Computer Systems Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2024.058327

ARTICLE

An Intelligent Security Service Optimization Method Based on Knowledge Base

Xianju Gao*, Huachun Zhou, Weilin Wang and Jingfu Yan

School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, 100044, China

*Corresponding Author: Xianju Gao. Email: 22120052 @bjtu.edu.cn

Received: 10 September 2024 Accepted: 28 October 2024 Published: 03 January 2025

ABSTRACT

The network security knowledge base standardizes and integrates network security data, providing a reliable foundation for real-time network security protection solutions. However, current research on network security knowledge bases mainly focuses on their construction, while the potential to optimize intelligent security services for real-time network security protection requires further exploration. Therefore, how to effectively utilize the vast amount of historical knowledge in the field of network security and establish a feedback mechanism to update it in real time, thereby enhancing the detection capability of security services against malicious traffic, has become an important issue. Our contribution is fourfold. First, we design a feedback interface to update the knowledge base with information such as features of attack traffic, detection outcomes from network service functions (NSF), and system resource utilization. Second, we introduce a feature selection method that combines PageRank and RandomForest to identify influential features in the knowledge base and dynamically incorporate them into the NSFs. Third, we propose a path selection method that combines graph attention network (GAT) and deep reinforcement learning (DRL) to learn the local knowledge of the knowledge base and determine the optimal traffic path within the Service Function Chains (SFC). Finally, experimental results demonstrate that the knowledge base can be updated in real time according to feedback information, and the optimized service achieves an accuracy, recall, and F1 score exceeding 96%. Compared to preset paths and paths selected using the deep Q-network (DQN) method, our proposed method increases the malicious traffic detection rate by an average of 12.4% and 4.6%, respectively, enhances the total malicious traffic detection capability (TMTDC) of the path by 18.1% and 11.5%, and significantly reduces path detection delay. It has been verified that the proposed intelligent security optimization method can monitor malicious traffic in real time, update knowledge, and enhance the system's detection capability against malicious traffic.

KEYWORDS

Network security knowledge base; feature selection; path selection; knowledge feedback

1 Introduction

With the rapid growth of the Internet's scale and the emergence of new technologies like the Internet of Things and cloud computing, the landscape of network security faces increasingly blurred boundaries. Traditional network security systems struggle to combat the continuous stream of network

threats effectively. In response, researchers have increasingly turned to integrating artificial intelligence algorithms with security services [1], aiming to automate the detection of abnormal traffic. These intelligent network service functions (NSF) are typically delivered to users in the form of Service Function Chains (SFC) [2]. However, attackers continuously update their attack strategies to make the traffic features resemble normal traffic more closely to evade detection. Most existing intelligent security service methods are inflexible and lack dynamic adaptability to changes in the network [3].

The advent of the network security knowledge base standardizes and integrates network security data, enabling continuous storage and updates of information on malicious traffic across highly interconnected terminals. This capability forms a dependable foundation for developing real-time network security protection solutions [4]. The network security knowledge base stores information in the form of knowledge graphs, which depict relationships between entities as a graph. This method serves as a semantic network, offering efficient query and display features, flexible storage, and update capabilities.

However, current research on network security knowledge bases mainly focuses on their construction, while the potential to optimize intelligent security services for real-time network security protection requires further exploration. Therefore, we propose an optimization method of intelligent security services based on the network security knowledge base that solves the three challenges described in the following paragraphs.

The first challenge is how to update the information on malicious traffic in the knowledge base in real-time. To address this, we design a feedback interface that sends information on malicious traffic, including attack traffic features, detection outcomes from NSFs, and system resource utilization, back to the knowledge base for continuous updates.

The second challenge is how to select the features of malicious traffic from the knowledge base for detection. We use a feature selection method that combines PageRank and RandomForest for feature selection. PageRank is used for effective knowledge inference and feature selection in the knowledge graph, while RandomForest is used for selecting information from the feature datasets.

The third challenge is how to select the path of traffic through the SFC using the information in the knowledge base. We use the graph attention network (GAT) to optimize deep reinforcement learning (DRL) to select an optimum detection path. GAT utilizes an attention mechanism, allowing the model to better capture the relational and feature information between data, enabling deep analysis and mining of the data. DRL allows agents to explore the environment and learn from experience without human heuristics.

In this work, our main contributions are as follows:

1. We design a feedback interface to update the knowledge base with information such as attack traffic features, detection outcomes from NSFs, and system resource utilization. This interface also dynamically adjusts the traffic path through the SFC.
2. We introduce a feature selection method that combines PageRank and RandomForest, identifying influential features in the knowledge base and dynamically incorporating them into the NSFs to improve their detection capability.
3. We propose a path selection method that combines GAT and DRL, enabling the system to learn the local knowledge from the knowledge base and determine the optimal traffic path within the SFC.
4. In the experimental environment, we performed offline training and online detection, systematically comparing the performance across different schemes. The results demonstrate

that the optimization method of intelligent security services based on the network security knowledge base in this paper can significantly improve the detection performance and reduce path detection delay.

The remainder of the paper is organized as follows: Section 2 introduces the related work. Section 3 describes the intelligent security services optimization method. Section 4 presents the experimental environment and results. Section 5 concludes this paper.

2 Related Works

The network security knowledge base contains all kinds of malicious traffic information in the network, which can be used as a valuable data source for security incident analysis and provide a reliable defense scheme for the network system. Many researches at home and abroad focus on the network security knowledge base and intelligent security services.

The knowledge base can display data in the form of knowledge graphs. As specific knowledge graphs in the security field, network security knowledge graphs consist of nodes and edges to form a large-scale security semantic network. This provides an intuitive modeling method for various attack and defense scenarios in the real security world. To describe the Distributed Denial of Service (DDoS) attacks comprehensively, Reference [5] constructed a malicious behavior knowledge base of DDoS attacks, which consists of a malicious traffic detection knowledge base and a network security knowledge base. The former is responsible for detecting and classifying malicious traffic generated by DDoS attacks. The network security knowledge base is responsible for data structure processing, malicious behavior knowledge graph creation, and behavior reasoning. Reference [6] established a knowledge base, which mainly includes two parts: knowledge construction and knowledge display. Knowledge construction is responsible for extracting knowledge from structured data sources and unstructured texts, and knowledge display is mainly responsible for the visualization function of network security knowledge. Reference [7] proposed a malicious behavior knowledge base construction model based on a five-element model. First, entities are extracted and constructed by the machine learning method to acquire network security knowledge, and the NER method is used to complete the knowledge extraction method in the field of network security.

In the field of intelligent security services, data transmission over wireless channels necessitates essential security measures to prevent attackers from eavesdropping or manipulating information. In this regard, Reference [8] proposed a key management protocol that uses three auxiliary keys along with a master key to encrypt information at three different levels within the network. Additionally, the detection and defense against DDoS attacks have been a persistent challenge in the realm of network security.

Feature selection is a key aspect for improving the performance of DDoS attacks detection. The existing attack detection work has the following representative works in feature selection. Reference [9] obtained a typical botnet feature set by Boruta feature selection method, but a single feature selection algorithm can't fully mine the correlation between data features and prediction results, which may lead to insufficient feature expression ability and affect the online detection performance. Reference [10] combined various feature selection algorithms, the features selected by three or more feature selection algorithms are important features. The features selected by only two feature selection algorithms are the second most important features. The remaining features form an unimportant feature set. Reference [11] used Pearson moment correlation to analyze the correlation between features and category labels, and used the area under the curve (AUC) measurement to measure the importance

of each feature. Only the first five features are used and good results are obtained. Reference [12] proposed a method called RFUTE based on RandomForest, to deal with the problem of feature selection in Phase Locked Loop (PLL). Firstly, the ambiguity of candidate tags is eliminated, and then the feature selection is carried out by calculating and ranking the total changes of information entropy of all features of all trees in the forest. PageRank, the web ranking algorithm, was created and applied by Google, mainly to evaluate the importance of each node in the network. It is widely used in the network, such as friend recommendations [13], web search [14], and link analysis [15]. However, it has yet to be applied in the field of feature selection.

SFC path selection aims to provide users with high-speed and low-delay diversified network function customization services, which is another key aspect of improving detection performance. Reference [16] proposed an algorithm for constructing the SFC based on a graph neural network. The algorithm uses the representation of nodes in the graph neural network to construct a flexible and efficient SFC under the influence of its neighboring nodes. Reference [17] proposed a dynamic arrangement framework of SFC for DRL of the Internet of Things to solve the problem of optimizing the deployment of Virtual Network Functions (VNF) and service path with end-to-end delay in the edge cloud. Reference [18] proposed a deep Q-network (DQN) path selection method for security SFC. However, its accuracy in detecting malicious traffic needs improvement, and the detection latency needs to be reduced. Reference [19] combined DQN with convolutional neural network (CNN), and proposed an SFC path selection algorithm. The algorithm adopts the CNN approximate Q function of state-action pair, which can extract data features with the convolution layer and pooling layer and provide automatic learning from the data structure. Aiming at the dense cellular network scene with network slices on multiple base stations, Reference [20] combined the GAT with DRL and designed an intelligent real-time inter-chip resource management strategy to cope with frequent base station handovers and meet the fluctuation of different service requirements.

The relevant literature indicates several key issues in current research on knowledge bases and intelligent security services. First, most studies on the network security knowledge base focus primarily on their construction, leaving the potential for optimizing intelligent security services to enable real-time network security protection largely unexplored. Second, the current feature selection methods, which rely solely on feature engineering algorithms like RandomForest, fail to fully capture the importance of data features, resulting in insufficient representation of malicious behavior traits. Moreover, while GAT can capture relational information between data for deep analysis and data mining, and DRL can enhance adaptability in SFC path selection, there is currently no research that integrates these two approaches in the context of SFC path selection.

These gaps have inspired the development of the proposed method, which leverages the rich data resources in knowledge bases to optimize intelligent security services. It combines graph-based feature selection methods called PageRank with traditional feature engineering techniques to more effectively uncover attack features. Additionally, the integration of GAT and DRL into SFC path selection is introduced to enhance the efficiency of SFC path selection, aiming to detect various types of malicious attacks, improve detection accuracy, and reduce detection latency, thereby offering strong practical value.

3 Intelligent Security Service Optimization Method

We propose an intelligent security service framework and divide it into four modules according to their functions. Section 3.1 provides an overview of the entire framework. Section 3.2 delves into the specifics of the data feedback module. Section 3.3 elaborates on the composition of the knowledge

base module. Section 3.4 details the implementation of the NSF feature selection module and the SFC path selection module.

3.1 Framework

The Intelligent security service framework introduced in this study is illustrated in Fig. 1, encompassing four key modules: the data feedback module, knowledge base module, security policy reasoning module, and service function module.

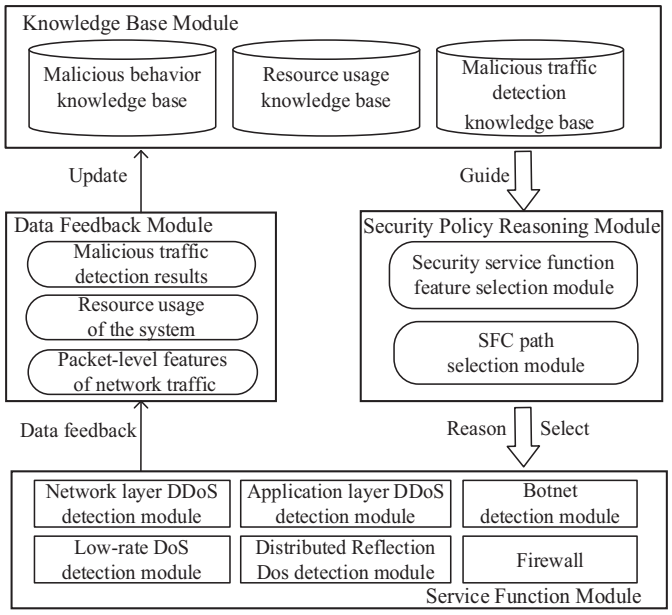


Figure 1: Intelligent security service framework

The main function of the data feedback module is to periodically collect feedback data from various NSFs, store it in the knowledge base, and update the knowledge base to guide the reasoning of security policies. The collected data includes the packet-level features of network traffic entering the SFC, malicious traffic detection outcomes of various NSFs, and system resource utilization.

