Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.039985
Article

Dynamic Security SFC Branching Path Selection Using Deep Reinforcement Learning

Shuangxing Deng, Man Li* and Huachun Zhou

School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, 100044, China
*Corresponding Author: Man Li. Email: 20111018@bjtu.edu.cn
Received: 27 February 2023; Accepted: 12 June 2023; Published: 11 September 2023

Abstract: Security service function chaining (SFC) based on software-defined networking (SDN) and network function virtualization (NFV) technology allows traffic to be forwarded sequentially among different security service functions to achieve a combination of security functions. Security SFC can be deployed according to requirements, but the current SFC is not flexible enough and lacks an effective feedback mechanism. The SFC is not traffic aware and the changes of traffic may cause the previously deployed security SFC to be invalid. How to establish a closed-loop mechanism to enhance the adaptive capability of the security SFC to malicious traffic has become an important issue. Our contribution is threefold. First, we propose a secure SFC path selection framework. The framework can accept the feedback results of traffic and security service functions in SFC, and dynamically select the optimal path for SFC based on the feedback results. It also realizes the automatic deployment of paths, forming a complete closed loop. Second, we expand the protocol of SFC to realize the security SFC with branching path, which improve flexibility of security SFC. Third, we propose a deep reinforcement learning-based dynamic path selection method for security SFC. It infers the optimal branching path by analyzing feedback from the security SFC. We have experimented with Distributed Denial of Service (DDoS) attack detection modules as security service functions. Experimental results show that our proposed method can dynamically select the optimal branching path for a security SFC based on traffic features and the state of the SFC. And it improves the accuracy of the overall malicious traffic detection of the security SFC and significantly reduces the latency and overall load of the SFC.

Keywords: Service function chaining; deep reinforcement learning; security service

1 Introduction

With the rapid development of mobile Internet, the explosive growth of network users and services has caused the network traffic to rise and the threat of network security to become increasingly

serious. Currently, network operators use a large number of dedicated hardware to provide security functions, including deep packet inspection, firewalls and intrusion detection systems, etc. The dedicated equipment is expensive, and the network framework lacks flexibility, manageability and scalability. The paths between security function nodes are inflexible and lack dynamic adaptation to network changes [1].

The emergence of SDN and NFV has changed the implementation method of network security functions and provided a new development direction to solve the above problems [2]. Network function virtualization separates network security functions from dedicated devices and deploys them on universal hardware devices through virtualization. Traditional hardware devices have fixed physical locations, while virtual network functions (VNF) can be flexibly deployed on any device with sufficient resources. Traffic is directed in an orderly manner through various network security functions based on SDN technology, which constitutes a security service function chaining. As shown in Fig. 1, SFC can provide services by combining different security functions according to requirements, which improves the flexibility of security functions [3].

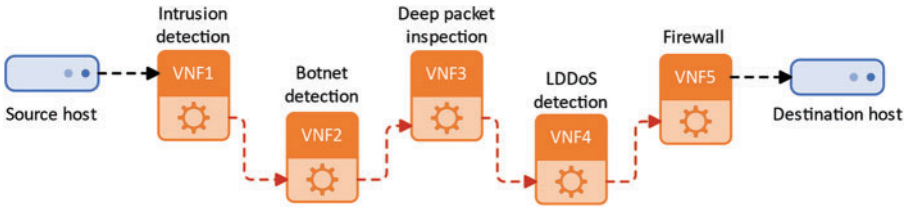

Figure 1: Security service function chaining

DDoS attacks are one of the most common malicious behaviors and one of the main threats to networks due to the low cost and effectiveness. With the increase in available bandwidth, the number of large-scale DDoS attacks has increased dramatically and the capability of DDoS attackers continues to improve. With the development of network devices, various kinds of new DDoS attacks are emerging [4]. SFC can combine multiple types of DDoS attack detection modules to defend against various DDoS attacks and improve the security of the network.

In the service function chaining, traffic traverses all security service functions based on predefined paths. To ensure security, traffic needs to be inspected by all security service functions, which leads to significant resource waste and increases latency. The security SFC path lacks an effective dynamic feedback mechanism. And it cannot dynamically adjust the path based on traffic changes and feedback results from security functions. It lacks adaptive capability and cannot meet the flexible and changing security requirements. To solve the above problems, we propose a deep reinforcement learning-based security SFC branching path selection scheme to dynamically select the optimal path for SFC.

Artificial Intelligence (AI) is now widely used in various fields, including network, novel materials and novel devices [5,6]. Deep reinforcement learning is a major area of focus for AI. The basic concept of reinforcement learning is to learn the optimal policy by maximizing the cumulative reward value that the intelligence obtains from the environment [7]. In the SFC path selection problem, there are many different possible paths, and the goal is to select the optimal path from them by exploring the paths. It is shown that the selection of the optimal path and reinforcement learning are a good fit. Moreover, deep reinforcement learning solves the problem of large-scale input data and is more suitable to be applied to SFC path selection. Compared with deep learning methods, the biggest advantage of deep reinforcement learning is that it does not require manual design of alternative paths. In the case of

a large number of security service functions, it is very difficult to design suitable alternative paths. Therefore, we use deep reinforcement learning algorithm for security SFC path selection.

In this paper, we first propose a security SFC path selection framework. The framework has an effective closed-loop feedback mechanism, where the statistical features of the entering SFC traffic and the results of each security service function are fed back to the knowledge base in real time. The knowledge base analyzes the feedback information and flexibly and dynamically adjusts the paths of secure SFCs through the controller. Second, we extend the protocol of SFC to implement a branching path, and traffic can avoid passing through unnecessary security service functions. Third, we propose a deep reinforcement learning-based path selection algorithm for security SFC, which is used by the knowledge base to inference the path policy for security SFC and select the optimal branching path for SFC. Because DDoS attack is one of the current highly representative network threats, we have experimented using DDoS detection modules as security functions. The proposed scheme can effectively improve the overall DDoS detection capability and reduce the latency and node load.

The rest of the paper is organized as follows: Section 2 presents related work on the security and path aspects of the service function chaining. Section 3 explains our proposed system model. Section 4 performs experimental validation, and Section 5 concludes the paper.

2 Related Works

The network security service has the characteristics of changing security processing requirements, responding quickly according to the security situation. SFC realizes the combination of security service functions according to security requirements, and achieves the purpose of security protection detection. The dynamic adjustment of the security SFC according to the system state can improve the security capability of the service function chain and avoid the waste of resources caused by unnecessary security functions.

SFC technology is widely used in network security. Shameli-Sendi et al. [8] proposed network security defense patterns that utilizes best practices and recommendations from security experts to effectively select deployment options. It meets the required security features according to various security constraints. Sanz et al. [9] proposed and developed SFCPerf, a framework for automated performance evaluation of SFC. SFCPerf enable repeatable consistency testing and performance comparison of network functions and entire function chaining. Zolotukhin et al. [10] focused on solving the optimal security function chaining problem using reinforcement learning and designed an intelligent defense system as a reinforcement learning agent. It observes the current network state and mitigates threats by redirecting network traffic and reconfiguring virtual security devices. Feng et al. [11] proposed a framework for integrating machine learning with virtualized SFC. They proposed a machine learning-based anomaly detection algorithm to be used as a service policy for SFC classifier, guiding the classifier for fast traffic classification and subsequent attack traffic redirection.

