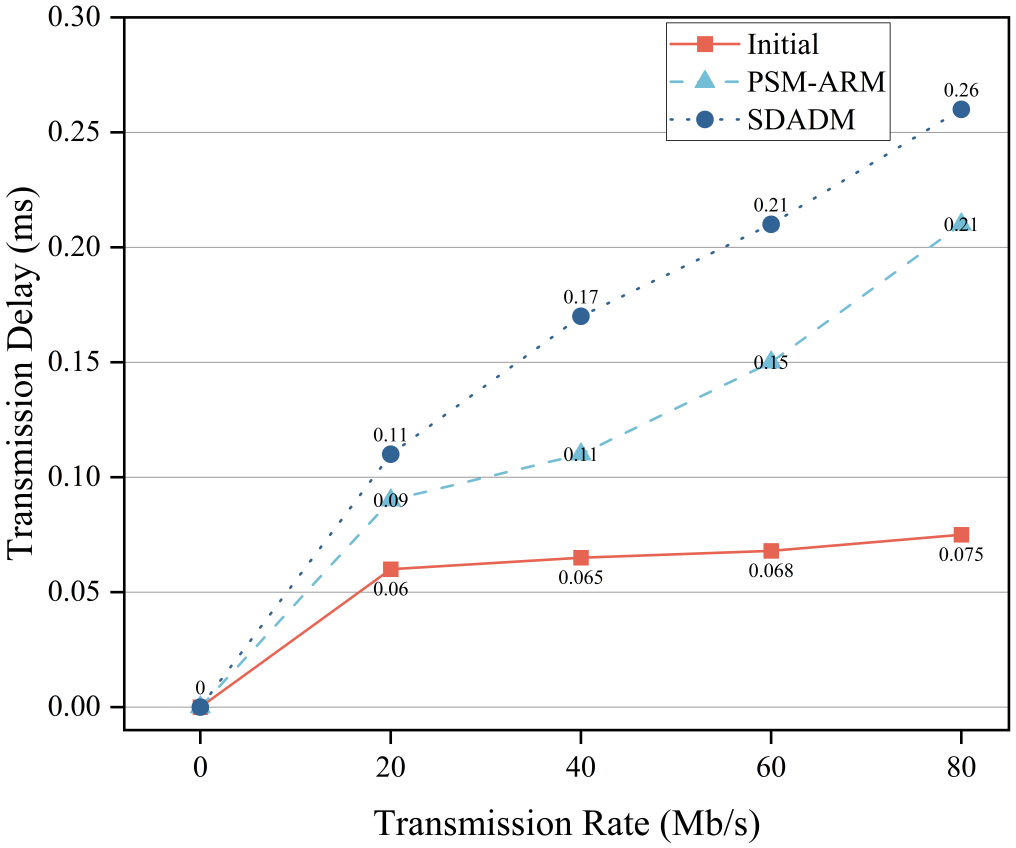

Figure 7. Comparison of delay at different data transmission rates.

The path switching strategy of SDADM, designed in this study, shows further improvements in transmission delays compared to PSM-ARM for several reasons: (1) The utilization of backtracking by PSM-ARM to generate non-repeating paths significantly constrains the path switching space, leading to repetitive path usage over time. In contrast, SDADM expands the path switching space to enhance the diversity and unpredictability of path switching, improving defensive capabilities by selecting all communication paths between hosts that meet the constraints. This may lead to longer paths during switching, causing increased transmission delays. (2) To prevent packet loss due to mishandling during path switching, the strategy in this study involves adding flow table entries in reverse order, affecting data transmission when flow table entries are issued for nodes shared between the switching path and the previous cycle's path, leading to increased delays. (3) SDADM imposes constraints on link remaining bandwidth and critical nodes to better counter eavesdropping and link-flooding attacks, employing two adaptive switching strategies. Switching strategies impact the issuance of flow tables, resulting in increased transmission delays. Despite the increased transmission delays of SDADM compared to PSM-ARM, they remain below 0.3 ms, within an acceptable range throughout data transmission. Experimental results demonstrate that SDADM offers a significantly larger path switching space than PSM-ARM, greatly enhancing the diversity and unpredictability of path switching, confusing attackers effectively, and defending against eavesdropping

and link-flooding attacks. Therefore, the sacrifices made in terms of delays are deemed worthwhile overall.