The knowledge base module consists of the malicious behavior knowledge base, malicious traffic detection knowledge base, and resource usage knowledge base. The malicious behavior knowledge base mainly includes flow-level feature datasets and packet-level feature datasets of DDoS attacks, as well as a malicious behavior feature graph. The malicious traffic detection knowledge base stores detection results from the service function module, such as accuracy, malicious traffic detection rate, detection capability, and detection time. The resource usage knowledge base tracks system resource utilization of the service function module, including CPU usage rate, memory usage rate, disk usage rate, packet loss rate, etc.

The security policy reasoning module consists of an NSF feature selection module and an SFC path selection module. Using the knowledge stored in the knowledge base, we design algorithms to dynamically adjust the NSFs that the traffic needs to pass through.

The service function module virtualizes the malicious traffic detection methods into NSFs, including the Network layer DDoS (NetDDoS) detection module [21], Application layer DDoS (AppDDoS) detection module [22], Botnet detection module [10], Low-rate DoS (LDDoS) detection

module [23], Distributed Reflection Dos (DRDoS) detection module [24] and firewall. Detection methods are combined to form the detection path of SFC, and different detection paths have different effects on security performance.

3.2 Data Feedback Module

The data feedback module is responsible for collecting the packet-level features of network traffic entering the SFC, malicious traffic detection outcomes of various NSFs, and system resource utilization. This information is fed back to the knowledge base in real time, enabling dynamically adjustments to the traffic path through the SFC. The flow chart is shown in Fig. 2.

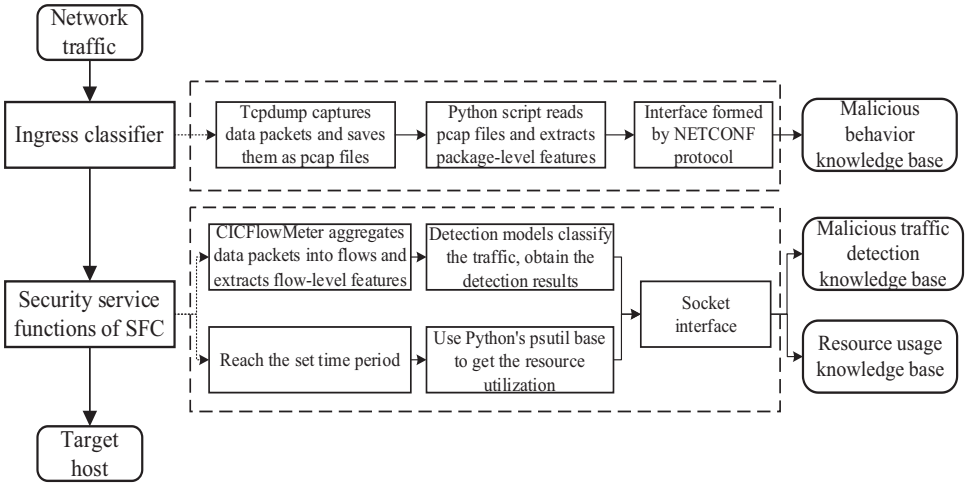


Figure 2: Data feedback diagram

We capture DDoS attack traffic entering the SFC with the Tcpdump data packet capture tool at the ingress classifier. This traffic is saved as pcap files, which are then parsed using a Python script to extract packet-level features. These features are saved as Comma Separated Values (CSV) files and fed back to the malicious behavior knowledge base via an interface established using the Network Configuration (NETCONF) protocol. In this step, the malicious behavior knowledge base acts as the client, while the ingress classifier, responsible for extracting the packet-level features from attack traffic, serves as the server. Initially, the malicious behavior knowledge base subscribes to the server. Upon extraction, the server automatically sends the subscribed client the packet-level features. Subsequently, the client utilizes Python's xml.dom library to analyze the packet-level feature information received through the NETCONF protocol. This information is saved to the malicious behavior knowledge base, which is updated in real time.

Each NSF [10,21–24] employs the CICFlowMeter [25] flow feature extraction tool to aggregate five identical data packets into a single flow. This process extracts detailed flow-level feature information, enabling model training and online deployment for fine-grained attack classification. Simultaneously, we utilize Python's psutil library to monitor CPU, memory, and other resource utilization within Docker containers hosting each NSF at specific intervals. The classification detection outcomes of each NSF and resource utilization of the system are transmitted back to the malicious traffic detection knowledge base and the resource usage knowledge base via a Socket interface. These knowledge bases serve as servers, employing multi-threaded Socket interfaces to receive feedback from the detection modules. Traffic detection information and resource utilization from each NSF,

acting as the clients, are transmitted via Socket to update both knowledge bases. Each module's traffic detection data includes accuracy, malicious traffic detection rate, detection capability, and detection time. Resource utilization metrics include CPU usage rate, memory usage rate, disk usage rate, and packet loss rate.

The knowledge base is continuously updated through a real-time feedback mechanism, where data such as attack features and system resource usage detected by the traffic monitoring modules are fed back into the system. This dynamic update strategy ensures that the knowledge base remains up-to-date, allowing the system to respond promptly to emerging threats and new attack patterns. We input the packet-level feature of DDoS attack traffic stored in the malicious behavior database into the SFC path selection module as input for the path selection algorithm. The detection outcomes from the malicious traffic detection knowledge base and the CPU usage rate of the resource usage knowledge base are used as parameters for the path selection algorithm. This approach aims to dynamically adjust the service path based on data from the real-time updated knowledge base, in order to determine the optimal detection path, which involves integrating various NSFs.

3.3 Knowledge Base Module

The knowledge base module comprises a malicious behavior knowledge base, a malicious traffic detection knowledge base, and a resource usage knowledge base. These correspond to the three types of information fed back to the knowledge base by the data feedback module. Fig. 3 depicts the structure diagram of each knowledge base and other modules.

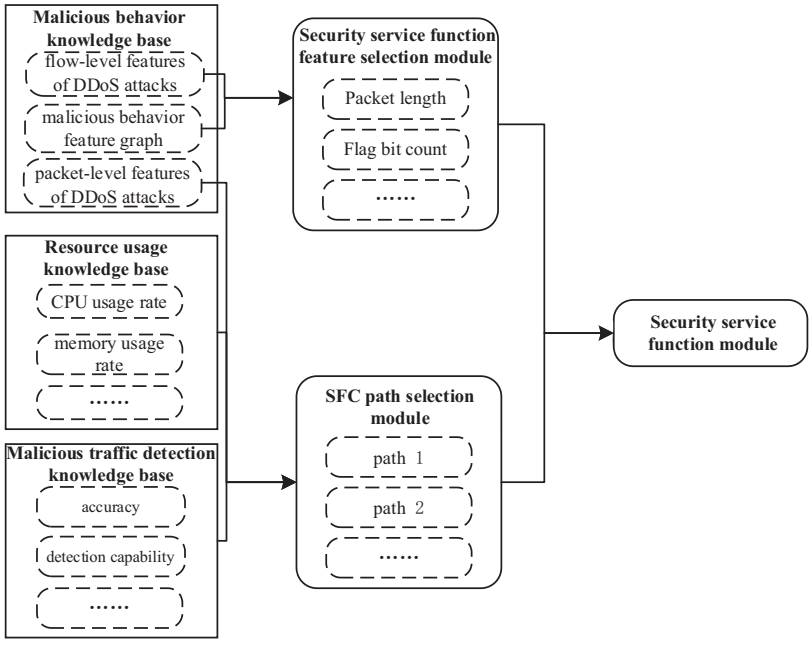


Figure 3: Relationship between the knowledge base and other modules

The primary components of the malicious behavior knowledge base include the flow-level feature datasets of DDoS attacks, the malicious behavior feature graph, and the packet-level feature datasets of DDoS attacks. The flow-level feature datasets of DDoS attacks encompass 83 feature attributes extracted by the CICFlowMeter, exemplified in Table 1.

Table 1: Flow-level features of DDoS attacks

Feature type	Feature name
Flow identification feature	Protocol
Marker bit features	FIN Flag Count, SYN Flag Count, RST Flag Count, ACK Flag Count, Fwd PSH Flag, Bwd URG Flag
Flow header features	Fwd Header Length, Bwd Header Length, FWD Init Win Bytes
Time features	Fwd IAT Min, Fwd IAT Max, Fwd IAT Mean, Fwd IAT Std, Fwd IAT Total, Flow IAT Min, Flow IAT Mean, Flow IAT Std, Flow duration, Idle Min, Idle Max, Idle Mean
Payload features	Total Fwd Packet, Total Bwd Packet, Total Length of Fwd Packet, Packet Length Min, Bwd Packet Length Max, AVG Bwd Segment Size, Flow Byte/s, Fwd Packets/s, Bwd Avg Bytes/Bulk

The malicious behavior feature graph [26] provides detailed classification and feature matching of DDoS attacks. As depicted in Fig. 4, purple nodes represent five types of DDoS attacks, light pink nodes denote 21 specific subtypes encompassed within these five types, and blue nodes signify 83 distinct feature attributes. For instance, within the category of DRDoS attacks, there are subtypes like Chargen, NTP, and TFTP. Each subtype exhibits unique features distinguishing it from normal traffic, such as Idle Mean. The flow-level feature datasets of DDoS attacks and the malicious behavior feature graph serve as inputs to the NSF feature selection module, providing data support for it.

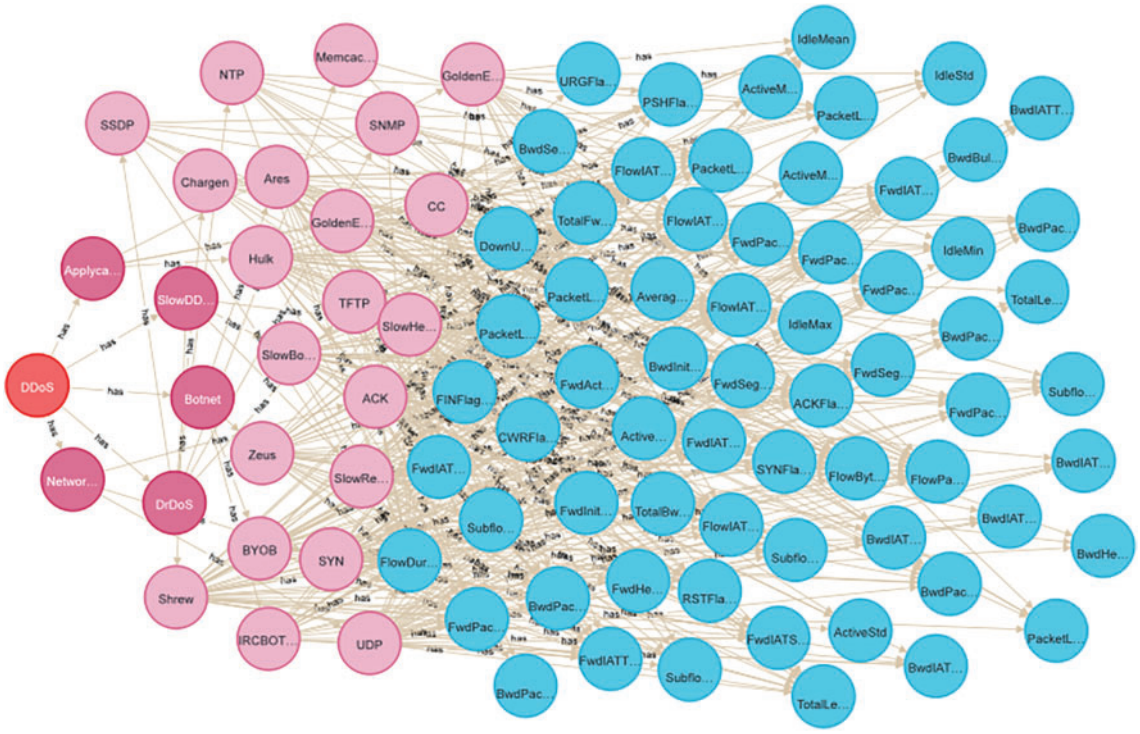


Figure 4: Malicious behavior feature graph

Table 2 illustrates the packet-level features utilized in identifying DDoS attacks, predominantly focusing on entropy-based features. The packet-level features include entropy of IP and port, packet rate, conditional entropy of packets, etc. DDoS attacks typically generate higher traffic rates compared to normal traffic, so we selected packet rate and byte rate as features. During a DDoS attack, a significant number of packets are directed to the same target IP address. By measuring the entropy of IP addresses within a specified time window, we can effectively identify the occurrence of a DDoS attack. Additionally, calculating the entropy of port numbers, along with the conditional entropy that combines both port and IP, allows us to effectively differentiate between various types of DDoS attack traffic. They serve as inputs to the SFC path selection module, providing data support for dynamically adjusting the service path to determine the optimal detection path.

