mathematics

MDPI

Article

From Replay to Regeneration: Recovery of UDP Flood Network Attack Scenario Based on SDN

Yichuan Wang [1,2], Junxia Ding [1], Tong Zhang [1,*], Yeqiu Xiao [1] and Xinhong Hei [1,2]

[1] School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China; chuan@xaut.edu.cn (Y.W.); dingjunxia@stu.xaut.edu.cn (J.D.); xiaoyeqiu@xaut.edu.cn (Y.X.); heixinhong@xaut.edu.cn (X.H.)

[2] Shaanxi Key Laboratory for Network Computing and Security Technology, Xi'an 710048, China

* Correspondence: zhangtong@xaut.edu.cn; Tel.: +86-139-9186-3253

Abstract: In recent years, various network attacks have emerged. These attacks are often recorded in the form of Pcap data, which contains many attack details and characteristics that cannot be analyzed through traditional methods alone. Therefore, restoring the network attack scenario through scene reconstruction to achieve data regeneration has become an important entry point for detecting and defending against network attacks. However, current network attack scenarios mainly reproduce the attacker's attack steps by building a sequence collection of attack scenarios, constructing an attack behavior diagram, or simply replaying the captured network traffic. These methods still have shortcomings in terms of traffic regeneration. To address this limitation, this paper proposes an SDN-based network attack scenario recovery method. By parsing Pcap data and utilizing network topology reconstruction, probability, and packet sequence models, network traffic data can be regenerated. The experimental results show that the proposed method is closer to the real network, with a higher similarity between the reconstructed and actual attack scenarios. Additionally, this method allows for adjusting the intensity of the network attack and the generated topology nodes, which helps network defenders better understand the attackers' posture and analyze and formulate corresponding security strategies.

Keywords: SDN; network attack; scenario reconfiguration; probabilistic model; topology reconfiguration model

MSC: 68M25

Citation: Wang, Y.; Ding, J.; Zhang, T.; Xiao, Y.; Hei, X. From Replay to Regeneration: Recovery of UDP Flood Network Attack Scenario Based on SDN. *Mathematics* **2023**, *11*, 1897. https://doi.org/10.3390/math11081897

Academic Editors: Zhiming Cai, Wencai Du, Zhihai Wang and Zuobin Ying

Received: 21 March 2023
Revised: 7 April 2023
Accepted: 13 April 2023
Published: 17 April 2023

1. Introduction

With the rapid development of computer networks and internet technologies, networks have permeated into various fields, facing increasingly complex security challenges. Network attacks are often recorded in the form of Pcap data, which contains numerous attack details and features. Traditional data processing methods often overlook many attack behavior features, leading to resource waste. Therefore, this paper proposes an SDN-based UDP flood network attack scene reconstruction method. By reconstructing network attack scenarios, the original data can be restored, and new mixed data can be generated by adjusting the intensity of network attacks, topology nodes, and other types of network attacks. This enables network defenders to better understand the attacker's posture, analyze the monitored data and information, and formulate corresponding security strategies to enhance their ability to respond to network attacks.

Due to the openness of the internet, the inherent imperfections of network protocols, and various application software, devices on the network are vulnerable to potential danger. Nowadays, the UDP protocol is widely used in networks and various applications. However, a network utilizing the UDP protocol is easily targeted in attacks since the sender does not need to establish a connection via three handshakes, while the receiver

has to receive and process the packet. Therefore, it is crucial to protect the network from attacks exploiting the vulnerabilities of the UDP protocol. UDP flood is one of the most common attacks targeting the UDP protocol, which typically targets DNS servers, RADIUS authentication servers, or streaming video servers by flooding them with a large number of small UDP packets [1–3]. Such attacks are often directed towards a random port on the target, and the victim system must analyze the incoming data to determine which application service has requested it. This makes it difficult for defenders to protect the network from such attacks. In fact, a dynamic game process occurs between attackers and defenders, where defenders develop corresponding security strategies in response to changes in attackers' techniques, while attackers constantly research new techniques to evade network security protection and achieve their attack objectives. Timely detection of potential security threats to the network is crucial for defenders. Therefore, this article proposes a network attack scene reconstruction method, which lays a solid foundation for defending against network attacks from the attacker's perspective. To achieve better results in network attack scene reconstruction, this article suggests using software-defined networking (SDN) for scene reconstruction.

SDN offers dynamic programmable network configuration, which improves network performance and management efficiency, and enables network services to provide flexible customization capabilities similar to those of cloud computing. In addition, SDN decouples the forwarding plane of network devices from the control plane, enabling the controller to manage network devices, orchestrate network services, and schedule service traffic [4]. SDN overcomes the limitations of traditional networks and offers benefits such as low cost, centralized management, and flexible scheduling [5]. In SDN testing, Mininet is commonly used as a testbed as it enables easy creation of an SDN-enabled network, with each host working like a real computer. Programs can launch applications and send packets to the Ethernet ports, which are received and processed by switches and routers. SDN also supports complex network topologies, allowing the addition of new features to the network, testing, and easy deployment into real hardware environments [6]. Therefore, SDN-based attack recovery can provide a realistic scenario for a network that is under attack.