To meet the requirements of highly flexible and fast configuration of service functions, SFC needs to orchestrate paths according to different policies to improve resource utilization. Li et al. [12] used graph neural networks to construct security SFC. The graph neural networks capture the complex relationships between physical links and routing policy paths in the network topology to make predictions about QoS and thus efficiently build a security SFC. Wang et al. [13] constructed a performance and resource-aware scheduling system for SFC, which achieves to guarantee the performance of SFC while avoiding resource idleness. Sun et al. [14] proposed an SFC deployment optimization algorithm to optimize resource consumption and end-to-end latency and improve network performance. Liu et al. [15] proposed an SFC dynamic orchestration framework for deep

reinforcement learning in IoT to solve the deployment problem of VNFs and service paths with optimized end-to-end latency in edge clouds.

Path orchestration takes more into account the overall resource utilization. For each individual SFC, paths also need to be selected based on requirements in order to better combine service functions and improve the capacity of the SFC. Ku et al. [16] proposed a reinforcement learning based algorithm to ensure an efficient SFC environment. It selects an appropriate path for the service function chain from multiple paths by considering the usage of each node and the link bandwidth between nodes. Hantouti et al. [17] discussed the concept of partial symmetry and proposed a novel SFC framework with an abstraction layer that can dynamically create partially or fully symmetric SFCs in multiple domains. Luo et al. [18] merged multiple SFCs into a security service function tree to reduce the requirement for resources during virtual security function assignment. Wang et al. [19] extended the network service header protocol and proposed a partially ordered multipath linking mechanism based on partially ordered SFC for low latency and partially ordered services. Zheng et al. [20] proposed network function parallelism (NFP) that allows multiple service functions to run in parallel. NFP reduces the processing delay from the serially-running service functions.

Deep reinforcement learning is an algorithm that combines deep learning with reinforcement learning to achieve end-to-end learning from perception to action, and has a wide range of applications in SFC. Deep reinforcement learning algorithms include value function-based deep Q-network (DQN) [21] and double deep Q-network (DDQN) [22], and policy-based deep deterministic policy gradient (DDPG) [23], etc. Zhu et al. [24] proposed a centralized training distributed execution architecture to solve the SFC deployment problem in IoT using the DQN model. Khoramnejad et al. [25] studied the service function chain on edge computing servers with partial offloading problem and solved the problem using DDQN algorithm. Liu et al. [26] studied the problem of efficiently embedding SFC in dynamic edge cloud scenarios and proposed a DDPG algorithm for small-scale network topologies to achieve lower latency.

All the above researches do not consider the relationship between SFC path and security capability, and cannot dynamically adjust the path according to the real-time network state and security service function state, and cannot improve the detection performance of security SFC by adjusting the path. In this paper, we use deep reinforcement learning algorithms to dynamically select the optimal path for the security SFC based on the traffic features and the detection results of the security service function, and improve the overall security capability of the security SFC.

3 System Model

In this section, we first propose a deep reinforcement learning-based branching path selection framework for security SFC, and then provide a detailed description of the design of branching paths and the principles of path policy inference in the framework.

3.1 Framework

The deep reinforcement learning-based security SFC branching path selection framework proposed in this paper is shown in Fig. 2. The proposed framework implements a closed-loop feedback mechanism for security SFC. It has the capability to sense the traffic entering the SFC and the state of each security service function in the SFC. And through the analysis of traffic and state, it dynamically selects and deploys the optimal path for SFC to improve the performance of security SFC. The framework mainly includes the security SFC module, SFC monitoring module, knowledge base module, controller module and various interfaces between modules. We use the security SFC

to achieve flexible combination of security service functions, and guide traffic to traverse the security service functions according to the set branching path. Security SFC detects traffic from multiple angles and achieves malicious traffic blocking to meet variable network security requirements. We design the knowledge base module to direct the path selection of the security SFC through deep reinforcement learning algorithms, which is connected to the security SFC through the SFC monitoring module and the controller module. It uses the monitoring information provided by the monitoring module as the basis for path selection, and implements the actual deployment of security SFC branching paths through the controller module. The knowledge base module dynamically selects the best path for the security SFC based on the monitoring information, improves the overall detection capability of the security SFC, and reduces the load of security nodes and the latency of the SFC.

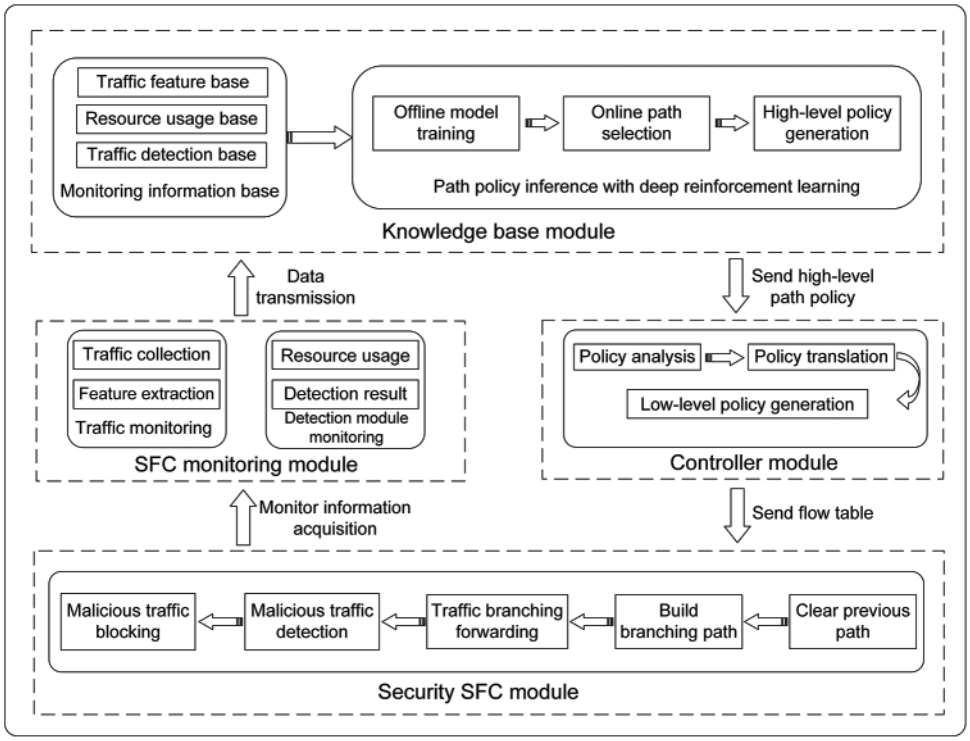

Figure 2: Security SFC branching path selection framework

The security SFC module is the underlying foundation for malicious traffic detection and security protection, and consists of a traffic forwarding submodule and multiple types of security service functions. The traffic forwarding submodule corresponds to the classifier and forwarder in the SFC framework. It receives the flow table sent from the controller module, constructs the corresponding branching path, and realizes the branching forwarding of traffic to make the traffic pass through the required security service functions. The security service functions are deployed on the universal computing platform through NFV technology to process the traffic and meet the security requirements from multiple perspectives. In this paper, a network layer DDoS detection module, a reflection DDoS (DRDoS) detection module, a low-rate DDoS (LDDoS) detection module, an application layer DDoS detection module, and a Botnet detection module are used to combine detection of DDoS attacks [27]. And we block the detected DDoS attack traffic through a firewall. In this paper, the protocol of SFC

is extended to achieve branching path of SFC, which further improves the flexibility of combining the detection modules for each DDoS attack.

The SFC monitoring module includes two parts: traffic monitoring and detection module monitoring. It monitors the operation status of the security SFC in real time and feeds the monitoring information to the knowledge base module in the upper layer as the basis for knowledge base path policy inference. Traffic monitoring refers to collecting all the traffic entering the SFC and extracting the statistical features of the traffic to detect whether the type of traffic entering the SFC has changed. In this paper, we build the security SFC mainly for DDoS attack detection and blocking, so we use the statistical features of the traffic as traffic monitoring information. These features can effectively distinguish normal traffic from each type of DDoS attack traffic. Detection module monitoring refers to real-time monitoring of the feedback results of each DDoS attack detection module and node resource usage status. The detection time of the detection module is affected by the complexity of the model itself, but also by the load of the nodes. And the deployment of detection modules on nodes with high load will lead to an increase in detection time. It is also necessary to consider the resource usage when constructing the SFC path to avoid excessive resource usage, which exceeds the upper limit and causes the service to crash.