Table 2: Packet-level features of DDoS attacks

Feature	Description
Packet rate	Average number of packets forwarded per second
Byte rate	Average number of bytes forwarded per second
Average packet length	Average length of data packets
Source IP entropy	Entropy of source IP address
Destination IP entropy	Entropy of destination IP address
TTL entropy	Entropy of packet lifetime TTL
TCP source port entropy	Entropy of TCP packet source port
TCP destination port entropy	Entropy of TCP packet destination port
UDP source port entropy	Entropy of UDP packet source port
UDP destination port entropy	Entropy of UDP packet destination port
Packet length entropy	Entropy of packet length
H(Sip|Dip)	Entropy of source IP when destination IP is given
H(Sip|Dport)	Entropy of source IP when destination port is given
H(Dport|Dip)	Entropy of source port when destination IP is given
Variance of packet number	Variance of the number of packets per unit interval

Table 3 outlines the content of the resource usage knowledge base, which includes critical indicators such as CPU usage rate, memory usage rate, disk usage rate, and packet loss rate. By analyzing CPU usage, memory usage, and disk usage, we can effectively identify and address causes of system performance degradation or failures. Monitoring packet loss rates also helps identify network connectivity issues, facilitating timely troubleshooting and resolution. These metrics are essential for constructing optimal SFC paths while ensuring that resource utilization remains within acceptable limits. They are used as parameters for the path selection algorithm.

Table 3: Resource usage knowledge base

Name	Description	Unit
cpu_usage	CPU usage rate	%
cpu_freq	CPU frequency	MHz

(Continued)

Table 3 (continued)

Name	Description	Unit
memory_total	Total system memory	GB
memory_available	Available memory	GB
memory_usage	Memory usage rate	%
disk_usage	Disk usage rate	%
disk_left	Disk remaining space	GB
packet_loss_rate	Packet loss rate	%

The core content of the malicious traffic detection knowledge base consists of detection outcomes from various NSFs, as shown in Table 4. These detection results correspond to traffic features, indicating accuracy, detection rate, detection capability, and detection time for each module at a given time, reflecting the effectiveness of the detection modules in identifying current traffic patterns. These results are used as parameters for the path selection algorithm, providing crucial data support.

Table 4: Malicious traffic detection knowledge base

Content	Description
Module name	Name of the detection module
Time stamp	Detected time stamp
Accuracy	The ratio of correctly predicted traffic to total traffic
Detection rate	The ratio of correctly predicted attacks to all predicted attacks
Detection capability	The ratio of total number of attacks to undetected attacks
Detection time	The time consumed by the detection module

The malicious behavior knowledge base, resource usage knowledge base, and malicious traffic detection knowledge base do not operate independently; they rely on traffic numbers as a common primary key for data linkage. Feedback information received within the same period is aggregated under the corresponding traffic number. This approach facilitates retrieval of traffic features, identifies which NSF detected the traffic, and provides access to results and resource usage data from detection modules. The entity relationship diagram is shown in Fig. 5.

To maintain the timeliness of the knowledge base, we implement an automatic expiration mechanism where each data entry is assigned a timestamp. Entries that exceed a predefined threshold are flagged as outdated and are either archived for further analysis or removed from active use. Additionally, the system periodically reviews the knowledge base content, ensuring that outdated information is purged or updated based on current network conditions and threat dynamics.

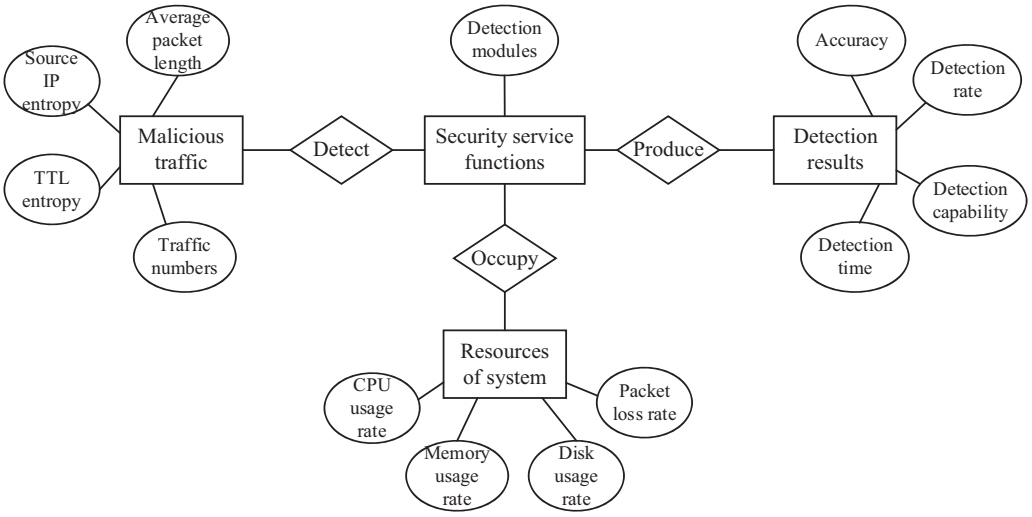


Figure 5: Entity relationship diagram

3.4 Security Policy Reasoning Module

3.4.1 NSF Feature Selection Module

Feature selection [27] plays a crucial role in machine learning, particularly for enhancing detection module performance by identifying a subset of original feature data. In this section, we propose a novel feature selection method that combines the PageRank feature ranking approach with the RandomForest algorithm. This method leverages the malicious behavior feature graph and the flow-level feature datasets of DDoS attacks from the malicious behavior knowledge base. The goal is to assess the importance of the features corresponding to each attack in the graph, pinpoint flow-level features that significantly influence attack classification, and integrate this information into various NSFs to enhance detection efficiency and capability. The overall flow diagram is depicted in Fig. 6.

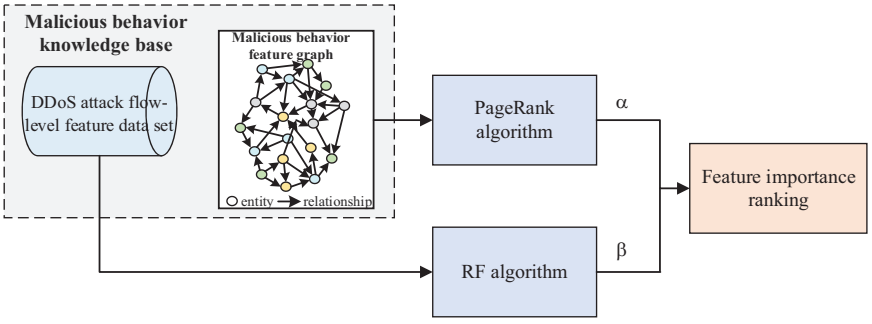


Figure 6: Feature ranking flow diagram

The malicious behavior feature graph of DDoS attacks shows that there is a correlation between each type of attack and the specific types of traffic features. However, relying only on feature engineering algorithms such as RandomForest cannot fully exploit the importance of data features, resulting in insufficient expression of malicious behavior features. PageRank is a well-known graph-based ranking algorithm [28] primarily used to rank web pages by analyzing the link structure within a

network. In this study, the PageRank algorithm is applied to effectively reason over and select features from the malicious behavior feature graph. Therefore, we combine PageRank and RandomForest for feature selection. PageRank is used for knowledge inference and feature selection in the malicious behavior feature graph, while RandomForest is used for selecting information from the flow-level feature datasets of DDoS attacks.

However, the malicious behavior feature graph within the knowledge base lacks relationships between features, which are essential for PageRank's random walk on a strongly connected directed graph. To address this, it becomes necessary to transform the graph from a tree structure into a strongly connected directed graph. The conversion formula is provided below:

$$A\colon X \to Y, X \to Z \tag{1}$$

$$B\colon Y \Leftrightarrow Z \tag{2}$$

where X represents the attack type, Y and Z represent the flow-level features. The graph undergoes a transformation from state A to state B. For instance, in the case of a DRDoS attack, Fig. 7 illustrates the relationships among its six attack subtypes, Chargen, NTP, and TFTP, and certain flow-level features such as Fwd Bulk Rate Avg and Packet Length Max. Each attack subtype's corresponding flow-level features are paired to construct a feature map, as depicted in Fig. 8. For example, for the TFTP attack subtype, the flow-level features include Fwd Bulk Rate Avg, Flow IAT Min, and Total Fwd Packet, all connected pairwise in Fig. 8.

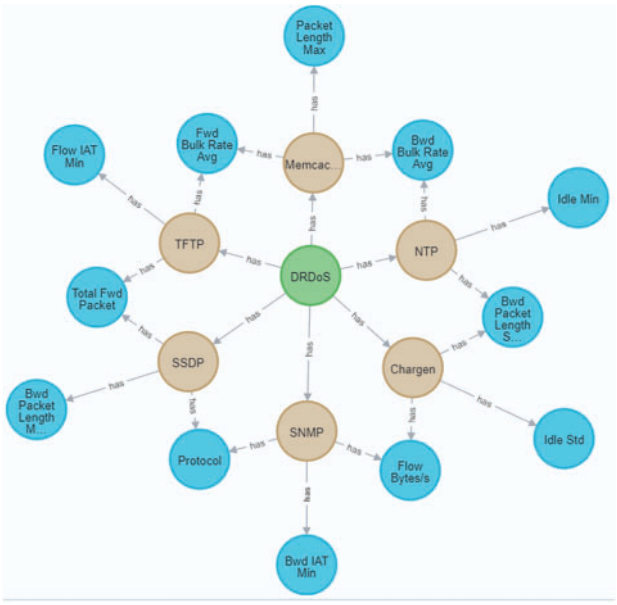


Figure 7: DRDoS attack features diagram

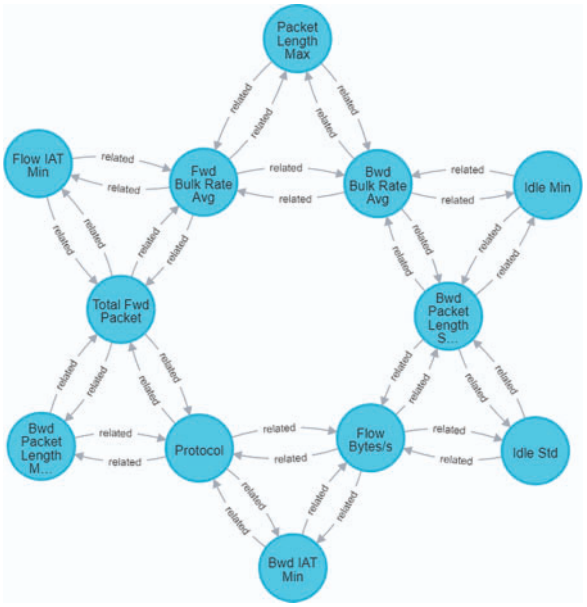


Figure 8: Feature map diagram

The RandomForest feature selection algorithm [29] assesses the importance of each type of DDoS attack and prioritizes features based on their significance. This method evaluates the contribution of each feature to every tree within the random forest, calculates their average contribution, and compares the importance across features.