In response to the challenge of lacking the reconstruction of traffic rebirth and elasticity in existing network attack scene reconstruction, this paper proposes an SDN-based UDP flood network attack scene reconstruction method. This method can automatically create network topology and regenerate network traffic using sample Pcap packets. Additionally, this solution allows users to modify any component in the virtual network and adjust the network attack-related parameters and intensity to meet the needs of different scenarios. The main contributions of this paper are threefold:

- Existing approaches to network attack scenario recovery lack the ability to regenerate real network attack traffic, and the research in this paper is one of the first articles to fill this gap.
- The method proposed in this paper can automate the network attack scenario recovery, it is studied for packet delivery probability events, and it can simulate network attack scenarios more realistically.
- This paper can change the network topology nodes and network attack intensity based on the network attack scenario recovery, which can bring convenience to the network attack defenders to better detect and defend against malicious attacks.

The remainder of this paper is structured as follows. Section 2 provides an introduction to the definition of SDN and related research. Section 3 presents the SDN-based topology reconfiguration model for network attack scenarios and the reconfiguration probability model. Section 4 analyzes the constructed models, while Section 5 offers a comparative analysis of the experiments from both qualitative and quantitative perspectives. Finally, Section 6 concludes our work.

2. Related Work

Regarding the problem of network attack scene recovery, we have reviewed the relevant literature from the past 5 years and broadly divided network attack scene recovery methods into four main types. The first type is network attack scene recovery based on traffic replay. The second type involves using graph networks for network attack scene recovery. The third type uses correlation analysis for network attack scene recovery. The fourth type encompasses other methods for network attack scene recovery. Details on each category are provided below:

(1) Network attack scene recovery method based on traffic replay.

A multi-node traffic replay method was proposed by [7]. This method designs a self-selected IP mapping algorithm to construct an IP mapping between the target network and the existing network in order to reproduce the interaction between the existing network nodes. The method is effective in aggregating large flows and achieving high similarity in playback timing sequences and bandwidth and can be used to reproduce real network scenarios for network device testing and network security experiments.

A deterministic TCP replay method for performance diagnosis was proposed by [8]. This method can faithfully replay packet traces using a low-overhead timer and an efficient file access method, capturing all interactive traffic in TCP connections for all hosts and replaying selected packets to reproduce performance issues at low overhead. However, these methods can only fully reproduce the last attack and do not account for weak points, reinforcement points, or chain events. In contrast, the network attack scenario reproduction method presented in this paper can automatically create the network topology and execute the attack based on the Pcap packets, allowing for manual addition of devices such as hosts and attack relationships between devices as needed.

A virtual network traffic replay method built on a network simulation platform was proposed by [9]. This method is capable of performing IP mapping-based virtual node replay of any traffic captured or generated by one or more interfaces. However, it is unable to simulate changes in attack strength based on the original data.

A precise traffic replay method based on interaction sequences and timestamps was proposed by [10]. The method achieves high consistency in playback time, but does not regenerate the data, rather it simply replays the network traffic at the recorded time.

(2) Network attack scene recovery based on graph networks.

The graph-based fusion module (GM) to fuse all captured attack information to reconstruct a multi-step attack scenario approach was proposed by [11]. The approach uses a weighted directed graph to model the network communication and a fusion algorithm to update it. The weighted directed graph with attributes is then used to fuse the attack information and reconstruct the attack scenario. However, some manual processing is still required to reconstruct the attack scenario when dealing with fake IP addresses.

The use of graph theory to construct a system connection matrix and network path function for forming a network topology model was proposed by [12]. The method analyzed its practicality in optimal routing design and verified the feasibility of the model step by step based on its structural characteristics, laying the foundation for modeling and simulating tactical communication systems. However, further verification is needed to ensure its conformity with the characteristics of the network topology.

A reconstruction method for a big data-based attack scenario was proposed by [13], which utilizes temporal concept maps and neural networks. This method allows the reconstruction of complex attack scenarios based on large amounts of data. In addition to tracing the entire attack scenario, a temporal concept graph is used to represent the big data and the dependencies between them. The model is able to classify possible attack scenarios in real time using RBF networks and converge to the most potential attack scenarios with the support of Elman networks. However, the processing of the proposed model is initiated based on the collection of event sets generated by traditional intrusion detection systems. These systems may not be suitable to represent alerts in big data environments.

A real-time mining method for reconstructing multi-step attack scenarios was proposed by [14]. This method constructs a directed graph through association analysis by analyzing the alarm logs from intrusion detection systems to achieve the construction of attack trees. This method can combine attack patterns between different hosts and reduce false alarms. However, this method only provides an abstract description of the attack process to achieve scenario reconstruction, and does not build network topology structures and regenerate traffic. In contrast, this paper analyzes Pcap data packets and reconstructs attack scenarios by creating network topology and regenerating traffic based on the analysis results. Moreover, this method can transform the original scenario based on the reproduction.

(3) Network attack scene recovery based on correlation analysis.

A attack scenario reconstruction method based on causal knowledge and spatio-temporal correlation has been proposed by [15]. This method utilizes a causal knowledge network to conduct correlation analysis on alerts from multiple dimensions including causality, time and space, in order to restore the complete attack penetration process of the attacker and reconstruct the attack scene. This method can discover potential hidden relationships to a certain extent, but it does not regenerate network attack traffic.

A method of reconstructing attack scenarios based on association analysis was proposed by [16]. It emphasizes the temporal relationship between alerts from a holistic perspective of the network and associates aggregated alerts to build the attack scenario. This method can restore attack relationships to a certain extent but does not reproduce network attack traffic.

(4) Other methods for network attack scene recovery.

An efficient reconstruction method for advanced persistent threat (APT) attacks based on the hidden Markov model was proposed by [17]. This method describes the action sequence based on the temporal order or the conditions reached by the attack, uses data association and advanced probabilistic methods to mine the hidden APT attack phases, and finally reconstructs the attack path. However, it only provides the network attack paths and does not create any network topology.