The knowledge base module is the core of the entire path selection system. It receives monitoring information from the SFC monitoring module, processes and analyzes this information, and generates the best branching path for the security SFC in the current state. The knowledge base module consists of a monitoring information base and a deep reinforcement learning path policy inference submodule. The monitoring information base receives and processes all kinds of monitoring information from the monitoring module, and stores them in the traffic feature base, resource usage base and traffic detection base, respectively. The path policy inference submodule uses the data in the monitoring information base to train the deep reinforcement learning algorithm. The trained model is used to dynamically select the best branching path for the current security service function chain to improve the overall detection capability of the security SFC and reduce the latency and load.

The controller module implements the best branching path derived by the knowledge base module into the security SFC, enabling traffic to traverse the security service functions according to the new branching path. The controller module contains the security policy controller and the SFC controller. And the security policy controller translates the abstract high-level path policies into low-level path policies. The security policy controller translates the names of security functions into IP addresses and assigns forwarders and corresponding forwarding rules to each security function based on the path information to get the low-level path policy. The SFC controller configures the flow table for the SFC forwarding component according to the low-level path policy, changes the forwarding rules for the traffic, and deploys a new branching path for the security SFC.

The running process of the whole system is as follows:

1. The type of traffic entering the SFC changes, and the previous security SFC path cannot meet the effective and efficient detection of new traffic.

2. The SFC monitoring module monitors the traffic and security service functions in real time, and uploads the new traffic statistical features and the latest feedback results and resource status of each security service function to the knowledge base.

3. The knowledge base uses deep reinforcement learning algorithms for path policy inference based on the latest monitoring information to derive the best branching path policy to cope with new traffic.

4. The controller module receives the high-level path policy from the knowledge base and translates the policy to the low-level path policy. Then the flow table is configured for the SFC according to the low-level path policy, and the best branching path is deployed to the security SFC.

5. The security SFC forwards traffic according to the new branching path, and the SFC has higher detection capability for current traffic under the new path.

3.2 Branching Path

The security SFC branching path proposed in this paper aims to enhance the flexibility of combining security service functions, and avoid traffic passing through unnecessary detection modules, which result in the waste of computational resources and the growth of latency. The original SFC can only classify traffic at a coarse-grained level by means of ingress classifier. The classified traffic is forwarded into a determined SFC, and the traffic must traverse all the service functions in the SFC in turn. In this way, the traffic is not flexible enough to be forwarded, the traffic cannot change the forwarding path based on the feedback results of the security service function. And the traffic may pass through unnecessary service functions for ineffective detection, resulting in wasted resources.

Our proposed security SFC branching path is shown in Fig. 3, where three different types of DDoS attack detection modules are used as security service functions. In the original service SFC, when DRDoS attack traffic enters the SFC, all traffic is forwarded along the main path. All three detection modules detect all traffic and the firewall blocks the attack traffic. Under SFC branching path, the DRDoS detection module forwards the detected DRDoS traffic directly along the branching path to the firewall for blocking. And the rest of the traffic continues to be forwarded along the main path to be detected by other modules. Reference [28] focused on the parallelism and diversions of SFC, we have partially modified it and apply it to path branching security SFC. The branching path can avoid redundant repetitive detection, and can effectively reduce the load of the security service function and avoid the waste of computing resources.

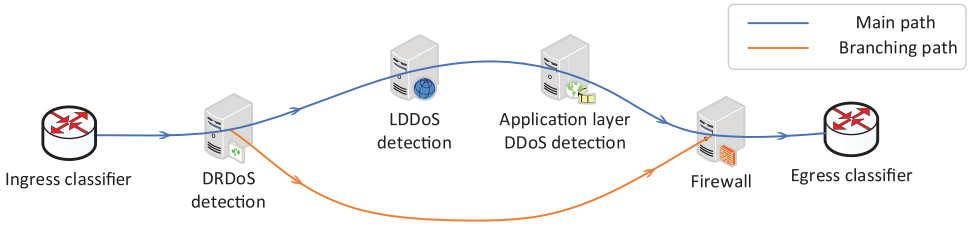

Figure 3: Branching path

The SFC architecture uses the NSH protocol to implement the forwarding of traffic between service functions [29]. The NSH header consists of a basic header for information about the protocol, a service path header for information about the path, and a context header carrying metadata. The service path header is the key to realize the forwarding of traffic within the SFC and contains a service path identifier (SPI) and a service index (SI). The SPI indicates a unique SFC path and the SI indicates the location of the traffic in that path. When the traffic enters the SFC, the NSH header is inserted by the ingress classifier to the packet, setting its SI value to the default 255, and the SI value is subtracted by 1 after the traffic passes through a service function. Combining SPI and SI, the traffic can be forwarded along the SFC path, traversing all the service functions within the path in turn.

We extend the Network Service Header (NSH) protocol and implement SFC branching path using the context header in the NSH protocol. First we propose the concept of division point and aggregation

point, which are both service functions in SFC. Traffic is divided into main and branching path at the division point, and the paths are merged at the aggregation point. Assume that the SPI of the main path and the branching path are SPI_1 and SPI_2, respectively, and the SI at the division point are SI_1 and SI_2, respectively, and the number of service functions between the division point and the aggregation point is n. The NSH header of the traffic before passing through the division point is shown in Fig. 4a. At the division point, the number of service functions n between the bifurcation point and the aggregation point is first obtained according to the path information. Then the SPI_1 and SI_{1-n} of the main path in the NSH header are saved to the context header and the SPI and SI are set to SPI_2 and SI_2, as shown in Fig. 4b. At this time, traffic is forwarded to the aggregation point along the branching path according to the service path header. At the aggregation point, the SPI and SI values of the main path are restored from the context header, and the traffic continues to be forwarded along the main path to achieve the merging of the branching path and the main path, at which moment the NSH header is shown in Fig. 4c.

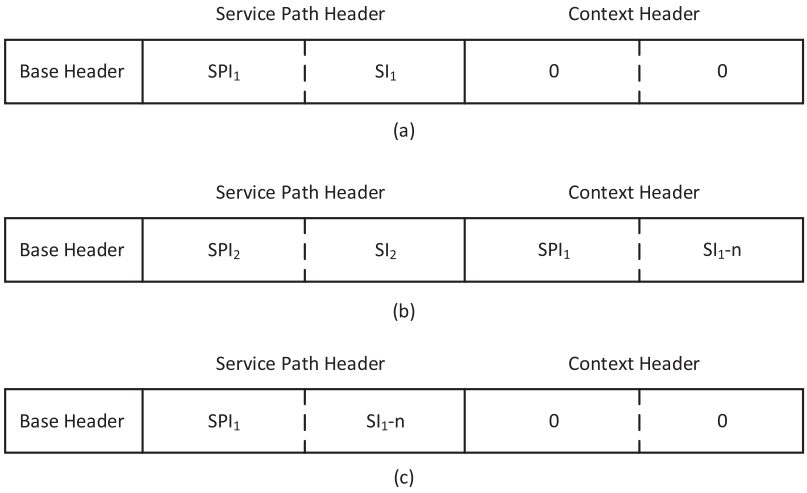

Figure 4: NSH header of branching path

At the division point, the traffic decides whether to forward along the branching path based on the feedback result from the security service function at the division point. In the scenario shown in Fig. 3, the DRDoS attack detection module works as a division point, and traffic detected as DRDoS attack will be forwarded directly to the firewall for blocking along the branching path. The rest of the traffic continues to be forwarded along the main path. In fact, the basis of whether to forward along the branching path is highly scalable and can be set flexibly according to the specific security service functions and security requirements.