The pseudo-code for combining PageRank with RandomForest to compute feature importance scores is outlined in Algorithm 1. The malicious behavior feature graph comprises 83 nodes representing DDoS attack features, thus setting the number of nodes to $n = 83$. To enhance the reliability of PageRank, parameters are configured as follows: the damping factor $d = 0.85$, the calculation accuracy $\sigma = 1e - 5$, and the maximum number of iterations $Q = 100$, as recommended in [14]. The first-order Markov chain transition matrix M is derived from the strongly connected feature map post-mapping, $x_{(1)}$ denotes the initial distribution matrix for feature importance scores, initialized assuming equal importance for each feature node, which is 1/83.

Algorithm 1: The algorithm of PageRank and RandomForest

Input: DDoS attack flow-level feature datasets, number of feature nodes n, damping factor d, first-order Markov chain transition matrix M, identity matrix E, calculation accuracy σ, maximum number of iterations Q, participation coefficient α, β, initial distribution matrix $x_{(1)}$

Output: The importance score of each feature

Establishing a RandomForest regression model *rf*
Using feature data X and target variables y to fit the RandomForest regression model
Extracting the importance score RF of each feature in the trained RandomForest model
for *epoches* $= 1$ to Q **do**
    Calculating the general transfer matrix: $A = dM + (1 - d)E$
    Iterate and normalize the result vector
    $y_{(epoch+1)} = Ax_{(epoch)}$
    $x_{(epoch+1)} = y_{(epoch+1)} / \left\| y_{(epoch+1)} \right\|$
    if $\left\| x_{(epoch+1)} - x_{(epoch)} \right\| < \sigma$ **then**
        $PR = x_{(epoch+1)}$
        break
        end if
end for
for $i = 1$ to n **do**
  $score[i] = \alpha RF[i] + \beta PR[i]$
end for
return score

PageRank is integrated with the RandomForest method, and the formula for computing the final score of each feature node is as follows:

$$score = \alpha RF + \beta PR \tag{3}$$

$$0 < \alpha, \beta, PR, RF < 1 \tag{4}$$

$$\alpha + \beta = 1 \tag{5}$$

where PR and RF denote feature importance scores from the PageRank and RandomForest algorithms, respectively. α and β represent the participation coefficients for RF and PR, ranging from 0 to 1. Subsequently, feature importance is ranked based on these scores, and the top flow-level features are selected. Using this refined feature set, the attack detection models in the service function module are retrained, resulting in optimized detection performance.

3.4.2 SFC Path Selection Module

The SFC path selection module trains a path selection model called GAT+DRL, which combines GAT and DRL to select the optimal detection path. As illustrated in Fig. 9, we first feed the packet-level feature datasets of DDoS attacks from the malicious behavior knowledge base into GAT. GAT learns both feature representations and structural relationships within the data. The aggregated features are then passed to DRL, which formulates the path selection task as Markov decision processes. DRL selects the optimal detection path by maximizing rewards derived from combining detection outcomes from the malicious traffic detection knowledge base and CPU usage rates from the resource usage knowledge base.

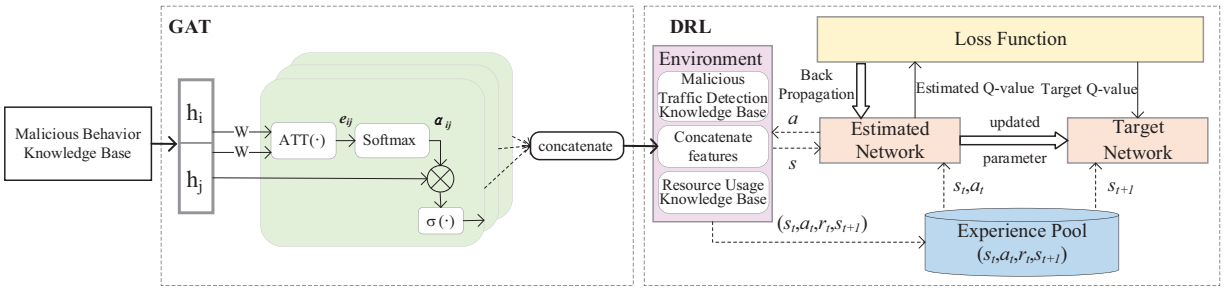


Figure 9: GAT+DRL model framework

GAT is a graph neural network model that incorporates an attention mechanism between data, enhancing its ability to capture both relational and feature information between data, thus enabling deep analysis and data mining capabilities. It can uncover high-level feature information about malicious traffic in the knowledge base, and based on these high-level features, DRL can more effectively select paths. On the other hand, DRL combines deep learning's perceptual capabilities with reinforcement learning's decision-making prowess. One prominent example is DQN, which approximates the Q-function using a deep neural network. The network takes the environment state as input and predicts Q-values for all possible actions in that state.

As depicted in Fig. 10, a substantial volume of packet-level feature data from diverse DDoS attacks in the malicious behavior knowledge base is initially inputted into GAT. Here, each attack node aggregates feature information transmitted by its neighboring nodes using an aggregation function. This accumulated information undergoes modification through a nonlinear update function. This iterative process repeats multiple times to generate the resultant features for each attacking node.

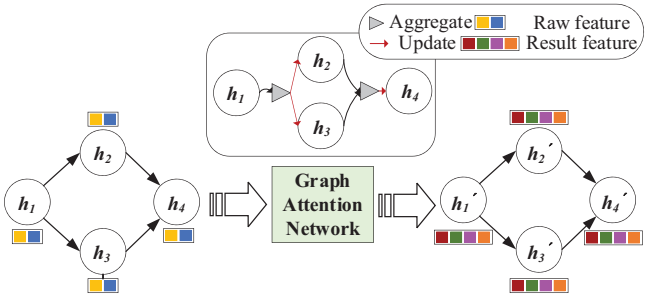


Figure 10: GAT architecture diagram

The feature information of input nodes is denoted as $h = \{h_1, h_2, \cdots, h_N\}, h_i \in R^F$, where N represents the total number of input malicious traffic and F represents the feature dimension of each traffic instance. The self-attention mechanism is then applied to compute the attention score between node i and its neighbor node j.

$$e_{i,j} = a\left(Wh_i, Wh_j\right) \tag{6}$$

where $W \in R^{F' \times F}$ denotes a shared linear transformation matrix that is trainable. The matrix is applied to each traffic node, enabling the original feature space to be transformed into a higher-level feature space, thereby enhancing the node's expression capability. The function $a(\bullet)$ computes the correlation between two nodes (vectors).

The importance of node j to node i is determined through $e_{i,j}$. To ensure coefficients are comparable across different nodes, we normalize using the softmax function, which yields standardized attention scores.

$$\alpha_{i,j} = \text{softmax}\left(e_{i,j}\right) = \frac{\exp\left(e_{i,j}\right)}{\sum_{K \in N_i} \exp\left(e_{i,k}\right)} = \frac{\exp\left(LeakReLu\left(a\left[Wh_i || Wh_j\right]\right)\right)}{\sum_{K \in N_i} \exp\left(LeakReLu\left(a\left[Wh_i || Wh_j\right]\right)\right)} \tag{7}$$

where $||$ denotes the concatenation of two transformed nodes, a is a trainable parameter vector, known as the attention parameter vector, with a size of $2 \times F'$ (twice the embedded size after conversion), which is utilized to learn the relative importance between nodes and their neighbors, the entire expression $a\left(Wh_i, Wh_j\right)$ represents the dot product between a and the transformed node.

After each neighbor's feature vector is multiplied by a dimension transformation vector parameter, weighted by attention scores, and subsequently passed through an activation function, the resulting aggregated features corresponding to each node are obtained.

$$h_i^{'} = \sigma\left(\sum_{j \in N_i} \alpha_{i,j} Wh_j\right) \tag{8}$$

Based on the aggregated features of GAT, the detection outcomes in the malicious traffic detection knowledge base and the resource utilization in the resource usage knowledge base, a DRL environment is constructed. Subsequently, the problem of SFC path selection is modeled as a Markov Decision Process (MDP), comprising state space, action space, and reward functions.

State: In the DRL framework, the state represents the information available to the agent from the environment. For the SFC path selection problem, the state space consists of the aggregated features output by the GAT and the accuracy of each detection module in the malicious traffic detection knowledge base. This state can be represented as a vector $S_t = \left\{h_i^{'}, \phi\right\}$, where $h_i^{'}$ denotes the high-level feature aggregated by GAT in the previous stage, and ϕ represents the detection accuracy of each detection module.

Action: In DRL, the selection of NSFs is guided by traffic features and detection outcomes. Therefore, the action space A is defined as the set of possible NSFs $A = \{f_1, f_2, \cdots, f_6\}$, including five detection modules and firewalls, totaling six schemes. When an agent takes an action, it adds the corresponding NSF to the path list. If the action is to add a firewall, the path list construction concludes, making the end of a training round.

Reward: The DRL agent improves its performance by receiving reward signals from the external environment. Typically, when an NSF is selected based on the current state, a positive reward is issued

to reinforce the likelihood of action being chosen. This positive reward correlated with the malicious traffic detection capability in the malicious traffic detection knowledge base and the CPU usage rate in the resource usage knowledge base. Conversely, if the selected action duplicates a detection module already included in the SFC path, a negative reward is given. This negative reinforcement prompts the agent to explore alternative decisions. The reward function for action at is defined as follows:

$$r_t = \begin{cases} -\omega, NSF \text{ duplicate} \\ \dfrac{MTDC \cdot (1 - cpu\_usage)}{n}, \text{otherwise} \end{cases} \tag{9}$$

where ω is a fixed positive number, $MTDC$ is the malicious traffic detection capability of the selected NSF, cpu_usage is the CPU usage rate, and n is the number of NSFs that the packet passes through.

In the training process of the DRL model, after each decision, the agent selects actions using an ε-greed strategy. ε is a value between 0 and 1. In this strategy, the agent selects the known optimal action with a probability of ε, and with a probability of 1-ε, it chooses a random action to explore unknown situations. Given the diversity and complexity of security service functions, the number of alternative paths can be very large. It is not practical to design alternative paths manually at this point, so the agent is needed to explore the environment and enhance adaptability in SFC path selection in order to select the optimal path. Upon executing an action, the corresponding state, action, reward, and subsequent state information are stored in the experience pool. Periodically, batches of data are sampled from this pool for training, during which the target network parameters are updated. The loss function used to assess the estimation performance during training is defined as follows:

$$J(\theta_t) = E\left[\left(r_t + \gamma Q\left(s_t, \arg\max_a Q(s_{t+1}, a|\theta_t)\,|\theta_t^{'}\right) - Q(s, a_t|\theta_t)\right)^2\right] \tag{10}$$

where θ_t and $\theta_t^{'}$ represent the parameters of the evaluated network and target network, respectively, and γ is the discount factor. The loss function is defined as the mean square error between the target Q-value $r_t + \gamma Q\left(s_t, \arg\max_a Q(s_{t+1}, a|\theta_t)\,|\theta_t'\right)$ from the target network, and the estimated Q-value $Q(s, a_t|\theta_t)$ from the evaluated network. Minimizing this loss function improves the evaluation performance of the model. The gradient descent algorithm is employed to update the parameters of the evaluated network, optimizing the loss function as follows:

$$\theta_t^{+} = \theta_t + \beta\left[r_t + rQ\left(s_t, \arg\max_a Q(s_{t+1}, a|\theta_t)\,|\theta_t'\right) - Q(s, a_t\theta_t)\right]\nabla Q(s, a_t\theta_t) \tag{11}$$

where $\theta_t{}^{+}$ represents the updated parameters of the estimated network after applying the gradient descent algorithm, and β represents the step size parameter used in the gradient descent update process.

This pseudo-code outlines the training process for a system that combines GAT with DRL for detecting DDoS attacks in Algorithm 2. It initializes the environment and network parameters, performs GAT-based feature extraction, allows the DRL agent to interact with the environment using ε-greedy exploration, stores experiences in a replay memory, updates the network parameters based on sampled experiences, and periodically updates the target network. Finally, it saves the trained model for use in the online detection of malicious traffic.