A RouteNet model was proposed by [18]. This is a novel network model based on SDN's graph neural network (GNN). This model can accurately estimate the delay distribution and packet loss per source/destination by understanding the complex relationships between topology, routing, and incoming traffic.

A network attack probability analysis method was proposed by [19]. The model takes into account the severity of vulnerabilities, attack scenarios, and various potential participants and their motives. Based on the results obtained from the model, the most likely attack scenarios are further inferred.

SDN has been applied to combat DDoS attacks since it has logically centralized control, network programmability, and separation of control and forwarding. Reference [20] proposes a real-time DDoS detection attack method for SDN controllers. Reference [21] analyzes simulated DDoS attacks in an SDN environment. Reference [22] present a flexible SDN-based architecture, which identifies and mitigates low-rate DDoS attacks via machine learning. Although the aforementioned works show that SDN is available for the analysis of network attacks, SDN has not been fully applied to defend networks against UDP flood. Therefore, we will reconstruct UDP flood scenario with the aid of SDN in this paper.

In conclusion, the network attack scenario recovery method based on traffic replay can replay existing traffic, but it cannot generate new traffic or change network topology configuration, structure, or enhance/reduce network attack intensity. The graph-based network attack scenario recovery method can display the attack relationship of the network in a graph to some extent and restore the attack relationship, but it does not reproduce traffic. The correlation-based network attack scenario recovery method can discover potential hidden relationships to some extent but also does not reproduce traffic. Other methods for recovering network attack scenarios focus only on recovering some scenarios in the attack

path, relationship, or steps without reproducing traffic. Therefore, this article proposes an SDN-based UDP flood network attack scenario recovery method, which can reconstruct the topology, reproduce traffic, scale network attack intensity, and change experimental topology in the recovered scenario.

3. Models

In this section, a topology reconstruction model, a probabilistic model, and an attack sequence model are established regarding the process of reconstructing a network attack scenario. The network attack scenario topology reconstruction model enables the creation of a topology that accurately and meaningfully reflects the true network topology as far as possible based on the available information. The probabilistic model and the attack sequence model are used to generate network attack commands for all events (including small sample events) to the greatest extent possible. The end result is that the regenerated data are highly similar to the sample data.

3.1. Network Attack Scenario Topology Reconfiguration

Due to the flexibility and programmability of SDN networks, this paper uses SDN networks for topology creation. The SDN network topology is created mainly through the Mininet platform, and the network devices are mainly selected from remote controllers, OpenFlow switches, hosts, etc. The network topology reconstruction model integrates SDN connection line attributes into the Waxman model and uses information such as IP addresses and MAC addresses from Pcap packets to configure the network. The goal is to create a topology structure that is as accurate and meaningful as possible in order to reflect the real network topology. According to the rules of network topology building, this paper abstracts the process of building network topology and forms a reconfiguration model of network attack scenarios. To better illustrate this, the following definitions are provided.

Definition 1. *The connection line attribute between nodes. A connection line attribute is an n-tuple* $(b_1, b_2, b_3, \ldots, b_n)$*, where* b_j $(1 \leq j \leq n)$ *denotes the jth attribute of the connection line between nodes and the connection line attribute* $q = (b_1, b_2, b_3, \ldots, b_n)$ *between nodes of the network.*

In this paper, the connection line property between nodes is defined as a 5-tuple, $q_j = (bandwidth, delay, loss, max\ queue\ size, jitter)$. Where *bandwidth* refers to the transmission speed of the link that connects two nodes in the network topology, expressed in Mbps (megabits per second), with a default value of 10. The "*delay*" parameter represents the delay of a link in the network topology, which is the time it takes for a packet to travel from one host to another. This delay includes not only physical delay but also other factors such as buffering, queuing, and transmission delays, and is measured in milliseconds (ms). The default value is 0. The "*loss*" parameter represents the packet loss rate of the network link, which refers to the proportion of lost packets during data transmission. This parameter is often used to simulate noise, interference, and other situations in the network to more realistically simulate the network environment. The value range is from 0 to 1, with a default of 0. The "*max queue size*" parameter refers to the maximum length of the queue on a network link, which is used to control the number of packets on the link and prevent network congestion. When a packet arrives at a link, if the queue is already full, packet loss may occur. Therefore, setting an appropriate queue length is crucial for controlling network congestion and reducing packet loss. The unit is in packets and the default value is infinite. "*Jitter*" refers to the variation in delay between adjacent data packets. Normally, data packets arrive at the receiving end at regular intervals, but congestion and packet loss in the network can cause fluctuations in packet delay. Generally speaking, the smaller the jitter, the more stable the network transmission, and the larger the jitter, the less stable the network transmission. The unit is milliseconds and the default value is 0.

The construction of the network topology in this paper is based on the Waxman model, which has a core idea: all nodes are randomly (or according to the heavy-tailed distribution) placed in a plane; considering each pair of nodes, an edge is added between the node pairs (u, v) with a certain probability $P(u, v)$ (also called edge probability). The Waxman model is a static model that can only simulate the number of nodes, edges, and distances between nodes in network topology. However, since network topology models are merely simulations of real network topology structures, a simplistic and fixed model cannot effectively simulate a real network as the network evolves and QoS requirements arise. Therefore, the Waxman model serves only as a reference for topology research and needs to add connection attributes, such as bandwidth, latency, cost, and packet loss rate (QoS parameters), on existing connections to better simulate a real network. Thus, improvements need to be made to the Waxman model.