3.3 Path Policy Inference

In the knowledge base module, we use deep reinforcement learning algorithms to analyze the data in the monitoring information base and make path policy inference to dynamically select the best branching path for the security SFC. The objective is to improve the overall malicious traffic detection capability and reduce the latency of the SFC and the load on security function nodes.

The monitoring information base consists of a traffic feature base, a traffic detection base, and a resource usage base. In this paper, we use the DDoS attack detection modules as security service functions to construct security SFC for DDoS attack detection and blocking. Therefore, we selected

17 traffic statistical features to form the traffic feature base. As shown in Table 1, the traffic statistics features includes entropy of IP and port [30,31], packets rate [32], conditional entropy of packet [33], etc. Because the traffic rate of DDoS attacks is generally larger compared to normal traffic, we chose packet rate and byte rate as features. When a DDoS attack occurs, there will be a large number of packets whose destination IP is the same attacked host, and by counting the entropy of IP within a time window can effectively distinguish whether a DDoS attack has occurred. Different types of DDoS attacks have different principles and are based on different protocols. By calculating the entropy of port number and the conditional entropy combining the port and IP can effectively distinguish different kinds of DDoS attack traffic. The feedback results of each DDoS attack detection module are recorded in the traffic detection base, as shown in Table 2. These feedback results correspond to the traffic features, which show the detection capability of the detection module for current traffic and provide data support for path policy inference. The resource usage base contains the occupancy of processors, memory and bandwidth of security nodes.

Table 1: Traffic feature base

Feature	Description
Packets rate	Number of packets forwarded per second
Bytes rate	Number of bytes forwarded per second
Packet size	Average packet size
Packet variance	Variance of the number of packets per unit time interval
H (TTL)	Entropy of packet survival time TTL
H (TCP Sport)	Entropy of TCP packet source port
H (TCP Dport)	Entropy of TCP packet destination port
H (UDP Sport)	Entropy of the source port of UDP packets
H (UDP Dport)	Entropy of the UDP packet destination port
H (Packet Size)	Entropy of packet size
H (Sip|Dip)	Conditional Entropy of source ip given destination ip
H (Sip|Dport)	Conditional Entropy of source ip given destination port
H (Dport|Dip)	Conditional Entropy of destination port given destination ip
H (Sip)	Entropy of the source IP address
H (Dip)	Entropy of the destination IP address
H (ΔSip)	Entropy of the change in source IP in the current time window compared to the previous time window
H (ΔDip)	Entropy of the change in destination IP in the current time window compared to the previous time window

Table 2: Traffic detection base

Content	Description
Module name	Name of detection module
Accuracy	Predicting the correct traffic ratio

(Continued)

Table 2 (continued)

Content	Description
Detection rate	Percentage of malicious traffic detected
Detection time	Duration of the detection
Flow	Number of flows in the time window
Malicious flow	Number of malicious flows in the time window
Detected flow	Number of detected flows in the time window
Detection capability	Malicious traffic detection capability
Precision	The percentage of true positives among predicted positives

We constructed a path policy inference sub-module to analyze the monitoring information using the DDQN algorithm [22] to select the optimal main path for the security SFC, and then add branching paths to the optimal main path.

The optimal path is defined as the path with the highest overall detection capability and the lowest SFC latency, while reducing the load on the nodes through branching path. In this paper, we use different types of DDoS attack detection modules as security service functions. The differences between detection modules refer to their different effectiveness in detecting different types of attacks. The differences of each module are reflected in their detection results. The detection results are uploaded to the knowledge base through the monitoring module. The feedback result of each detection module contains the detection capability, which indicates the proportion of malicious traffic detected by the module to the malicious traffic entering the security SFC. As shown in Eq. (1), where A represents the number of real malicious traffic and T represents the number of detected malicious traffic.

$$C = \frac{1}{1 - T/A} \tag{1}$$

The most important factor affecting the SFC latency is the number of detection modules in the security SFC, so we use the number of security service functions to indicate the value of the SFC latency.

The path selection process of the security SFC can be considered as Markov Decision Process (MDP), where the agent does not need an exact mathematical model, but only needs to perform actions based on the state and obtain a reward. The MDP tuple we designed is shown as follows:

State: Different types of attack traffic corresponds to different optimal SFC path. To distinguish the type of traffic, we select features in traffic feature base as the first 17 dimensions of the state as shown in Table 1. In the process of constructing security SFC paths, duplicate detection modules do not improve the overall detection capability and lead to an increase in latency. The last 15 dimensions of the state indicate the detection modules already contained in the current path by different values to avoid duplication of detection modules. Each detection module corresponds to three dimensions, and when the module is not selected, the value of all three dimensions is 0. When the module is selected, the three dimensions used as flags are set to specific values of different orders of magnitude.

Action: The action set contains all the detection modules and the firewall. When the agent executes an action, the detection module corresponding to the action is added to the end of the SFC path.

Because the firewall blocks malicious traffic at the end of the security SFC, the security SFC path construction is completed when the action is a firewall.

Reward: Our proposed system is to select the optimal path for SFC with the highest overall detection capability and the shortest latency. And we use the ratio of the detection capability of the detection module corresponding to the action to the length of the SFC path as the reward. Duplicate detection modules in a security SFC path will not improve the overall detection capability and will lead to increased latency due to additional detection. To avoid the SFC path containing duplicate detection modules, after the reinforcement learning agent selects an action, the environment checks whether the detection module is already present in the path, and if it is, a negative reward of −150 is given. We set the value to −150 because it is numerically closer to the positive reward, which makes it easier for the model to converge in training. When the negative reward is too small, the intelligence will tend to not select the module regardless of whether it is a duplicate or not, and the training will have difficulty converging to the optimal result. And when the negative reward is too large, the path chosen by the intelligence may contain duplicate modules. The reward function is defined as shown in Eq. (2).

$$r_s = \begin{cases} C/L, & \text{a not in path} \\ -150, & \text{a in path} \end{cases} \tag{2}$$

where C is the detection capability shown in Eq. (1) and L is the length of the constructed SFC path, i.e., the number of detection modules included in the constructed SFC path.

For deep reinforcement learning training, after our practical test, the training can be performed for 19 rounds in 1 s. But the time to reselect a path for safety SFC in a real environment is about 2 s, due to the latency generated by the policy translation and clearing of the original path. If the agent interacts with the real environment, it will lead to nearly 20 times increase in training time, and there are risks such as system breakdown affecting the training. Therefore, we train the DDQN path selection model in an offline approach and use the trained model to select the optimal main path for the security SFC. Algorithm 1 describes the offline training process for path selection using the DDQN algorithm.