Algorithm 2: The algorithm of GAT+DRL

Input: The packet-level features of DDoS attacks in malicious behavior knowledge base $h = \{h_1, h_2, \cdots, h_N\}, h_i \in R^F$, learning rate LR, ε-greedy strategy, target network update frequency $Target\_update$, capacity of the memory base $capacity$, accuracy ϕ in malicious traffic detection knowledge base, malicious traffic detection capability $MTDC$

Output: The trained GAT+DRL model

1. Initialize the parameters of estimated network and target network, $\theta_t' = \theta_t$
2. **for** each training episode **do**
3. Calculate the attention scores $e_{i,j}$ according to Eq. (6)
4. Normalize to get the attention matrix $\alpha_{i,j}$ according to Eq. (7)
5. Output the aggregated features $h_i^{'}$ according to Eq. (8)
6. Get the current status s_t from the environment
7. Use ε-greed to choose action $a_t = \begin{cases} \arg\max_{a \in A} Q(s_t, a), rand() < \varepsilon \\ random, otherwise \end{cases}, \varepsilon \in [0, 1)$
8. Execute the action a_t to get the reward r_t and the next status s_{t+1}
9. Store samples (s_t, a_t, r_t, s_{t+1}) in the experience pool
10. **if** $memory\_counter > capacity$ **then**
11. Randomly extract sample (s_t, a_t, r_t, s_{t+1}) from the memory base for learning
12. Calculate the loss function according to Eq. (10)
13. Update the parameters of the evaluated network according to Eq. (11)
14. **if** $learn\_step\_counter \% Target\_update = 0$ **then**
15. Update the parameters of target network
16. **end if**
17. **end if**
18. **if** $a_t =$ 'firewall' **then**
19. break
20. **end if**
21. **end for**
22. torch.save('model.pkl')

4 Experimental Results

This section focuses on experimental testing and result analysis of the proposed intelligent security service optimization method. Section 4.1 outlines the construction of the experimental environment. Section 4.2 presents some new evaluation metrics. The update process of the knowledge base data is evaluated in Section 4.3. Section 4.4 scrutinizes the performance of both the NSF feature selection algorithm and the SFC path selection algorithm. Section 4.5 assesses the system's online detection performance under different paths. Finally, to verify the impact of updating the knowledge base with real-time feedback data on path selection performance, Section 4.6 evaluates the effectiveness of the knowledge base before and after updating based on detection delay and TMTDC.

4.1 Experimental Environment

This section establishes an experimental environment using VMware vSphere, with the experimental topology illustrated in Fig. 11. Each virtual machine runs Ubuntu 18.04 with 16 GB of memory and 30 GB of disk space allocated. The experimental setup comprises a knowledge base

host, two classifiers, four repeaters, a host for generating attack and normal traffic, and a target host. For software implementation, Open vSwitch and OpenDaylight are utilized to construct the SFC, facilitating the virtualized deployment of NSFs through Docker.

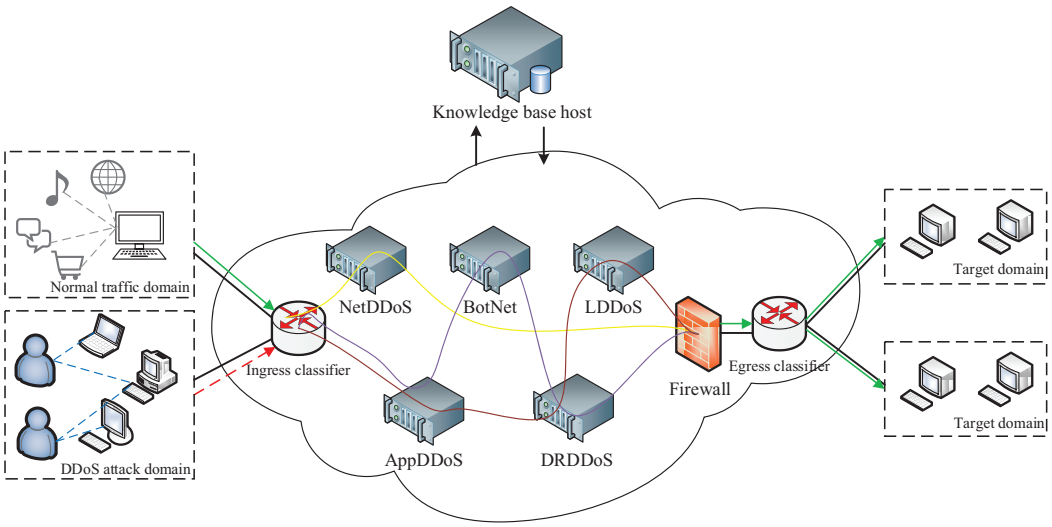


Figure 11: Experimental topology

In our experimental environment, several NSFs are employed, including the NetDDoS detection module, AppDDoS detection module, Botnet detection module, LDDoS detection module, DRDoS detection module, and firewall. To evaluate the system, we utilize Python scripts and attack tools to simulate 21 types of attacks on designated hosts. Additionally, normal traffic data is collected from activities such as voice calls, video streaming, and online gaming within a 5G environment [21].

4.2 Evaluation Metrics

In the experiment, five evaluation metrics are employed to assess the detection effectiveness: confusion matrix, accuracy, precision rate, recall rate, and F1 score. The confusion matrix illustrates the relationship between the model's predictions and the actual labels.

In addition, we define new evaluation metrics to assess the effectiveness of online detection: malicious traffic detection rate, path detection delay, and total malicious traffic detection capability (TMTDC). The malicious traffic detection rate quantifies the proportion of attack traffic correctly identified after detection. Here, i is the attack traffic category, n is the total number of attack traffic categories, T_i is the number of correctly detected instances within category i, and A_i is the total number of instances in category i. The malicious traffic detection rate is defined as:

$$Detection\ Rate = \frac{\sum_{i=1}^{n} T_i}{\sum_{i=1}^{n} A_i} \tag{12}$$

The path detection delay: For an SFC path, the path detection delay reflects its response speed and serves as a crucial performance evaluation metric. Eq. (13) outlines the calculation formula for

the path detection delay:

$$Delay_{SFC} = \frac{\sum_{i=1}^{C} dealy\left(NSF_c, NSF_{c+1}\right)}{C} \tag{13}$$

where C represents the length of the SFC, and $\sum_{i=1}^{C} dealy\left(NSF_c, NSF_{c+1}\right)$ denotes the path delay from the c-th NSF to the $(c+1)$-th NSF, which is the sum of delays along the SFC path.

TMTDC: This metric measures the malicious traffic detection capability along the selected path and serves as a crucial evaluation index of path quality. It is defined as:

$$TMTDC = \frac{\sum_{c=0}^{C} MTDC\left(NSF_c\right)}{C} + \frac{Average\left[MTDC\left(NSF_c\right)\right] \times \left[\frac{\sum_{c=0}^{C} MTDC\left(NSF_c\right)}{Average\left[MTDC\left(NSF_c\right)\right]} - C\right]}{C} \tag{14}$$

where C is the number of NSFs selected in the path list, $MTDC\left(NSF_c\right)$ represents the detection capability of the c-th NSF, $\sum_{c=0}^{C} MTDC\left(NSF_c\right)$ denotes the sum of detection capabilities of the selected NSFs, and $Average\left[MTDC\left(NSF_c\right)\right]$ represents the average detection capability of the selected NSFs. Detection capability refers to the ratio of the total number of malicious traffic to the undetected malicious traffic, which is defined as:

$$MTDC = \frac{1}{1 - \frac{\sum_{i=1}^{n} T_i}{\sum_{i=1}^{n} A_i}} \tag{15}$$

4.3 Knowledge Base Update

The traffic detection outcomes from each detection module within the service function module include accuracy, malicious traffic detection rate, detection capability, and detection time. Resource utilization metrics encompass CPU usage rate, memory usage rate, disk usage rate, and packet loss rate. These metrics are transmitted via Socket interface feedback to the malicious traffic detection knowledge base and the resource usage knowledge base respectively. To manage this data, multi-threaded processes are created by these knowledge bases to receive feedback from each detection module. Fig. 12 illustrates the knowledge base listening to ports, showing data reception from port 7001, corresponding to the DRDoS detection module.

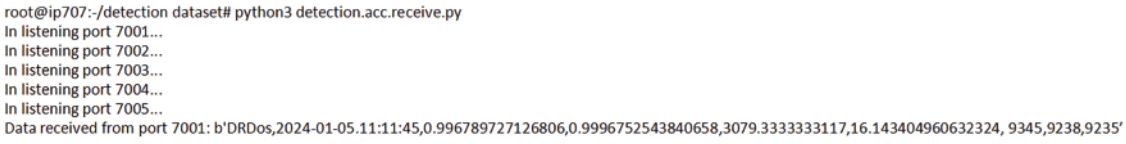


Figure 12: Monitoring of knowledge base

Fig. 13 illustrates that the traffic detection outcomes are stored in the malicious traffic detection knowledge base located at */root/detection_dataset/drdos-acc.csv*, while the resource utilization of the host hosting the detection module is stored in the resource usage knowledge base at */root/detection_dataset/drdos-usage.csv*. The timestamp in the malicious traffic detection outcomes

has been updated from 2023-12-29 to 2024-01-05, indicating the recent update of both the malicious traffic detection outcomes and the system resource utilization in the knowledge base.

```
root@ip707:~/detection_dataset# ls
appddos.csv                 clean.py                    detection-usage-receive.py  drdos-usage.csv           preset_policy_results.py
benchmark_policy_results.py detection-acc-receive.py    drdos                       dynamic_policy_results.py
botnet.csv                  detection_dataset_receive2  drdos-acc.csv               lddos.csv
clean2.py                   detection_dataset_receive.py drdos.csv                  network.csv
root@ip707:~/detection_dataset# cat drdos-acc.csv
Module,Timestamp,Accuracy,Detection Rate,Detection Capability,Detection Time,Total traffic,Total malicious traffic,Detected malicious traffic,
DRDoS,2023-12-29-03_44_37,0.9995051138238206,0.9994991652754591,1996.6666666667631,10.481446743011475,6062,5990,5987
""
DRDoS,2024-01-05-11:11:45,0.9996789727126806,0.9996752543840658,3079.333333333117,16.143404960632324,9345,9238,9235
""
```

Figure 13: Update of malicious traffic detection and resource usage knowledge base

The traffic feature information, which includes packet-level features of attack traffic such as packet rate, source IP entropy, destination IP entropy, source port entropy, destination port entropy, TTL entropy, etc., is fed back to the malicious behavior knowledge base via the NETCONF protocol. This feedback process is illustrated in Fig. 14.

```
----------------------------------------
Extracting features.....
packets per second:779.79
bytes per second: 8529.90
packet size mean: 167.95
src_ip_entropy: 4.92
dst_ip_entropy: 2.33
delta_src_ip_entropy: 5.39
delta_dst_ip_entropy: 3.81
TTL_entropy: 1.46
tcp_src_port_entropy: 1.0
tcp_dst_port_entropy: 1.0
udp_src_port_entropy: 4.93
udp_dst_port_entropy: 2.49
packet_size entropy: 4.05
sip l dip_entropy: 2.87
sip l dport_entropy: 2.16
dport l dip_entropy: 0.22
deviation: 0.6608
----------------------------------------
```