According to the improved Waxman model proposed in this paper, the attribute of connecting lines between nodes of 5-tuples will be added, and the distance between any two node functions will be taken as the independent variable to calculate the probability of direct connection between two nodes. The probability function of the model is given by

$$P(u, v) = \alpha e^{-\frac{d}{L\beta}}, \tag{1}$$

where $\alpha \geq 0$, $\beta \leq 1$; d is the European distance from u to v; L is the longest distance between two points. When increasing α, the model will have more short edges, longer hop diameter and shorter length diameter. Increasing β will increase the proportion of long edges in the model.

3.2. Reconstructing Probabilistic Models for Network Attack Scenarios

During the process of traffic regeneration, there are many uncertain and complex factors in the network. Introducing probability models can reduce the difficulty of traffic regeneration to some extent, decrease errors, and improve the effectiveness of traffic regeneration.

In the network attack scenario reconstruction probability model, this paper divides the behavior of each host sending packets into three categories: not sending packets, sending packets in the form of forged IPs, and sending packets with the real IPs of hosts. If there are n hosts, the probability of not sending packets from host i to host j (where i denotes the number of the host sending packets, j denotes the number of the host receiving packets, and all host numbers are unique) is P_{ij}^1, the probability of sending packets in the form of forged IP is P_{ij}^2, and the probability of sending packets with the real IP of the host probability is P_{ij}^3, where ${P_{ij}}^1 + {P_{ij}}^2 + {P_{ij}}^3 = 1$, but in a network attack, P_{ij}^3 is almost 0. In particular, for small sample events, $0 < {P_{ij}}^2 \ll {P_{ij}}^1 < 1$. According to the network scenario reconstruction probability rules proposed in this paper, the network attack sequence scenario reconstruction probability model is shown in Figure 1, where the ... in the figure indicates the label of the host.

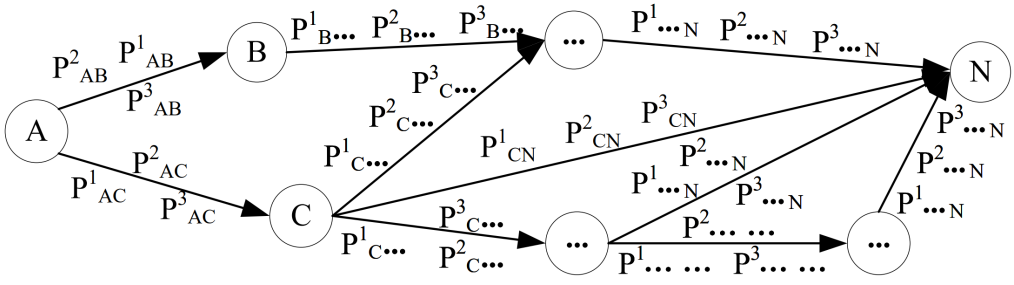

Figure 1. Host Probability Distribution Chart.

Additionally, for P_{ij}^{2} , there is a different case for host i. That is, when host i sends packets by forging a different IP address or forging a different port, etc., it is considered to be a different action. This paper gives the different actions statistics of their probability of occurrence. These different actions constitute probabilities satisfying ${P_{ij}}^{2} = {P_{ij}}^{21} + {P_{ij}}^{22} + \ldots + {P_{ij}}^{2n}$, where ${P_{ij}}^{2k}(1 \leq k \leq n)$ is a certain action of this host, n means that there are n actions in this host, but the value of n may be different for different hosts. The action probability distribution diagram of a host sending packets is shown in Figure 2.

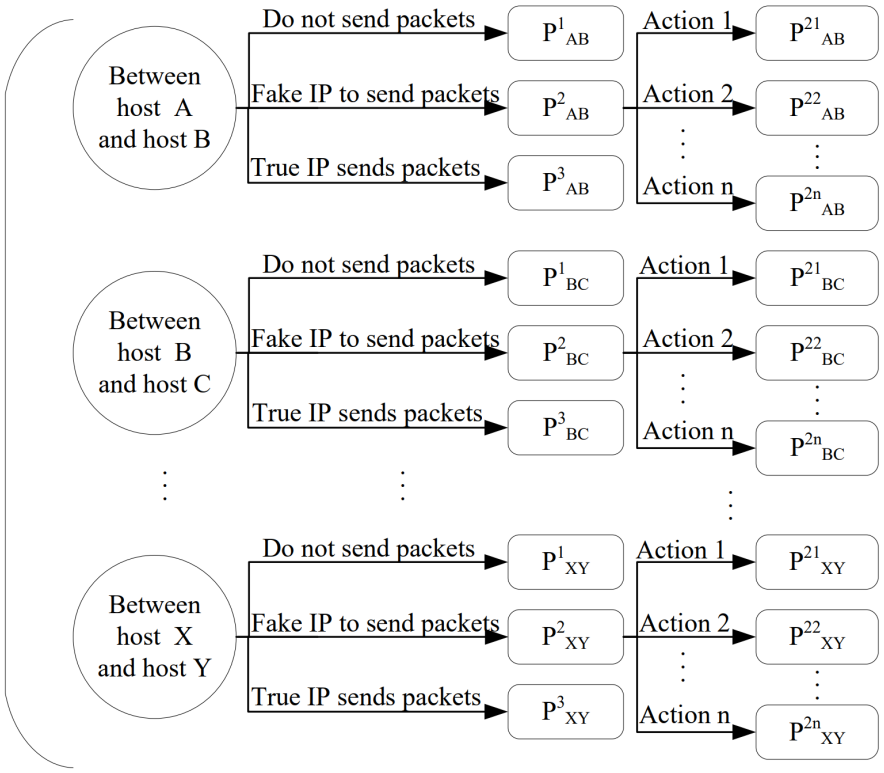

Figure 2. Probability diagram of contracting action event.