6. Conclusions

To defend against eavesdropping and link-flooding attacks, this paper proposes an SDN-based multi-constraint and multi-policy path hopping active defense method. Addressing the high time-complexity issue caused by using SMT constraint solving methods in path generation and the problem of overly strict or insufficient constraints in path switching space generation in current research, this study suggests using a DFS algorithm to compute all paths between two points as the path switching space. Constraints are then applied to residual bandwidth, path selection periods, and path similarity parameters during path selection to reduce the complexity of path switching space generation while expanding the space, enhancing the diversity and unpredictability of path switching, and increasing the difficulty for attackers. Moreover, existing research often employs a single triggering mechanism for path switching, mostly using fixed switching periods, reducing the unpredictability of path switching and the ineffectiveness against diverse attacks. In this regard, this paper calculates path weights based on residual bandwidth and critical nodes to adaptively switch path switching modes according to different attacks. Path switching periods are dynamically varied based on path weights, increasing the difficulty for attackers and enhancing defense effectiveness. Comparative experiments on the effectiveness and performance overhead of SDADM show significant improvements in path switching diversity and effectiveness against eavesdropping and link-flooding attacks, with a noticeable decrease in path switching space generation time compared to SMT. Although there is a slight increase in data transmission delays compared to PSM-ARM, it remains within an acceptable range. Overall, SDADM, while ensuring performance, effectively defends against attackers by enhancing path switching diversity.

Author Contributions: Conceptualization, H.L. and N.W.; methodology, J.S.; software, B.Z.; validation, B.Z., S.Z. and W.X.; writing—original draft preparation, B.Z.; writing—review and editing, N.W., H.W., and H.L.; visualization, S.Z.; supervision, H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Natural Science Foundation of China (No. 62106053), the Natural Science Foundation of Guangxi Province of China (No. 2024GXNSFAA010242), the Guangxi Education Department Program (No. 2021KY0347), the Doctoral Fund of Guangxi University of Science and Technology (No. XiaoKe Bo19Z33), and the Innovation Project of Guangxi Graduate Education (No. YCSW2023478).

Data Availability Statement: The data are not publicly available due to the sensitive nature.

Acknowledgments: We are extremely grateful to the editors and the anonymous reviewer for their valuable comments and suggestions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Pratyusa, K.; Manadhata; Wing, J.M. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*; Springer: New York, NY, USA, 2011.
2. Hu, R. Study on Key Technologies of Network Layer Mobile Target Defense Based on SDN. Ph.D Thesis, School of Cryptographic Engineering, Strategic Support Force Information Engineering University, Zhengzhou, China, 2022.
3. Dolev, S.; David, S.T. SDN-based private interconnection. In Proceedings of the 2014 IEEE 13th International Symposium on Network Computing and Applications, Cambridge, MA, USA, 21–23 August 2014; pp. 129–136.
4. Duan, Q.; Al-Shaer, E.; Jafarian, H. Efficient random route mutation considering flow and network constraints. In Proceedings of the 2013 IEEE Conference on Communications and Network Security (CNS), National Harbor, MD, USA, 14–16 October 2013; pp. 260–268.
5. Jafarian, J.H.; Al-Shaer, E.; Duan, Q. Formal approach for route agility against persistent attackers. In Proceedings of the 18th European Symposium on Research in Computer Security, Egham, UK, 9–13 September 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 237–254.