Figure 14: Knowledge feedback

Fig. 15 demonstrates that the packet-level feature information of traffic is stored in the malicious behavior knowledge base located at */monitor/monitor_information/feature*. The CSV file containing this information has been updated from 2023-06-12 to 2024-01-05, indicating the recent update of packet-level traffic features in the knowledge base.

```
root@ip707:~/monitor/monitor_information# cd feature
root@ip707:~/monitor/monitor_information/feature# ls
2022-07-19-12-37.csv  2023-06-06-02-20.csv  2023-06-07-08-58.csv  2023-06-12-05-51.csv
2022-07-19-12-38.csv  2023-06-06-02-21.csv  2023-06-07-08-59.csv  2023-06-12-05-52.csv
2022-07-19-12-39.csv  2023-06-06-02-22.csv  2023-06-07-09-00.csv  2023-06-12-05-53.csv

2023-06-06-02-14.csv  2023-06-07-08-52.csv  2023-06-12-05-45.csv  2024-01-05-11-07.csv
2023-06-06-02-15.csv  2023-06-07-08-53.csv  2023-06-12-05-46.csv  2024-01-05-11-08.csv
2023-06-06-02-16.csv  2023-06-07-08-54.csv  2023-06-12-05-47.csv  2024-01-05-11-09.csv
2023-06-06-02-17.csv  2023-06-07-08-55.csv  2023-06-12-05-48.csv  2024-01-05-11-10.csv
2023-06-06-02-18.csv  2023-06-07-08-56.csv  2023-06-12-05-49.csv  2024-01-05-11-11.csv
2023-06-06-02-19.csv  2023-06-07-08-57.csv  2023-06-12-05-50.csv  2024-01-05-11-12.csv

781.36,977748.09,1251.34,0.0,0.0,0.0,0.0,0.0,0,0,1.6,5.06,1.57,0.0,0.0,5.06,0.0359
781.36,977748.09,1251.34,0.0,0.0,0.0,0.0,0.0,0,0,1.6,5.06,1.57,0.0,0.0,5.06,0.0359
```

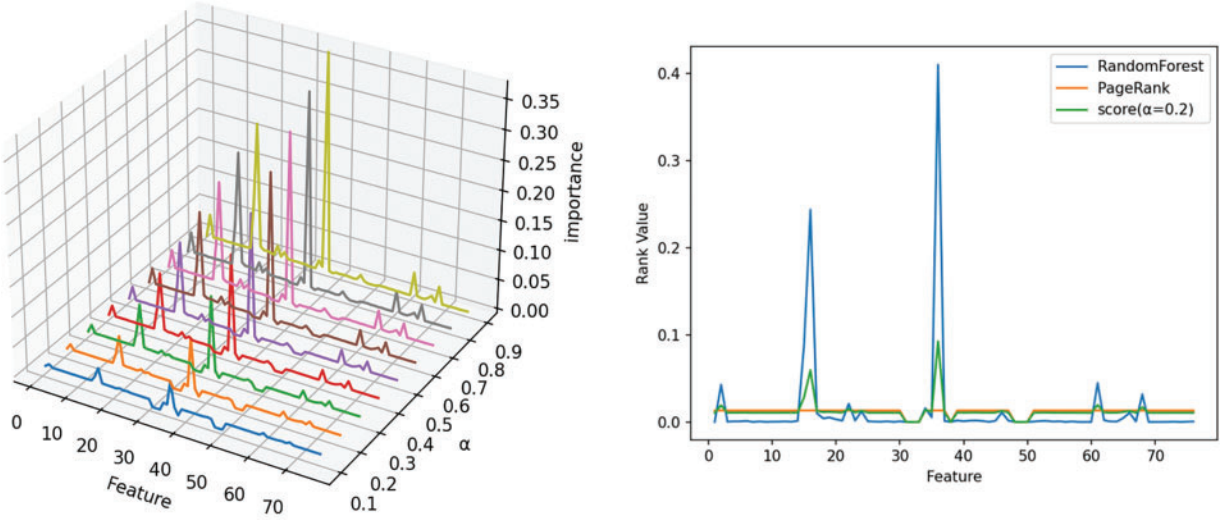
Figure 15: Update of malicious behavior knowledge base

4.4 Offline Training Results

This section includes the training results of the feature selection model and the SFC path selection model. Based on Python tools, this section first introduces the implementation of parameter tuning and optimization for the feature selection model, along with the result analysis using known datasets. It then presents the training results of the SFC path selection model and the corresponding result analysis using known datasets.

4.4.1 Training Results of Feature Selection Model

According to Eq. (3), the importance score varies with parameters α and β. In Fig. 16a, the score is depicted under different values of α and β. The x-axis represents 83 feature types, the y-axis represents the value of α, and the z-axis represents the importance score. We increment α from 0.1 to 0.9 in steps of 0.1, while β varies oppositely. The importance score based on RandomForest shows significant variability, whereas the distribution of PageRank is relatively uniform. Therefore, $\alpha = 0.2$ is chosen to ensure that the score is not overly influenced by RandomForest, as illustrated in Fig. 16b.



(a) The feature importance transformation of PageRank

(b) Comparison of three scores when α = 0.2 and RandomForest with α, β values from 0.1 to 0.9

Figure 16: The importance score under different values of α and β

To validate the detection performance of the feature selection algorithm combining PageRank and RandomForest proposed in our paper, we generated three sets of feature importance rankings. These rankings include PageRank and RandomForest combined, PageRank only, and RandomForest only used in Reference [12]. For each ranking, the top 20 features were selected and are presented in Table 5.

Based on the 20 effective features outlined in Table 5, feature selection was conducted, and the datasets were partitioned into training and test sets with a ratio of 7:3, as detailed in Table 6. The total number of data samples in the DRDoS feature datasets is 1,299,286, comprising 909,500 samples for training and 389,786 for testing. The performance of the feature selection methods was compared under the condition of using the same training datasets.

Table 5: Selected features of three methods

Feature selection method	Feature name
PageRank combined with RandomForest	Fwd Packets/s, Flow IAT Mean, Flow Packets/s, Subflow Fwd Packets, Total Fwd Packet, Fwd Seg Size Min, Fwd IAT Std, Fwd Header Length, Fwd IAT Min, PSH Flag Count, Bwd Init Win Bytes, Flow IAT Std, Bwd Header Length, Flow IAT Min, FWD Init Win Bytes, Flow IAT Max, Fwd IAT Total, Subflow Fwd Bytes, Packet Length Max, Packet Length Variance
PageRank	Fwd Bulk Rate Avg, Fwd Packet/Bulk Avg, Average Packet Size, Bwd Segment Size Avg, Packet Length Min, Subflow Fwd Bytes, ACK Flag Count, Idle Min, PSH Flag Count, RST Flag Count, SYN Flag Count, FIN Flag Count, Packet Length Std, Packet Length Mean, Bwd Bytes Avg, Bwd Packet Avg, Bwd Bulk Rate Avg, Subflow Fwd Packets, Subflow Fwd Bytes, Subflow Bwd Packets
RandomForest	Fwd Packets/s, Flow IAT Mean, Flow Packets/s, Subflow Fwd Packets, Total Fwd Packet, Average Packet Size, Fwd IAT Std, Fwd Header Length, Subflow Fwd Bytes, PSH Flag Count, Bwd Init Win Bytes, Flow IAT Std, Bwd Header Length, Flow IAT Min, Packet Length Min, Flow IAT Max, Fwd IAT Total, Subflow Fwd Bytes, Packet Length Max, Packet Length Variance

Table 6: DRDoS feature datasets

Datasets type	Number of normal traffic samples	Number of attack traffic samples
Training set	426,545	482,955
Test set	182,805	206,981

We applied the test dataset to XGBoost within the DRDoS detection module [30] and examined its output results. Fig. 17 displays the confusion matrix for features selected by three different methods of feature selection. Comparatively, the feature selection method combining PageRank and RandomForest achieves a high accuracy of 0.9999 overall, particularly excelling with perfect accuracies (1.0) in detecting Chargen, Memcached, and TFTP attacks. In contrast, the RandomForest feature selection method achieves an accuracy of 0.9970 overall, with perfect accuracies (1.0) for NTP and TFTP attacks but slightly lower accuracy (approximately 0.9892) for Chargen attacks. Meanwhile, the PageRank feature selection method achieves an accuracy of 0.9603 overall, with perfect recognition (1.0 accuracy) for TFTP attacks but lower accuracy for Chargen attacks.

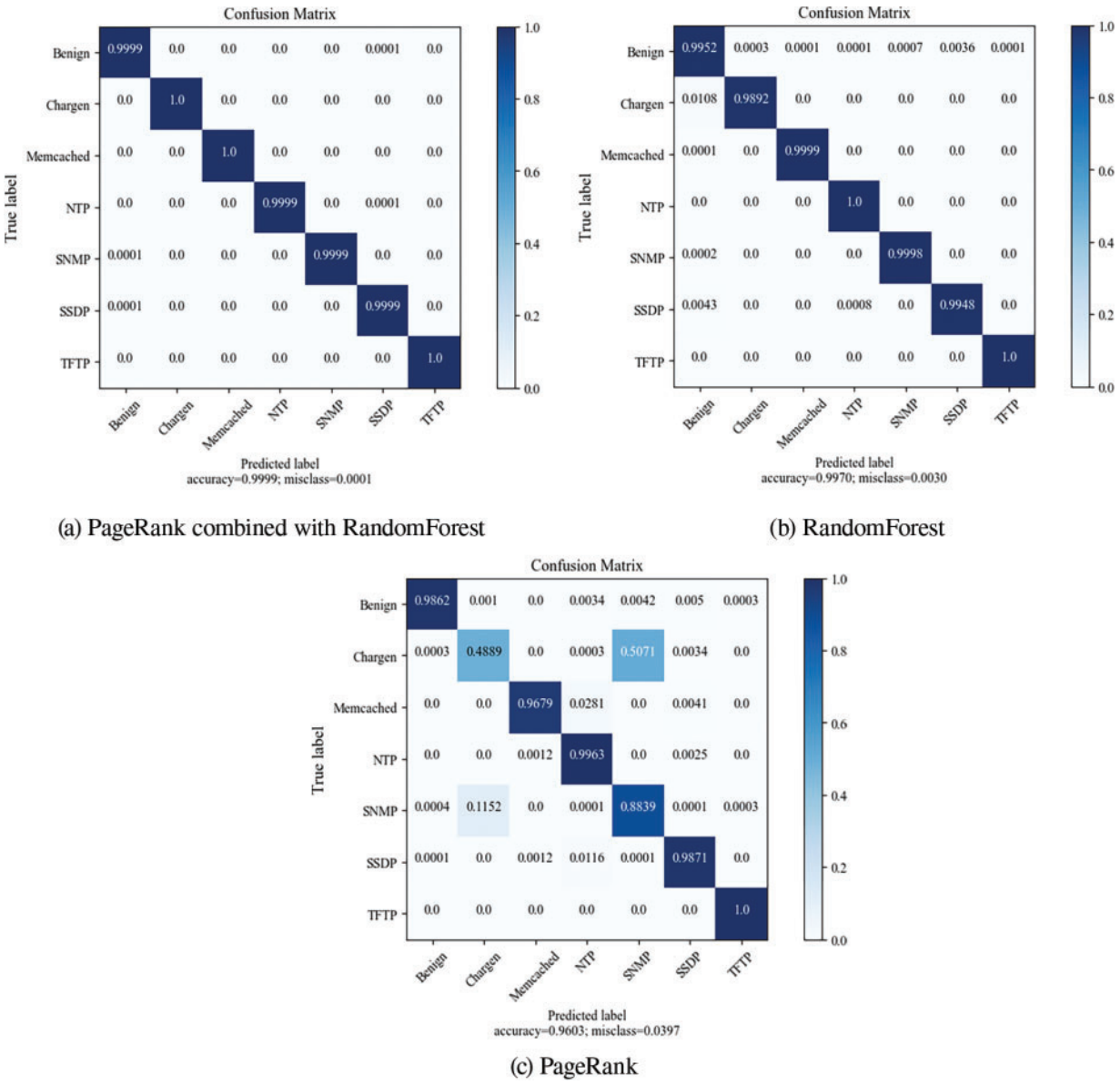


Figure 17: Confusion matrix of XGBoost model under three feature selection methods

Based on the results from three feature selection methods, we further apply two additional machine learning models, LightGBM [31], KNN [32], along with the Stacking integrated learning model discussed in reference [19], and the previously mentioned XGBoost detection model to compare their training times and accuracy.