3.3. Network Attack Relationship Sequence Generation

A key point of network attack scene reconstruction is to build a set of action event attack relationship sequences, that is, from the set of action events (set if action events, SAE) to find all the highly relevant attack relationship sequences, constituting a set of attack scene sequence (set of attack scene sequence, $SASS$).

From the characteristics of network attacks, it is known that there are different network attack actions in network attacks. Each action has a purpose to reach the destination address of that attack, and these different action behaviors can be reflected by the preprocessing of Pcap packets. In order to get the attack corresponding attack commands from the preprocessing of Pcap packets, this paper first preprocesses the original Pcap packets, standardizes the format of the packets, and classifies and converts the data into a collection of data containing the network attack relationships. To better illustrate the content of this paper, the following definitions are given.

Definition 2. *Action events. An action event is an n-tuple* $(c_1, c_2, c_3, \ldots, c_n)$*, where* $c_i(1 \leq i \leq n)$ *denotes the ith attribute of the action event, and note that the action event* $P = (c_1, c_2, c_3, \ldots, c_n)$.

Definition 3. *Network attack relationship sequence. A set of strongly consistent action events in chronological order is called a network attack relationship sequence, denoted as* $NARS$. $NARS =< e_1, e_2, e_3, \ldots, e_n >$, *satisfies* $e_i.timestamp < e_j.timestamp(1 \leq i < j \leq n)$ $\&\&e_i.srcMAC == e_j.srcMAC \&\& e_i.dstMAC == e_j.dstMAC$.

In this paper, the action event is designed as an 8-tuple, $P_i = (timeline, srcIP, dstIP, srcMAC, dstMAC, dType, srcPort, dstPort)$. where $timeline$ denotes the timestamp of the action event; $srcIP$ and $dstIP$ denote the source and destination IP addresses of the action event, respectively; $srcMAC$ and $dstMAC$ denote the source and destination MAC addresses of the action event, respectively; $dType$ denotes the type of the attack; $srcPort$ and $dstPort$ denote the source and destination ports of the action, respectively.

The network attack relationship sequence is crucial for the reproduction of network attacks through which the attack relationship between hosts can be known and form cyber attack orders. The core of the algorithm: First, the MAC address pairs of action events are extracted through the algorithm. Then, according to the association relationship between address pairs, it iteratively judges whether the current action event belongs to a certain type of existing action event collection. If it exists, it will be directly placed in the appropriate location of the current action event collection. If it does not exist, it will create a new action event collection to place it. After all the action events are placed in the appropriate positions of the action event sets of different categories, all the action event sets have been orderly, all the action event sets are attack relationship sequences at the same time, thus finally completing the construction of network attack relationship sequence sets. The network attack relationship sequence construction algorithm is presented as Algorithm 1.

Algorithm 1: Constructing Network Attack Relationship Sequence Sets $(CNARS)$ based on Action Event Sets (SAE)

Input: Collection of action events $SAE = \{e_1, e_2, \ldots, e_n\}$
Output: A collection of cyber attack relationship sequences $CNARS = \{NARS_1, NARS_2, \ldots, NARS_n\}$

```
Create CNARS = null;
while SAE is not ∅ do
    temp = SAE.first and delete SAE.first;
    for NARS_i in CNRAS do
        if (temp.srcMAC, temp.dstMAC couple is in NARS_i.MACSet) then
            index = f(temp, NARS_i); /*This function is used to find the index of
              the last action event in NARS_i with a MAC pair equal to temp.*/
            if (index == NARS_i.size − 1) then
                add temp to NARS_i;
                add temp.MACnew couple to NARS_i.MACSet;/*NARS_i.MACSet
                  can be automatically de-duplicated*/
                GoTo 2;
            else
                Create NARS_inew;/*Take the NARS_i 0-index items and create a new
                  attack relation sequence NARSinew derived from NARS_i*/
                Create NARS_inew.MACSet;
                add temp to NARS_inew;
                add temp.MAC couple to NARS_inew.MACSet;
                CNARS = CNARS∪NARS_inew;
                GoTo 2;
    Create NARS_new;
    add temp to NARS_new and create NARS_new.MACSet;
    add temp.MAC couple to NARS_new.MACSet;
    CNARS = CNARS ∪ NARS_new;
Return CNARS
```

The main body of the algorithm is run by double layer iteration. Using induction, it can be concluded that when the size of the set of action events is n, the number of times the subject is executed in its worst case of operation is approximately $n(n-1)/2$, so the time

complexity of the algorithm is $O(n^2)$. The body of the algorithm is temporarily the space complexity of the algorithm, which is $O(n)$ when the size of the set of action events is n.

Using the SDN-based network attack scenario topology model, nodes can be connected together in an orderly way, and with the attributes of network topology connecting lines, the network attack scenario topology reconstruction model is completed.

4. Model Analysis

4.1. Relevance

This paper uses Pearson's correlation coefficient to evaluate the quality of regenerative flows, which reflects the degree of correlation between the variables. Pearson's correlation coefficient is calculated by the product-difference method, which reflects the correlation between two variables based on the deviation of their mean values from their respective values. The higher the value of the correlation coefficient is, the higher is the correlation between the two sets of data and the higher the fidelity of the regenerative flows.

In this paper, the experimental results from Section 5.5 were subjected to similarity calculation using Pearson correlation coefficient. The results showed that the similarity score was above 90% for various experimental data such as IP frequency distribution, protocol information, and changes in attack flow over time. This indicates a high degree of similarity in the experimental data analyzed. The specific correlation coefficients are calculated in the experimental section of Section 5.