Algorithm 1: DDQN offline training

Input: Traffic feature set $F = \{f_1, f_2, \ldots, f_n\}$, detection results of each detection module corresponding to the traffic features $D=\{[d_1^1, d_1^2, d_1^3, d_1^4, d_1^5], \ldots, [d_n^1, d_n^2, d_n^3, d_n^4, d_n^5]\}$, initialize the evaluation network $Q(\theta)$, initialize the target network $Q(\theta') = Q(\theta)$, initialize replay memory M

Output: The trained neural network model

1: **for** *episode*=1, ... , N **do**
2: Initialize the path length counter $counter = 0$
3: Initialize *done*=*False*
4: Randomly select set of traffic feature f, initialize state $s = [f, 0, \ldots, 0]$
5: **while** *done*=*False* **do**
6: $counter = counter+1$
7: For the current state s, randomly chose an action with probability ε or choose the action a according to the output of the neural network
8: Modify the last 15 dimensions of the state $s = s'$ according to the action a
9: Calculate the reward r according to Eq. (2)
10: Store transition (s,a,r,s') in M, and sample random minibatch from M
11: Update the evaluation network $Q(\theta)$

(Continued)

Algorithm 1: (continued)

Copy the parameters of the evaluation network $Q(\theta)$ to the target network $Q(\theta')$ each K episode
if $a = exit\ classifier$ **then**
  $done = True$
end if
end while
end for

After the DDQN model selects the optimal main path for the security SFC, it gets the precision of each detection module in the path from the traffic detection base. The precision of the detection module for a specific attack (e.g., DRDoS detection module for DRDoS attack) is used as the basis to determine whether to branch at that module. The precision rate represents the probability that among all the samples predicted to be specific attacks are actually the samples of a specific attack, as shown in Eq. (3). Where true positive (TP) is the number of attack samples that are correctly classified as specific attack traffic and false positive (FP) is the number of other samples that are incorrectly classified as specific attack traffic.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

If the precision is higher than 95%, the specific attack traffic that has been detected is forwarded directly to the firewall for blocking through the branching path based on the detection results. The construction of a branching path based on the precision can avoid traffic from passing through unnecessary detection modules to achieve the reduction of SFC node load. It can also avoid reducing the overall malicious traffic detection rate of the security SFC due to poorly performing detection modules.

4 Experimental Results

4.1 Experimental Environment

We build the system environment shown in Fig. 5 on a server using the VMware vSphere platform. The server CPU model is Intel(R) Xeon(R) CPU E5-2609 v4, 12 virtual machines were used for the experiment, and the system are Ubuntu 15.04 and Ubuntu 18.04. We use OpenDaylight as the SFC manager, Open vSwitch to implement the forwarder and classifier functions, and Docker containers to virtualize each security service function, with security functions deployed decentralized on server nodes.

In our experimental environment, we use network layer DDoS detection module, DRDoS detection module, LDDoS detection module, application layer DDoS detection module, botnet detection module and firewall as security service functions. Detection modules detect DDoS attack traffic, and the firewall blocks malicious traffic. The detection module has optimal detection performance for specific types of DDoS attack traffic and also has generalization to detect multiple types of DDoS attacks. The types of DDoS attacks we send correspond to the detection module, and the specific subclasses of attack traffic are shown in Table 3. We use tools such as hping3 and SlowHttpTest to generate attack traffic in an isolated environment, and use the tcpdump tool to capture and save this traffic. To facilitate the combination of multiple attack traffic, we input the attack traffic into SFC in a replayed manner using the tcpreplay tool. And we collected traffic from multiple scenarios such as browser, video and game in 5G environment as normal traffic.

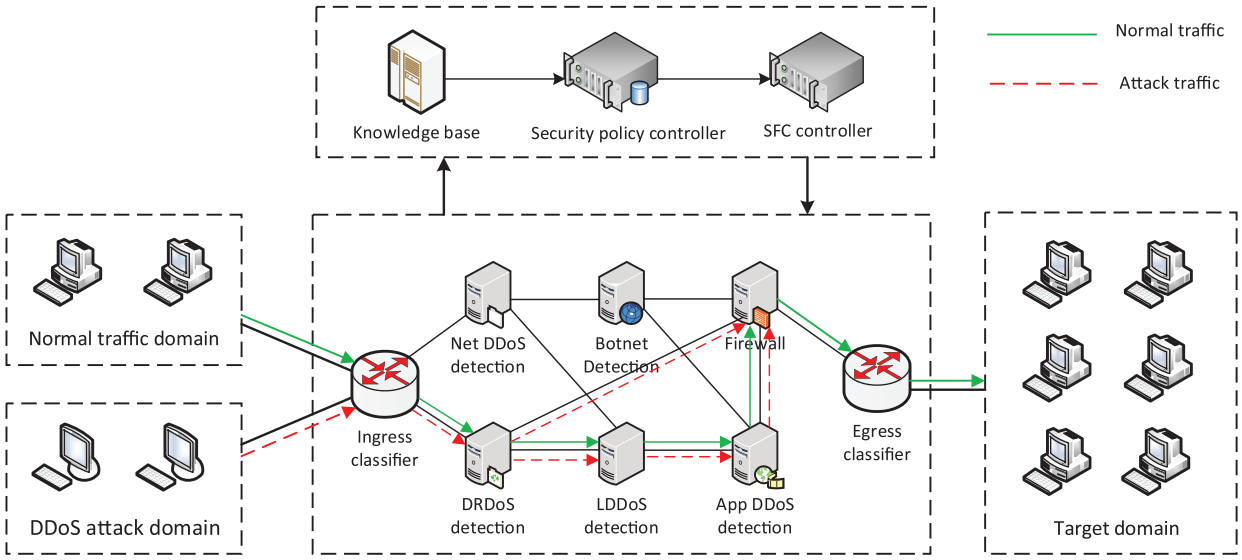

Figure 5: Experimental topology

Table 3: DDoS attack types

DDoS major type	Specific attack type
Network layer DDoS	ACK UDP SYN
DRDoS	TFTP Memcached SSDP NTP Chargen
LDDoS	SlowBody Shrew SlowHeaders Slowread
Application layer DDoS	CC HTTP-Get HTTP-Flood HTTP-Post
Botnet	Ares BYOB Mirai Zeus

4.2 Branching Path Verification

We constructed a branching path security SFC as shown in Fig. 3. The main path includes DRDoS detection module, LDDoS detection module, application layer DDoS detection module and firewall. The branching path uses the DRDoS detection module as a division point to forward the detected DRDoS attacks directly to the firewall for blocking, and the rest of the traffic continues to be forwarded along the main path. We send a mixture of normal traffic and DRDoS attacks into the security SFC to verify the branching path. The division point modifies the NSH header of the packet according to the result, and the forwarder forwards the traffic according to the NSH header and selects the corresponding path for it.

Fig. 6 shows the traffic rates changes in the LDDoS detection module and firewall, after the security SFC path is branched. Before the T1 moment, the branching path is not configured and all traffic is forwarded along the main path in Fig. 3. The traffic rates at the LDDoS detection module and the firewall are approximately the same. After configuring the branching path at the moment of T1, the traffic rate of the LDDoS detection module decreases significantly. It is proved that DRDoS attack traffic is forwarded along the branching path at the branching point and the remaining normal traffic is forwarded on the main path. The traffic rate at the firewall does not change significantly after the

branching, indicating that the branching path and the main path are merged at the aggregation point and all traffic is forwarded to the firewall. The change of traffic rate at the LDDoS detection module and firewall indicates that the branching path is successfully configured. According to the rules, the detected DRDoS traffic is forwarded along the branching path.

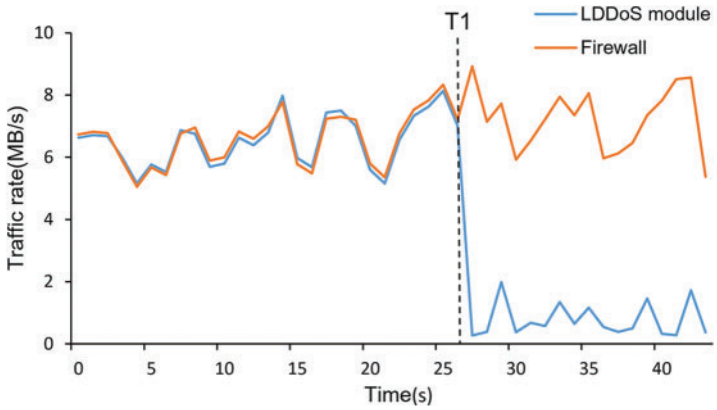

Figure 6: Traffic rate of security function

Fig. 7 shows the latency comparison between the original path and the branching path of the security SFC. The latency of the branching path is reduced by 46% compared to the original path. Under the branching path, the traffic rate of LDDoS detection module and application layer DDoS detection module is significantly reduced, which reduces the queuing delay and time required for packet forwarding and detection, so the branching path has lower latency.