Fig. 18 depicts the training times associated with three feature selection results across each model. Specifically, the KNN model, XGBoost model, LightGBM model, and Stacking model utilizing PageRank and RandomForest feature selection exhibit training times of 75.81, 160.09, 73.10, and 364.67 s, respectively. These times consistently show that models utilizing PageRank and

RandomForest feature selection have the shortest training times compared to other feature selection methods.

Fig. 19 compares the accuracy of three feature selection results across each model. The results indicate that features selected using the combination of PageRank and RandomForest consistently demonstrate higher accuracy in each training model compared to the other two feature selection methods. This approach not only achieves shorter detection times but also enhances performance in identifying various types of DRDoS attacks and normal traffic. Experimental findings underscore the capability of our feature selection method to extract more influential features related to DDoS attacks, thereby significantly improving the detection efficiency and performance of NSFs.

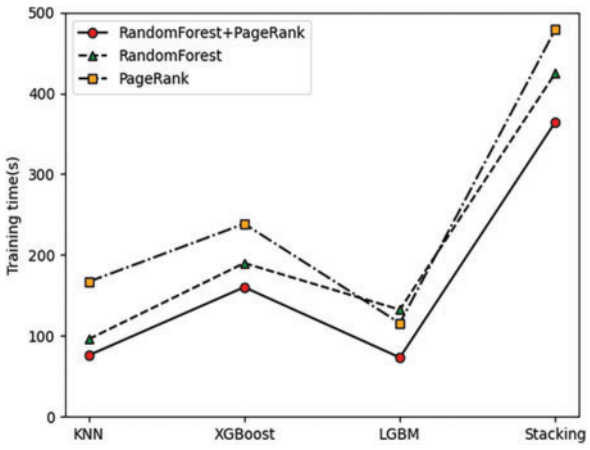


Figure 18: Training time of four models

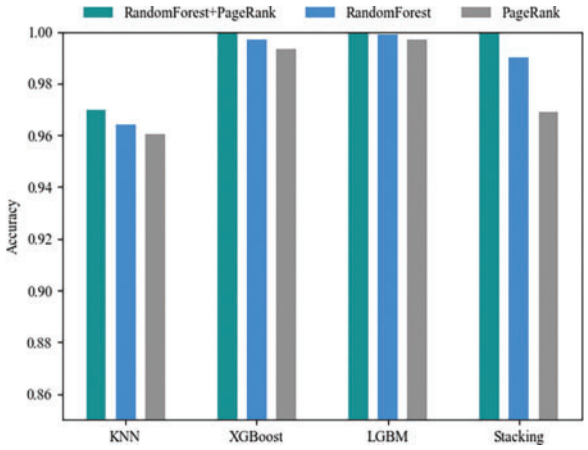


Figure 19: Accuracy of four models

4.4.2 Training Results of SFC Path Selection Model

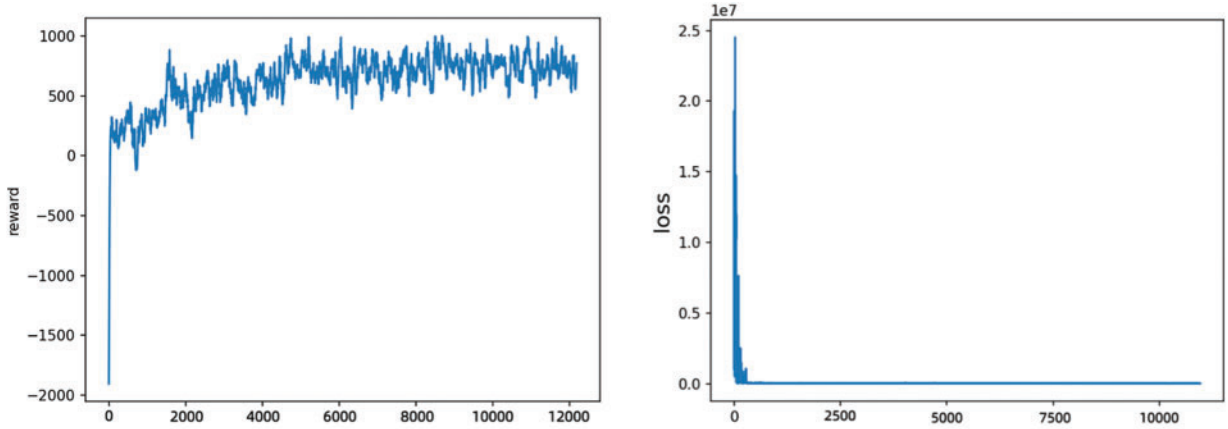
In the SFC path selection algorithm proposed in our paper, the GAT+DRL model is trained using the Pytorch framework. We present the relevant parameters and training process used for training the SFC path selection model, as well as the results of path selection testing using the trained model on known attack datasets.

The experimental implementation is coded in Python, and the relevant parameters settings during the training process [18] are detailed in Table 7.

Table 7: Parameter settings

Parameter	Value
Learning rate	0.001
Total training period	12,000
Memory bank	10,000
Exploration factor	0.9
Discount factor	0.9
Parameter update frequency	200
Optimizer	Adam

Fig. 20 illustrates the training process of the loss and reward values for the GAT+DRL model using selected parameters, over a total of 12,000 iterations. As depicted in Fig. 20a, the loss value steadily decreases as the number of training iterations increases. Fig. 20b demonstrates that the reward value of the path exhibits a gradual increase with training iterations, reaching convergence to its maximum after approximately 5000 iterations.



(a) Loss value (b) Reward value

Figure 20: The training process of the GAT+DRL model

The model achieves convergence within a training duration of 19.4 min and can distinguish between five different types of DDoS attacks and mixed attacks. Fig. 21 demonstrates the selection of the optimal path by inputting traffic of various attack types.

```
------------------test------------------------
env.step_counter: 2
env.path_choice: ['drdos', 'firewall']
reward_sum: 978.0
```

(a) Path of DRDoS attack

```
------------------test------------------------
env.step_counter: 2
env.path_choice: ['botnet', 'firewall']
reward_sum: 1013.0
```

(b) Path of Botnet attack

```
------------------test------------------------
env.step_counter: 3
env.path_choice: ['application', 'lddos', 'firewall']
reward_sum: 920.17
```

(c) Path of mixed attacks of AppDDoS and LDDoS

Figure 21: Path selected by GAT+DRL model

4.5 Online Test Results

The online test implements real-time detection of various types of DDoS attack traffic using the trained models and provides feedback data to the knowledge base module. This section aims to assess the effectiveness of enhancing system detection performance and reducing detection time through intelligent security service optimization. The evaluation compares system performance before and after optimization across metrics such as precision rate, recall rate, F1 score, malicious traffic detection rate, path detection delay, and TMTDC.

In the system, a preset path incorporating all NSFs is established, through which traffic sequentially passes all detection modules. We conducted a replay of five types of DDoS attack traffic and applied both the DQN algorithm proposed in [18] and the intelligent security service optimization method proposed in this paper to select SFC paths. Subsequently, we compared the detection effectiveness across different paths. Table 8 presents the performance comparison of SFC detection for the five types of DDoS attack traffic under preset path, the SFC1 path selected by the DQN algorithm, and SFC2 path optimized by our method. For instance, for DRDoS attacks, preset path is ['network', 'application', 'botnet', 'lddos', 'drdos', 'firewall'], while the SFC1 path chosen by the DQN algorithm is ['drdos', 'lddos', 'botnet', 'firewall']. The SFC2 path selected through our optimization method is ['drdos', 'firewall'].

Table 8: Comparison of detection performance under different paths

Attack type	Selected path	Precision rate	Recall rate	F1 score	Detection rate
AppDDoS	Preset path	0.7653	0.7592	0.7892	0.7782
	SFC1 path	0.8668	0.8798	0.8849	0.8841
	SFC2 path	0.9785	0.9749	0.9667	0.9697
DRDoS	Preset path	0.8898	0.9778	0.9317	0.9331
	SFC1 path	0.9292	0.9566	0.9574	0.9582
	SFC2 path	0.9977	0.9943	0.9959	0.9991
NetDDoS	Preset path	0.9352	0.8924	0.9028	0.9028
	SFC1 path	0.9813	0.9146	0.9492	0.9292
	SFC2 path	0.9805	0.9922	0.9943	0.9925
LDDoS	Preset path	0.8778	0.8653	0.8898	0.8524
	SFC1 path	0.9317	0.9428	0.9541	0.9510
	SFC2 path	0.9934	0.9874	0.9757	0.9950
Botnet	Preset path	0.9232	0.9274	0.9268	0.9245
	SFC1 path	0.9853	0.9425	0.9617	0.9646
	SFC2 path	0.9613	0.9641	0.9711	0.9679

The precision rate, recall rate, and F1 score of SFC2 path, selected after optimizing the intelligent security service, consistently exceed 96% across all types of DDoS attack traffic. Compared to the preset path, SFC2 path also achieves an average increase of 12.4% in malicious traffic detection rate and 4.6% compared to the path selected by the DQN method, demonstrating overall excellent performance.

Fig. 22 shows the path detection delay comparison of three different types of attack traffic, namely, LDDoS, DRDDoS, NetDDoS and mixed attacks of AppDDoS and LDDoS, before and after the intelligent security service optimization. When attack traffic traverses the preset path, it must pass through all detection modules. Conversely, when using paths selected by the DQN method or through intelligent security service optimization, traffic only passes through selected detection modules, resulting in significantly reduced delays compared to the preset path. In our method of intelligent security service optimization, the features of attack traffic chosen by detection modules and the SFC path through which traffic flows are redesigned. This reduces both detection time and the number of NSFs each detection module encounters compared to the DQN method-selected path, thereby minimizing delays from detection and forwarding. Consequently, across all types of attack traffic, the path delay achieved through intelligent security service optimization in this study has a latency that is approximately 42.8% lower than that of the DQN method, and about 58.3% lower than the preset path.

Fig. 23 illustrates the TMTDC for three different paths: the preset path (blue line), the path chosen by the DQN method (yellow line), and the path optimized by intelligent security service (green line) under mixed attack traffic conditions. The TMTDC for the preset path is approximately 82, while the path selected by DQN shows a TMTDC of about 87. In contrast, the path optimized by intelligent security service exhibits a significantly higher TMTDC of 97, surpassing both the preset and DQN-selected paths by 18.1% and 11.5%, respectively. This indicates superior detection capability.

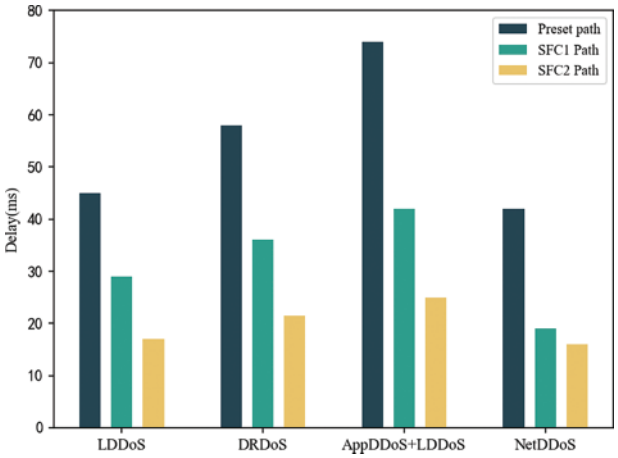


Figure 22: Comparison of path detection delay

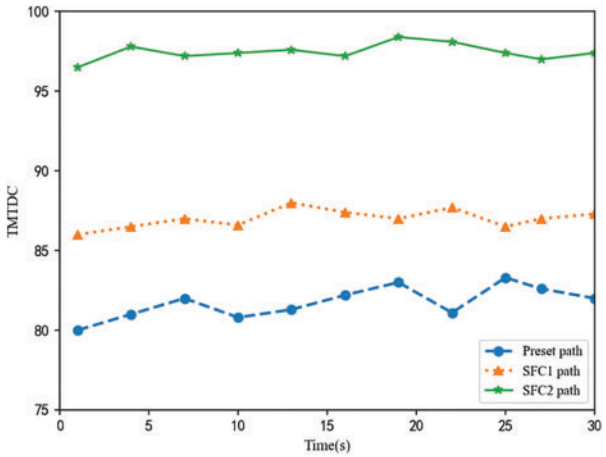


Figure 23: Comparison of TMTDC

4.6 Comparison of Knowledge Base before and after Updating

To assess the effectiveness and intelligence of the path selected after updating the knowledge base, we initially set the traffic path through SFC as the preset path and subsequently subjected it to a DRDoS attack within 30 s. Fig. 24 illustrates the path detection delays before and after the knowledge base update. Initially, the delay of the preset path fluctuates between 40 and 70 s (purple solid line). Upon adjusting the path, the delay range decreases to between 20 and 40 s (purple dotted line). This reduction in delay compared to the preset path demonstrates the improved efficiency of the path selected after updating the knowledge base.