4.2. Authenticity

The network topology reconfiguration model proposed by this paper is not a simple, fixed network topology model. It adds the properties of connection lines based on the Waxman model, such as adding parameters such as connection bandwidth, connection delay, connection maximum queue size, and connection packet loss rate. The network topology with these connection parameters can meet the QoS requirements proposed in the network. Mininet can easily simulate the operation and architecture of networks in real environments, mainly by using the namespace mechanism of the Linux kernel. In layman's terms, the namespace mechanism is to be able to simulate a space for each virtual device in the network. The experiments conducted on Mininet can be seamlessly moved to the real environment. This satisfies the first step in reconfiguring the network attack scenario: topology reconstruction. The probabilistic model is then added, and the attack made by this model is not simply a completion of the attack event but is able to simulate a small sample of events at the time of the attack with a small sample of packet sending behavior. In this way, the network attack scenario recovery performed is no longer just a network traffic replay but a high degree of reduction in the attack scenario. The generated packets can achieve a high degree of match with the sample packets, as demonstrated in the experiments of Section 5.

4.3. Efficiency

Network attack scene recovery includes three steps: data preprocessing, network topology creation, attack command generation and attack implementation. Assume that the data preprocessing time is T_1, the network topology creation time is T_2, and the attack command generation and implementation time is T_3. Then the total time t of network attack scene recovery is given by

$$T = T_1 + T_2 + T_3. \tag{2}$$

In the whole process of network attack scene recovery, the consumption of time and space is mainly reflected in data preprocessing. Before the network topology is generated, the preprocessing part has already prepared the configuration information such as the IP address when the network topology is created, and the connection line attributes that meet the QoS requirements. Because Mininet itself has the characteristics of rapid reconfiguration and restart, it takes very little time to create the network topology. A lot of experiments

show that the process is at a constant, that is, $T_2 \approx 1.5$ s. Before the attack command is generated and implemented, the data preprocessing stage will also compare the data packet information with the rule base to obtain the corresponding attack command parameters, which need to be spliced and the attack started. Once the attack is started, it only takes about 40 s, and the CPU and other occupancy rates of the network will reach 100%. So overall, the speed of network attack scene reconstruction is still very fast. The experiments in Section 5.5 show that processing 500 pieces of data takes less than 8 s, and even with 100,000 pieces of data, it only takes less than 1 min. The specific time statistics in the experiments are detailed in Section 5.

4.4. Scalability

The experimental approach in this paper not only enables the replication of a network attack scenario but also allows for the addition or reduction of network attack intensity, changes to network topology nodes, etc. on the basis of this experiment. Through the replication and extension of the attack scenario, it is possible to gain insight into the network attack posture view, analyze the data and information monitored during the attack, and formulate corresponding security strategies. It can bring convenience to network attack defenders to better detect and defend against malicious attack phenomena. Specific experiments are described in Section 5.

The model is implemented with the idea of high cohesion and low coupling. The necessary interfaces are reserved in the process of model implementation, which greatly improves the scalability of the model and facilitates subsequent expansion. Based on the scalability of the model, the model can be replicated for more types of network attack scenarios, thus further enhancing the compatibility of the model in the future. The model analyzes and splits the data in the process of implementation and constructs pluggable tool libraries such as custom function libraries and attack relationship libraries. They are highly reusable and lay the foundation for the subsequent expansion of the model.

5. Experimental Analysis

5.1. Development Environment

The experimental environment in this paper uses Ubuntu operating system, version 21.04; the network topology is created based on the SDN network topology of the Mininet platform; the controller used is the Ryu controller. The specific environment configuration for the intelligent reconfiguration of the SDN-based network attack scenario is shown in Table 1.

Table 1. Environment Configuration Details.

Environment Matching	Usage Details
CPU	Intel(R) Core(TM) i5-10400F CPU @ 2.90 GHz 2.90 GHz
GPU	NVIDIA GeForce GTX 1660 SUPER
Operating system	Ubuntu 21.04 64-bit operating system
Memory	16.0 GB
Debugging environment	Mininet 2.3.0, Ryu 4.34
Development language	Python language
Packet capture tool	Wireshark

5.2. Introduction to the Simulation Environment

The environment for this experiment is the SDN network simulation environment tool Mininet with Ryu.

Mininet is a process virtualisation network simulation tool developed by Stanford University based on the Linux Container architecture [23]. It can be used to create a virtual network containing hosts, switches, controllers and links with OpenFlow support for switches and highly flexible custom software-defined networks [24]. With this platform,

we can easily simulate the network operation and architecture in a real environment. In addition, Mininet combines many of the advantages of emulators, hardware test beds and simulators [25]:

- Comparison with emulators: fast startup; large scalability; more bandwidth provision; easy installation and easy to use.
- Comparison with emulators: can run real code; easy to connect to real networks.
- Comparison with hardware testbeds: cheap; fast reconfiguration and restart.

There are three ways to create a network topology through Mininet. First, create a basic network topology with a quick command and then add the required nodes and node information through the command line if needed. Second, open the visualization tool MiniEdit; after opening the tool, you can select the controller, OpenFlow switch, host, and other tools as needed. The controller is usually configured as a remote controller, i.e., Ryu controller; after drawing the topology, you can export it as a .py file via File -> Export Level2 Script. Next time, you can continue to open and edit it next time. Third, write a Python script file directly through a compiler or editor to create a network topology.