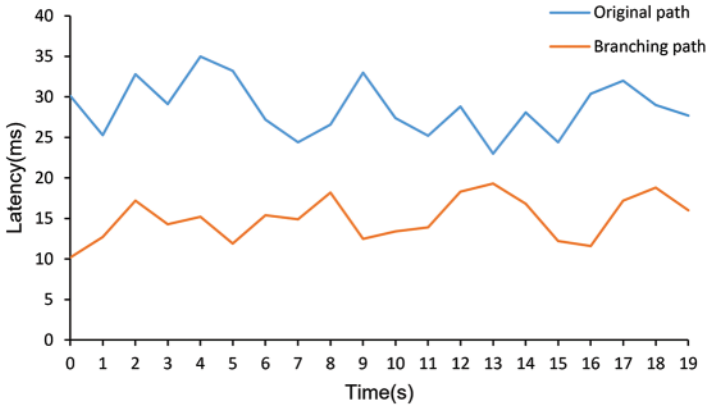

Figure 7: Latency of original and branching path

The above experiments show that the branching path security SFC proposed in this paper can combine security service functions more flexibly, avoid traffic passing through unnecessary detection modules, and avoid wasting resources. As the branching path reduces the traffic forwarded on the main path and reduces the queuing delay of packets forwarded on the main path, configuring the branching path can reduce the overall latency of the security SFC.

4.3 DDQN Path Selection

4.3.1 Offline Training

In the offline training phase, the experiment has been implemented in Python v3.8 and we use pytorch 1.6.0 for deep neural networks. We send DRDoS attack, network layer DDoS attack, LDDoS attack, application layer DDoS attack, botnet attack traffic and mixed traffic of different types of DDoS attacks to SFC. And we use the monitoring module to collect the traffic features and the detection results of each module are saved in the monitoring information base. The data in the monitoring information base is used to train the DDQN model to select the optimal path for the security SFC.

The DDQN model proposed uses a fully connected neural network containing three hidden layers with 300, 240, and 240 neurons. After experimental testing, we determine the parameter settings for the DDQN model with optimal results, as shown in Table 4.

Table 4: DDQN parameter setting

Parameter	Value
Learning rate	0.001
Discount factor	0.9
Exploration degree	0.8
Batch size	128
Experience replay buffer size	5000
Target network update interval	200

We compared the proposed DDQN model with DQN and DDPG and trains 15000 episodes respectively, and the variation of reward value with episodes is shown in Fig. 8. The DDPG algorithm is more suitable for continuous action space due to the gradient descent algorithm to update the policy function, and performs poorly in the discrete action space environment designed in this paper, and cannot converge to the optimal value. Both DDQN algorithm and DQN algorithm can converge to the optimal, DQN algorithm is less stable and converges slowly, and DDQN algorithm has better convergence.

We input random types of attack traffic into the trained model, use different algorithms for path selection, and obtain the performance of the paths according to the selected paths in the monitoring information base. Table 5 shows the results of path selection by the three models DDQN, DQN, and DDPG. Among them, the DDPG algorithm performs poorly and contains more than two duplicate detection modules in the path, which cannot select the optimal path for SFC. The DDQN algorithm has the best performance and is stronger than the DQN algorithm in terms of path length and detection capability, and the path does not contain duplicate detection modules.

4.3.2 Path Comparison

A path containing all detection modules is preset in the prototype, under which all traffic traverses all detection modules in turn, and the path has the ability to detect all types of DDoS attacks. We send botnet traffic, network layer DDoS attack traffic, mixed traffic of DRDoS and LDDoS attacks, and mixed traffic of application layer DDoS and LDDoS attacks to the security SFC separately. And we

use the algorithm proposed in this paper to select the best branching path for the SFC under these traffic flows.

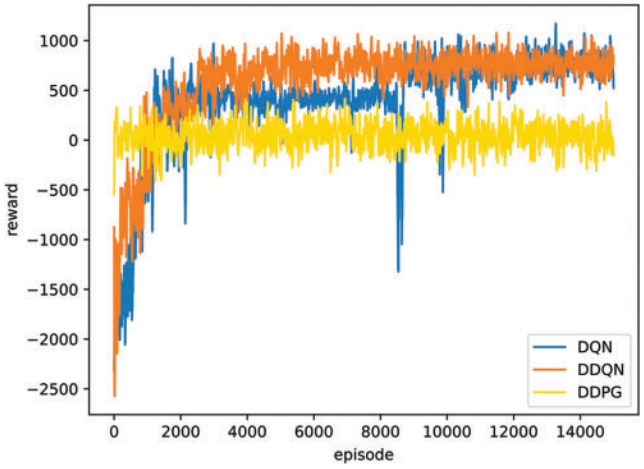

Figure 8: Comparison of reward

Table 5: Path selection results for each algorithm

Algorithm	Average path length	Average detection capability	Number of duplicate detection modules
DDQN	2.7	944.1	0
DQN	3.1	906.6	0.3
DDPG	5.8	355.2	2.6

When the DDoS detection module we use detects traffic, it first divides the traffic into different flows based on IP address and port number and detects whether each flow is malicious or not. For an individual detection module, the accuracy, precision, recall, F1 score, and malicious traffic detection capability shown in Eq. (1) measure the performance of the module in detecting attack traffic. However, the detection results of different detection modules for the same flow may be different, and the detection results of each module in the security SFC are not uniform, and the performance index of the security SFC in detecting malicious traffic cannot be directly statistically measured. Therefore, in this experiment, we make a detection rule for Security SFC, as long as any detection module in the path of Security SFC detects a flow as malicious, the final detection result of Security SFC for that flow is malicious traffic. We statistically measure the final detection results of the flows by Security SFC, and evaluate the overall malicious traffic detection performance of Security SFC by accuracy, precision, recall, and F1 score. The accuracy, recall and F1 score are defined as shown in following equation.

$$A = \frac{TP + TN}{TP + TN + FP + FN}, R = \frac{TP}{TP + FN}, F1 = \frac{2 * P * R}{P + R} \tag{4}$$

where TP represents the number of samples that are correctly classified among the actual type of malicious traffic samples; TN represents the number of samples that are correctly classified by the

model among the actual type of normal traffic samples; FP represents the number of samples that are incorrectly classified as malicious traffic types among the actual type of normal traffic samples; and FN represents the number of samples that are incorrectly classified as normal traffic among the actual type of malicious traffic samples.

Figs. 9–12 show the comparison of malicious traffic detection performance between the preset path and the DDQN selected branching path for four different types of DDoS attack traffic. As shown in Fig. 9, DDQN selected branching path has a higher accuracy rate under each type of DDoS attack traffic, with an average improvement of 4%. This is because the pre-defined path contains all types of detection modules, and there will always be detection modules with poor performance for the incoming attack traffic. The poorly performing detection modules in the path cause a decrease in the overall detection accuracy of the security SFC. In contrast, the path selected by DDQN only includes modules with better performance, so it achieves a higher accuracy. As shown in Fig. 10, the precision of DDQN selected branching path improve by an average of 3.7%. Under DDQN selected branching path, less normal traffic is detected as malicious traffic. As shown in Fig. 11, the two paths are similar in terms of recall, indicating that path selection does not increase the rate of missed detection of malicious traffic. F1 score is a comprehensive reflection of precision and recall, which can evaluate the detection results more comprehensively. Fig. 12 shows that the F1-score of DDQN selected branching path improve by an average of 2.5%. The improvement in F1 scores indicates that the DDQN path selection algorithm improve the overall detection performance of the security SFC.