Fig. 25 illustratesTMTDC before and after the knowledge base update. During the attack, the TMTDC for the preset path remains below 84 (solid line). However, after 15 s, the adaptive model selects an adjusted path based on current conditions (dotted line), resulting in a significantly higher TMTDC of 97, which is 15.5% greater than the preset path.

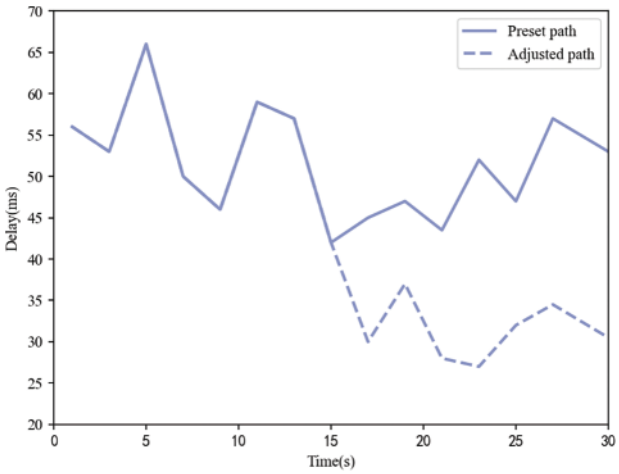


Figure 24: Comparison of path detection delay

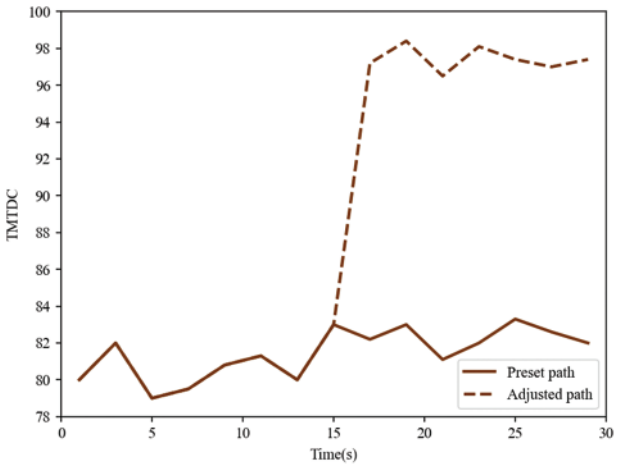


Figure 25: Comparison of TMTDC

Based on the analysis, the intelligent security service optimization method proposed in this paper achieves excellent online detection performance, with precision rate, recall rate, and F1 scores for each DDoS attack type consistently exceeding 96%. Compared to both preset and DQN-selected paths, it increases malicious traffic detection rate by an average of 12.4% and 4.6%, respectively, enhancing TMTDC by 18.1% and 11.5% while reducing detection delays. Furthermore, it enables feedback on detection outcomes, system resource utilization, and packet-level network traffic features to their corresponding knowledge bases, guiding future security policy reasoning updates.

5 Conclusion

We optimize intelligent security services based on a comprehensive network security knowledge base. It leverages rich data resources such as features of attack traffic, detection outcomes from NSF, and system resource utilization. Corresponding algorithms are developed to enhance feedback on security services, and to update and reason over the knowledge base. Real-time monitoring of malicious traffic allows for continuous knowledge updates and enables rapid, precise closed-loop optimizations. This approach facilitates real-time adjustment of path strategies. Experimental results demonstrate that the proposed scheme achieves SFC path selection, enhances system detection capabilities against DDoS attacks, and effectively reduces path latency for traffic traversing through SFC.

In future work, we aim to validate the efficacy of intelligent security service optimization methods based on the knowledge base in real network environments. We will also assess the impact of varying parameters in the GAT+DRL model on path selection to enhance path detection capabilities.

Acknowledgement: The authors thank all colleagues who provided us with moral support.

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

Author Contributions: Conceptualization, methodology, validation, formal analysis, data curation, writing—original draft, visualization: Xianju Gao; Project administration, funding acquisition: Huachun Zhou; Writing—review and editing: Huachun Zhou and Weilin Wang; Investigation, validation: Weilin Wang and Jingfu Yan. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data is available with the corresponding author and can be shared on request.

Ethics Approval: Not applicable.

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

References

[1] A. Mehmood, A. Khanan, and M. M. Umar, "Secure knowledge and cluster-based intrusion detection mechanism for smart wireless sensor networks," *IEEE Access*, vol. 6, pp. 5688–5694, 2017.

[2] 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 Netw.*, vol. 32, no. 6, pp. 152–159, 2018. doi: 10.1109/MNET.2018.1700467.

[3] M. Mittal, K. Kumar, and S. Behal, "Deep learning approaches for detecting DDoS attacks: A systematic review," *Soft. Comput.*, vol. 27, no. 18, pp. 13039–13075, 2023.

[4] K. Li, H. Zhou, Z. Tu, and H. Zhang, *CSKB: A Cyber Security Knowledge Base Based on Knowledge Graph*. Singapore: Springer, 2020, pp. 100–113.

[5] F. Liu *et al.*, "Construction of DDoS attacks malicious behavior knowledge base construction," *Telecommun. Sci.*, vol. 37, no. 11, pp. 17–32, 2021.

[6] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 494–514, 2021. doi: 10.1109/TNNLS.2021.3070843.

[7] Y. Jia, Y. Qi, H. Shang, R. Jiang, and A. Li, "A practical approach to constructing a knowledge graph for cybersecurity," *Engineering*, vol. 4, no. 1, pp. 53–60, 2018. doi: 10.1016/j.eng.2018.01.004.

[8] B. Hamid *et al.*, "A hierarchical key management method for wireless sensor networks," *Microprocess. Microsyst.*, vol. 90, 2022, Art. no. 104489.

[9] M. Alauthman, N. Aslam, M. Al-kasassbeh, S. Khan, A. Al-Qerem and K. -K. Raymond Choo, "Choo an efficient reinforcement learning-based botnet detection approach," *J. Netw. Comput. Appl.*, vol. 150, no. 11, 2020, Art. no. 102479. doi: 10.1016/j.jnca.2019.102479.

[10] Q. Shen, Z. Tu, K. Li, Y. Qing, and H. Zhou, "Online botnet detection method based on ensemble learning," *App. Res. Comput.*, vol. 39, no. 6, pp. 1845–1851, 2022.

[11] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "CorrAUC: A malicious Bot-IoT traffic detection method in IoT network using machine-learning techniques," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3242–3254, Mar. 2021. doi: 10.1109/JIOT.2020.3002255.

[12] S. Xianran and C. Jing, "Random forest feature selection for partial label learning," *Neurocomputing*, vol. 561, 2023, Art. no. 126870.

[13] H. Zhao, X. Xu, Y. Song, D. Lee, Z. Chen and H. Gao, "Ranking users in social networks with motif-based pagerank," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 2179–2192, 2019.

[14] W. -C. Yeh, W. Zhu, C. -L. Huang, T. -Y. Hsu, Z. Liu, and S. -Y. Tan, "A new BAT and PageRank algorithm for propagation probability in social networks," *Appl. Sci.*, vol. 12, no. 14, 2022, Art. no. 6858. doi: 10.3390/app12146858.

[15] V. Amelkin and A. K. Singh, "Fighting opinion control in social networks via Link recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Dis. Data Min.*, New York, NY, USA, 2019, pp. 677–685.

[16] W. Li *et al.*, "Security service function chain based on graph neural network," *Information*, vol. 13, no. 2, 2022, Art. no. 78. doi: 10.3390/info13020078.
[17] Y. Liu, H. Lu, X. Li, and Y. Zhang, "Dynamic service function chain orchestration for NFV/MEC-enabled IoT networks: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 8, no. 9, pp. 7450–7465, 2020. doi: 10.1109/JIOT.2020.3038793.
[18] S. Deng, M. Li, and H. Zhou, "Dynamic security SFC branching path selection using deep reinforcement learning," *Intell. Autom. Soft Comput.*, vol. 37, no. 9, pp. 2919–2939, 2023. doi: 10.32604/iasc.2023.039985.
[19] M. Li, S. Deng, H. Zhou, and Y. Qin, "Qin A path selection scheme for detecting malicious behavior based on deep reinforcement learning in SDN/NFV-Enabled network," *Comput. Netw.*, vol. 236, no. 1, 2023, Art. no. 110034. doi: 10.1016/j.comnet.2023.110034.
[20] Y. Shao, R. Li, B. Hu, Y. Wu, Z. Zhao and H. Zhang, "Graph attention network-based multi-agent reinforcement learning for slicing resource management in dense cellular network," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10792–10803, 2021. doi: 10.1109/TVT.2021.3103416.
[21] M. Li, H. Zhou, and Y. Qin, "Two-stage intelligent model for detecting malicious DDoS behavior," *Sensors*, vol. 22, no. 7, 2022, Art. no. 2532. doi: 10.3390/s22072532.
[22] Y. Li, M. Li, P. Dong, and H. Zhou, "Multi-type application-layer DDoS attack detection method based on integrated learning," *J. Comput. Appl.*, vol. 42, no. 12, pp. 3775–3784, 2022.
[23] L. Li, M. Li, H. Bi, and H. Zhou, "Multi-type low-rate DDoS attack detection method based on hybrid deep learning," (in Chinese), *Chin. J. Netw. Inform. Secur.*, vol. 8, no. 1, pp. 73–85, 2022.
[24] T. Yang, "Design and implementation of DRDoS attack detection based on machine learning," Beijing Jiaotong Univ., China, 2023.
[25] M. Sarhan, S. Layeghy, and M. Portmann, "Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection," *Big Data Res*, vol. 30, 2022, Art. no 100359. doi: 10.1016/j.bdr.2022.100359.
[26] W. Wang, H. Zhou, K. Li, Z. Tu, and F. Liu, *Cyber-Attack Behavior Knowledge Graph Based on CAPEC and CWE towards 6G*. Singapore: Springer, 2022.
[27] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, 2014. doi: 10.1016/j.compeleceng.2013.11.024.
[28] D. Ienco, R. Meo, and M. Botta, "Using PageRank in feature selection," in *Proc. Sixteenth Italian Symp. Adv. Database Syst., SEBD 2008*, Mondello, PA, Italy, 2008, pp. 93–100.
[29] W. Zhuo, "Research on feature selection methods based on random forest," *Tehnički Vjesnik*, vol. 30, no. 2, pp. 623–633, 2023.
[30] T. Yang and W. Wang, "Multi-class DRDoS attack detection method based on feature selection," *Res. Briefs Inform. Commun. Technol. Evol.*, vol. 7, pp. 173–187, 2021. doi: 10.56801/rebicte.v7i.127.
[31] X. Ni, H. Wang, L. Chen, and R. Lin, "Classification of aviation incident causes using LGBM with improved cross-validation," *J. Syst. Eng. Electron.*, vol. 35, no. 2, pp. 396–405, 2024. doi: 10.23919/JSEE.2024.000035.
[32] H. Holmstrm and J. E. S. Fransson, "Combining remotely sensed optical and radar data in KNN-estimation of forest variables," *For. Sci.*, vol. 49, no. 3, pp. 409–418, 2003. doi: 10.1093/forestscience/49.3.409.