Ryu is an open-source SDN controller led by Nippon Telegraph and Telephone Corporation. It provides users with a flexible and programmable network control interface while enabling simple and logical centralised control of thousands of OpenFlow switches [26]. As a platform for building SDN applications, Ryu is based on the Python language for development, so it is simple and easy to use for novices. Ryu, a rising star among SDN controllers, is now widely used in the industry [27].

The method used to create the network topology in this article is the third one, written directly by writing a Python script file. The controller used is the Ryu controller.

5.3. Datasets

The data used for the experiments in this paper is the 2019 Canadian Institute for Cyber Security Dataset (CIC-DDoS2019). The data contains benign and up-to-date common distributed denial-of-service (DDoS) attacks in the CICDDoS2019 dataset, which is extremely similar to real-world data (Pcap) [28]. It also includes the results of CICFlowMeter-V3 network traffic analysis using tagged flows based on timestamps, source and destination IPs, source and destination ports, protocols, and attacks (CSV files). Using their proposed B-Profile system, Sharafaldin et al. (2016) characterized the abstract behavior of human interactions and generated natural benign background traffic in the proposed testbed. The dataset is collated once a day, and the raw data, including network traffic and event logs (Windows and Ubuntu event logs) for each machine, are recorded daily. For the feature extraction process of the raw data, 80+ dimensional features were extracted using CICFlowMeter-V3 and saved as CSV files for each machine. For this dataset, abstract behaviors of 25 users were constructed based on HTTP, HTTPS, FTP, SSH, and email protocols, and the dataset is used by universities, private companies and independent researchers around the world [29].

5.4. Experimental Results

In this paper, recovery of UDP flood network attack scenario is based on SDN. The Ryu controller is first started and the program is run to process the Pcap packets through data preprocessing. The network attack scenario topology reconstruction model is then used to automatically compose the corresponding network topology, while information such as the type of attack and the attack relationship is derived from the inter-node information. Based on this information and the network attack scenario reconstruction probability model, corresponding attack commands are automatically generated. Finally, the automated replication of the corresponding attack scenarios is completed and a large amount of attack data is regenerated.

In this paper, we take the CICDDoS2019 dataset from the Canadian Institute for Cybersecurity Research as an example. First, a Pcap packet is selected from this dataset, the Ryu controller is opened, and the corresponding program is run, which automatically parses

and preprocesses the sample Pcap packet by running it to generate the corresponding four intermediate files. These four intermediate files have a corresponding role in the subsequent creation of the network topology and the generation of the attack script. The information of the corresponding intermediate files is shown in Table 2.

Table 2. Description of Intermediate Documents.

File Name	File Content	Document Description
Sum.txt	srcMAC,dstMAC,srcIP,dstIP	This file is used to store the source MAC address, destination Mac address, source IP address, destination IP address of the packet
SrcIP.txt	srcIP	This file is used to save the source IP address after de-duplication
DstIP.txt	dstIP	This file is used to save the destination IP address after de-duplication
Mac.txt	Mac	This file is used to save all Mac address information after de-duplication
MacRelation.txt	(srcMAC,dstMAC)	This file is used to save the source Mac address and destination Mac address as a whole after de-duplication
IpRelation.txt	(srcIP,dstIP)	This file is used to save the source IP address and destination IP address as a whole after de-duplication

Based on the information files of the three intermediate files, SrcIP.txt, DstIP.txt, and Mac.txt, the number of hosts, switches, etc., are calculated. The network attack scenario topology reconstruction model is then used to automatically compose the corresponding network topology. The network topology for this sample is shown in Figure 3, where the topmost person represents the attacker, the small blue circles represent the attacker, and each dotted circle represents a different identity of the attacker. The different hosts in this experiment represent multiple IP addresses, and the small ovals represent all but one identity shared by that host.

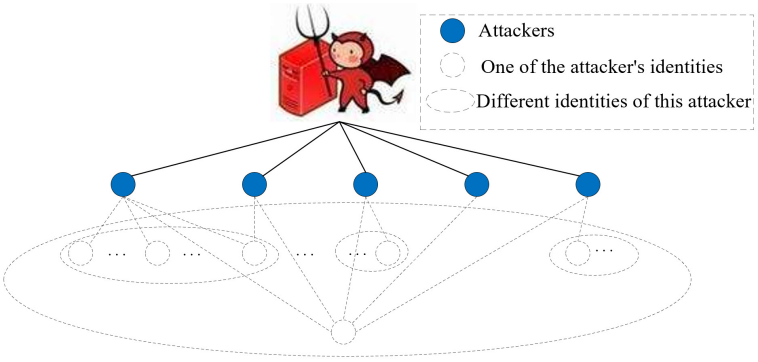

Figure 3. Network Topology.

After the network topology is generated, in this paper, we run the generated network attack script on the network topology to execute the attack and count the network attack traffic during the network attack. In this case, this paper counts the network attack for 60 s, and the number of sFlow bytes over time is shown in Figure 4. In the first 5 s, the attack script is executed, the attack starts to proceed, and the rate at which packets are sent begins to grow rapidly. After the attack lasts for 5 s, almost every attacker is in working condition and the rate of packets sent by the attacker increases slowly. Until 25 s, the attack rate almost reaches the set peak, and after 25 s, the packet rate is in a stable fluctuation.