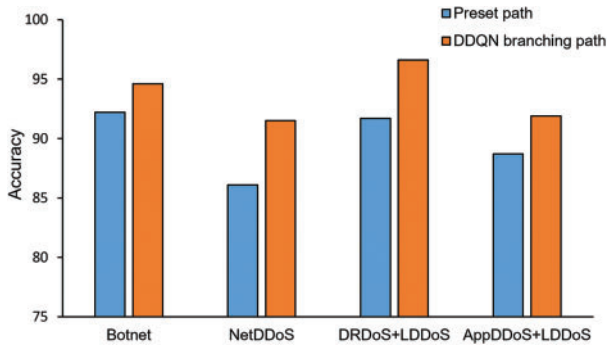

Figure 9: Accuracy of security SFC

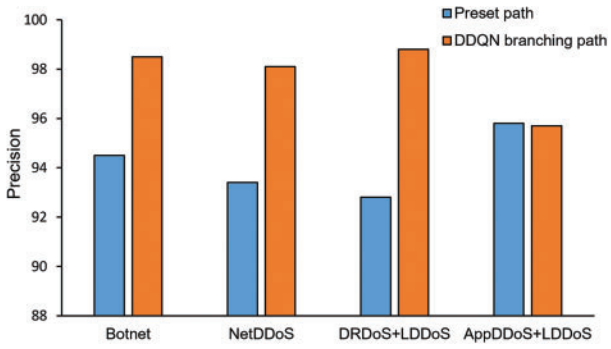

Figure 10: Precision of security SFC

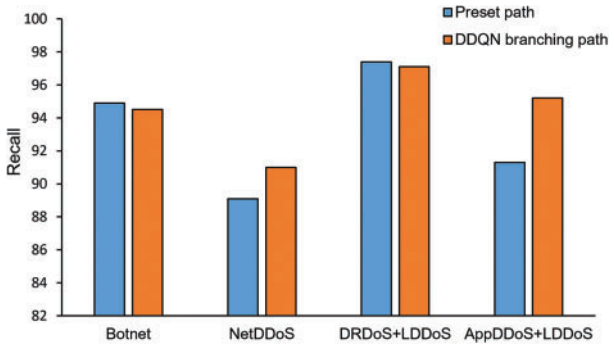

Figure 11: Recall of security SFC

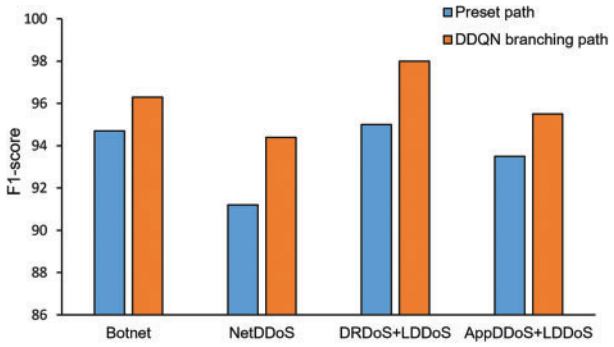

Figure 12: F1-score of security SFC

Fig. 13 shows the average latency comparison between the preset path and the selected branching path of DDQN under attack traffic and normal traffic. When inputting normal traffic, the traffic needs to pass through all detection modules in the preset path. While in the DDQN selection path, the traffic only needs to pass through the firewall. Therefore, the latency of normal traffic in the DDQN selection path is significantly less than the preset path. When inputting the mix of normal traffic and DDoS attack traffic, in the DDQN selected path, normal traffic is forwarded along the main path, and the number of detection modules in the main path is less than the preset path. It reduces the latency caused by forwarding and detection, and some of the attack traffic is forwarded directly to the firewall along the branching path, reducing the queuing latency of the main path traffic. Therefore, under all kinds of traffic, the latency of the path selected by DDQN is significantly lower than the preset path.

Fig. 14 shows the total SFC load for the preset path and the branching path selected by DDQN under DDoS attack traffic and normal traffic. Since all loads of the SFC are generated by forwarding and detecting the traffic, we use the traffic rate sum of all detected modules in the SFC path to approximate the SFC load, which is normalized for presentation purposes. We send botnet traffic into the service function chain at the initial moments and change the input traffic to network layer DDoS traffic, mixed DRDoS and LDDoS traffic, mixed application layer DDoS and LDDoS traffic and normal traffic at moments T1, T2, T3 and T4. After the input traffic changes, our proposed system receives timely feedback and dynamically selects and deploys the optimal path for security SFC. Under the preset path, all traffic traverses each detection module in turn. Under the DDQN selected path, the path contains fewer detection modules, and the traffic is avoided to pass through unnecessary

detection modules by branching the path, so the load of the DDQN selected path is lower than the preset path.

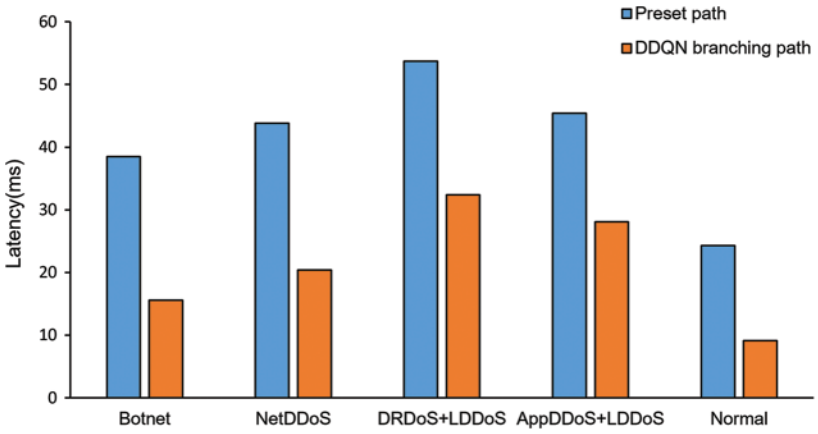

Figure 13: Latency of different SFC paths

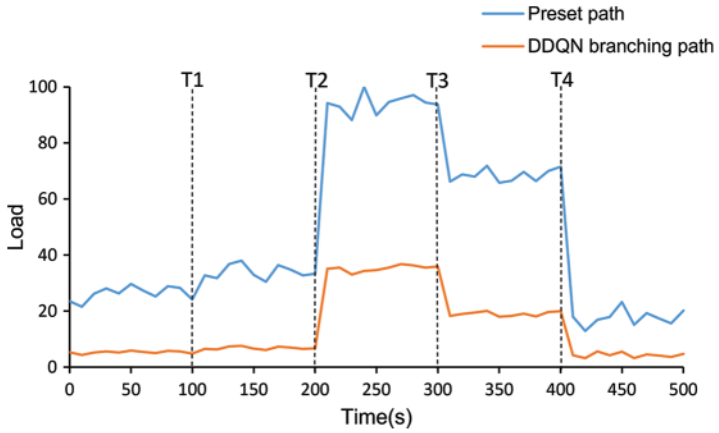

Figure 14: Variation of security SFC load

5 Conclusion

In this paper, we focus on the closed-loop feedback mechanism of security SFC, which makes adaptive and dynamic selection of paths. We first propose a security SFC path selection framework. It dynamically selects and deploys the optimal paths for security SFC based on feedback information. Second, we extend the NSH protocol to implement security SFC with branching path. The branching paths enhance the flexibility of SFC to combine security service functions. We also propose a DDQN-based security SFC path selection algorithm for inferring the optimal path policy. We conducted experiments with the DDoS attack detection module as security service functions. The results show that our proposed branching path selection scheme achieves feedback loop, and improve the overall malicious traffic detection capability and effectively reduce the latency and load of the SFC. In the future, we will focus on building a more advanced knowledge base and guide the optimal branching path selection for security SFC through the knowledge base.