6. Zhao, Z.; Gong, D.; Lu, B.; Liu, F.; Zhang, C. SDN-based double hopping communication against sniffer attack. *Math. Probl. Eng.* **2016**, *2016*, 8927169. [CrossRef]
7. Liu, J.; Zhang, H.; Guo, Z. A defense mechanism of random routing mutation in SDN. *IEICE Trans. Inf. Syst.* **2017**, *100*, 1046–1054. [CrossRef]
8. Zhang, L.; Wei, Q.; Tang, X.; Fang, J. Active Defense Technology for SDN Network Based on Path and End Address Hopping. *Comput. Res. Dev.* **2017**, *54*, 2748–2758.
9. Chen, Y.; Hu, H.; Cheng, G. The design and implementation of a software-defined intranet dynamic defense system. *Acta Electon. Sin.* **2018**, *46*, 2604.
10. Karim, Z.K.I.K.; Sebbar, A.; Baddi, Y.; Boulmalf, M. Secure multipath mutation SMPM in moving target defense based on SDN. *Proc. Comput. Sci.* **2019**, *151*, 977–984.
11. Wang, S. Research on Key Technologies of Dynamic Target Defense Network Based on SDN. Ph.D Thesis, National University of Defense Technology, Changsha, China, 2019.
12. Lei, C.; Ma, D.; Zhang, H.; Yang, Y.J. Network Mobile Target Defense Technology Based on Optimal Path Hopping. *J. Commun.* **2017**, *42*, 3249–3262.
13. Zhou, Z.; Xu, C.; Kuang, X.; Zhang, T.; Sun, L. An efficient and agile spatio-temporal route mutation moving target defense mechanism. In Proceedings of the 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
14. Tan, H.; Tang, C.; Zhang, C.; Wang, S. Area-Dividing route mutation in moving target defense based on SDN. In Proceedings of the 11th International Conference on Network and System Security, Helsinki, Finland, 21–23 August 2017; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 565–574.
15. Zhang, B. Research on Network Layer Mobile Target Defense in SDN Environment. Ph.D Thesis, Tianjin University of Technology, Tianjin, China, 2022.
16. Zhang, T.; Xu, C.; Zhang, B.; Kuang, X.; Wang, Y.; Yang, S.; Muntean, G.M. DQ-RM: Deep reinforcement learning-based route mutation scheme for multimedia services. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 291–296.
17. Xu, C.; Zhang, T.; Kuang, X.; Zhou, Z.; Yu, S. Context-aware adaptive route mutation scheme: A reinforcement learning approach. *IEEE Internet Things J.* **2021**, *8*, 13528–13541. [CrossRef]
18. Li, C. Design and Implementation of Network Hopping Strategy Based on SDN. Ph.D Thesis, Zhengzhou University, Zhengzhou, China, 2022.
19. Mehraban, S.; Yadav, R.K. Traffic engineering and quality of service in hybrid software defined networks. *China Commun.* **2024**, *21*, 96–121. [CrossRef]
20. Rauf, U.; Gillani, F.; Al-Shaer, E.; Halappanavar, M.; Chatterjee, S.; Oehmen, C. Formal approach for resilient reachability based on end-system route agility. In Proceedings of the 2016 ACM Workshop on Moving Target Defense, Vienna, Austria, 24 October 2016; pp. 117–127.
21. Li, Y.; Hao, Z.; Li, N.; Lu, Y. Research on SDN Architecture Simulation Based on Mininet. *Comput. Netw.* **2014**, *40*, 57–59.
22. Zhao, Z. Research on Key Technologies of Mobile Target Defense Based on Software-Defined Network. Ph.D Thesis, PLA Information Engineering University, Zhengzhou, China, 2017.
23. Ye, J.; Cheng, X.; Zhu, J.; Feng, L.; Song, L. A DDoS attack detection method based on SVM in software defined network. *Secur. Commun. Netw.* **2018**, *2018*, 9804061. [CrossRef]
24. Knight, S.; Nguyen, H.X.; Falkner, N.; Bowden, R.; Roughan, M. The internet topology zoo. *IEEE J. Sel. Areas Commun.* **2011**, *29*, 1765–1775. [CrossRef]