Funding Statement: This paper is supported by NSFC under Grant No. 62341102, and National Key R&D Program of China under Grant No. 2018YFA0701604.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Zhang, Z. Wang, N. Ma, T. Huang and Y. Liu, "Enabling efficient service function chaining by integrating NFV and SDN: Architecture, challenges and opportunities," *IEEE Network*, vol. 32, no. 6, pp. 152–159, 2018.

[2] Q. Duan, A. Nirwan and T. Mehmet, "Software-defined network virtualization: An architectural framework for integrating SDN and NFV for service provisioning in future networks," *IEEE Network*, vol. 30, no. 5, pp. 10–16, 2016.

[3] H. U. Adoga and D. P. Pezaros, "Network function virtualization and service function chaining frameworks: A comprehensive review of requirements, objectives, implementations, and open research challenges," *Future Internet*, vol. 14, no. 2, pp. 59, 2022.

[4] NSFOCUS, *2022 Global DDoS attack landscape report*. Santa Clara, California, USA: NSFOCUS, 2023, 2022. [Online]. Available: https://nsfocusglobal.com/wp-content/uploads/2023/03/2022-Global-DDoS-Attack-Landscape-Report.pdf

[5] Y. Guo, D. Yang, Y. Zhang, L. Wang and K. Wang, "Online estimation of SOH for lithium-ion battery based on SSA-Elman neural network," *Protection and Control of Modern Power Systems*, vol. 7, no. 1, pp. 40, 2022.

[6] M. Zhang, D. Yang, J. Du, H. Sun, L. Li *et al.,* "A review of SOH prediction of li-ion batteries based on data-driven algorithms," *Energies*, vol. 16, no. 7, pp. 3167, 2023.

[7] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[8] A. Shameli-Sendi, Y. Jarraya, M. Pourzandi and M. Cheriet, "Efficient provisioning of security service function chaining using network security defense patterns," *IEEE Transactions on Services Computing*, vol. 12, no. 4, pp. 534–549, 2016.

[9] I. J. Sanz, D. M. F. Mattos and O. C. M. B. Duarte, "SFCPerf: An automatic performance evaluation framework for service function chaining," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symp.*, Taipei, Taiwan, pp. 1–9, 2018.

[10] M. Zolotukhin, P. Kotilainen and T. Hämäläinen, "Intelligent IDS chaining for network attack mitigation in SDN," in *2021 17th Int. Conf. on Mobility, Sensing and Networking (MSN)*, Exeter, UK, pp. 786–791, 2021.

[11] B. Feng, H. Zhou, G. Li, Y. Zhang, K. Sood *et al.,* "Enabling machine learning with service function chaining for security enhancement at 5G edges," *IEEE Network*, vol. 35, no. 5, pp. 196–201, 2021.

[12] W. Li, H. Wang and X. Zhang, "Security service function chain based on graph neural network," *Information*, vol. 13, no. 2, pp. 78, 2022.

[13] J. Wang, H. Qi, K. Li and X. Zhou, "PRSFC-IoT: A performance and resource aware orchestration system of service function chaining for internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1400–1410, 2018.

[14] G. Sun, Z. Xu, H. Yu, X. Chen, V. Chang *et al.,* "Low-latency and resource-efficient service function chaining orchestration in network function virtualization," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5760–5772, 2019.

[15] Y. Liu, H. Lu, X. Li, Y. Zhang, L. Xi *et al.,* "Dynamic service function chain orchestration for NFV/MEC-enabled IoT networks: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7450–7465, 2020.

[16] H. J. Ku, J. Jung and G. I. Kwon, "A study on reinforcement learning based SFC path selection in SDN/NFV," *International Journal of Applied Engineering Research*, vol. 12, no. 12, pp. 3439–3443, 2017.
[17] H. Hantouti, N. Benamar, M. Bagaa and T. Taleb, "Symmetry-aware SFC framework for 5G networks," *IEEE Network*, vol. 35, no. 5, pp. 234–241, 2021.
[18] J. L. Luo, S. Z. Yu and S. J. Peng, "SDN/NFV-based security service function tree for cloud," *IEEE Access*, vol. 8, pp. 38538–38545, 2020.
[19] Y. C. Wang, R. H. Hwang and Y. D. Lin, "Low-latency service chaining with predefined NSH-based multipath across multiple datacenters," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4721–4733, 2022.
[20] D. Zheng, G. Shen, X. Cao and B. Mukherjee, "Towards optimal parallelism-aware service chaining and embedding," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2063–2077, 2022.
[21] I. Osband, C. Blundell, A. Pritzel and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances in Neural Information Processing Systems*, Barcelona, Spain, Curran Associates Inc, pp. 4026–4034, 2016.
[22] H. Van Hasselt, A. Guez and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. of the AAAI Conf. on Artificial Intelligence*, Palo Alto, California, USA, pp. 2094–2100, 2016.
[23] Y. Hou, L. Liu, Q. Wei, X. Xu and C. Chen, "A novel DDPG method with prioritized experience replay," in *2017 IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC)*, Banff, AB, Canada, pp. 316–321, 2017.
[24] Y. Zhu, H. Yao, T. Mai, W. He, N. Zhang *et al.,* "Multiagent reinforcement-learning-aided service function chain deployment for internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15674–15684, 2022.
[25] F. Khoramnejad, R. Joda and M. Erol-Kantarci, "Distributed multi-agent learning for service function chain partial offloading at the edge," in *2021 IEEE Int. Conf. on Communications Workshops (ICC Workshops)*, Montreal, QC, Canada, pp. 1–6, 2021.
[26] Y. Liu, Y. Lu, X. Li, W. Qiao, Z. Li *et al.,* "SFC embedding meets machine learning: Deep reinforcement learning approaches," *IEEE Communications Letters*, vol. 25, no. 6, pp. 1926–1930, 2021.
[27] M. Li, H. Zhou and Y. Qin, "Two-stage intelligent model for detecting malicious DDoS behavior," *Sensors*, vol. 22, no. 7, pp. 2532, 2022.
[28] D. Eastlake, *Service function chaining (SFC) parallelism and diversions*. IETF, 2022. [Online]. Available: https://datatracker.ietf.org/doc/draft-eastlake-sfc-parallel/
[29] P. Quinn, U. Elzur and C. Pignataro, *Network service header (NSH)*. IETF, RFC8300, 2018. [Online]. Available: https://datatracker.ietf.org/doc/rfc8300/
[30] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrão and M. L. Proença Jr, "Network anomaly detection system using genetic algorithm and fuzzy logic," *Expert Systems with Applications*, vol. 92, pp. 390–402, 2018.
[31] M. E. Ahmed, S. Ullah and H. Kim, "Statistical application fingerprinting for DDoS attack mitigation," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 6, pp. 1471–1484, 2018.
[32] P. Du and S. Abe, "Detecting DoS attacks using packet size distribution," in *2007 2nd Bio-Inspired Models of Network, Information and Computing Systems*, Budapest, Hungary, pp. 93–96, 2007.
[33] Y. Gu, K. Li, Z. Guo and Y. Wang, "Semi-supervised K-means DDoS detection method using hybrid feature selection algorithm," *IEEE Access*, vol. 7, pp. 64351–64365, 2019.