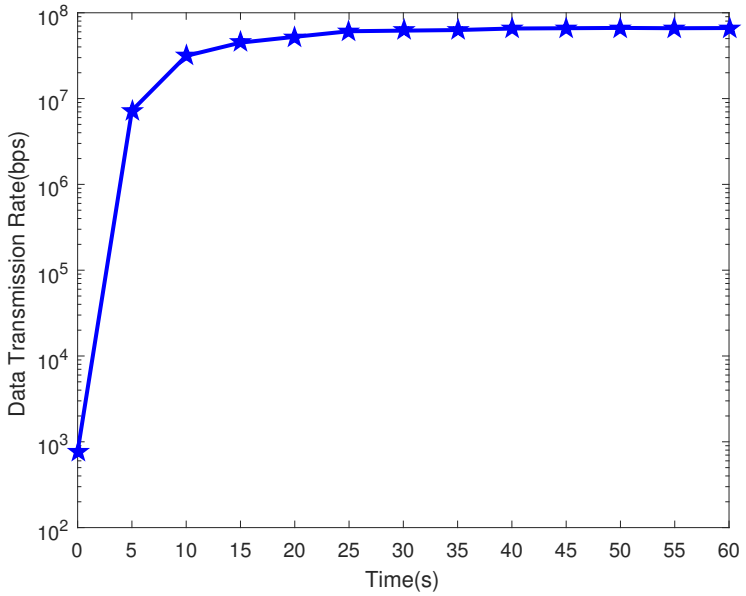

Figure 4. Change Diagram of sFlow Bytes in Network Attack Process.

During the attack, there is a direct correlation between sFlow bytes and network throughput rate. As the sFlow bytes increase, the network throughput rate also keeps increasing. That is, it increases rapidly from 0 to 5 s and slowly from 5 to 25 s until 25 s, when the attack rate almost reaches the set peak. After 25 s, the sFlow packets are also basically at a stable value because the packet rate is in a stable state. In this paper, we counted 60 s of network attacks, and the sFlow packet variation over time is shown in Figure 5.

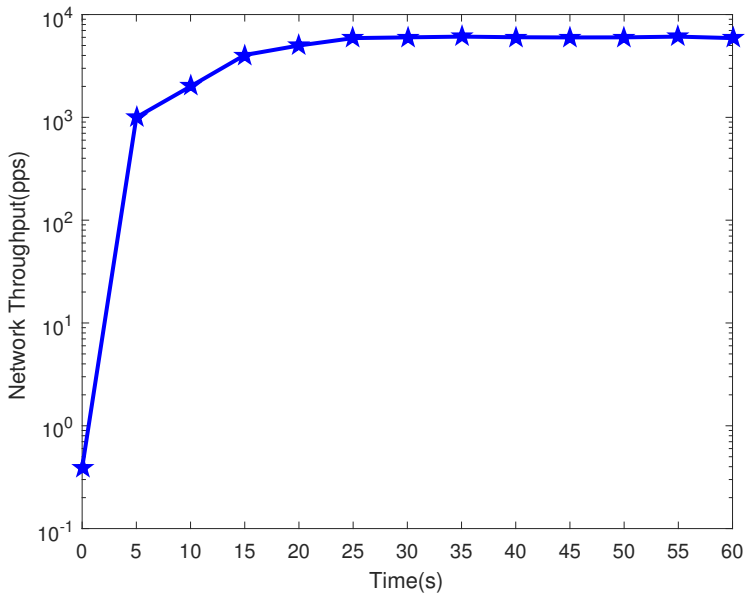

Figure 5. Change Diagram of SFlow Packets in Network Attack Process.

During the network attack, the rapid increase of sFlow bytes and sFlow packets make the transmission traffic grow rapidly, and the CPU occupancy and memory occupancy also increase significantly from 0 to 30 s. After that, with sFlow bytes and sFlow packets at a stable value, the attack is also at a critical moment, and the CPU occupancy reaches 100%. After 20 s, the packets are continuously sent. After 50 s, the CPU occupancy and memory occupancy reach the maximum value, and the CPU and memory change during the network attack, as in Figure 6.

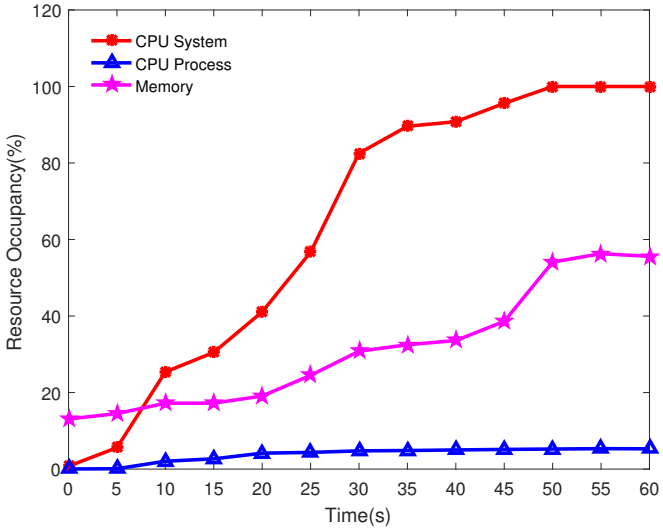

Figure 6. Graph of Changes in CPU, Memory, etc. During Network Attacks.

5.5. Experimental Comparison

When reconstructing the network attack scenario, the sample packets are first preprocessed, and the results returned from the preprocessing are some basic information for reconstructing the network topology and the attack parameters needed in the network attack. such as the packet type, the port information, the IP information of the attacker and the attacked, the length of the attacked packets, etc. Then the network topology is reconstructed and the network attack is implemented based on the preprocessed information. This reproduces the network attack scenario to achieve a high degree of matching with the sample.

To demonstrate the similarity between our proposed method and the original data, we will compare and analyze the five following aspects: IP address distribution and usage frequency, similarity calculated by Pearson algorithm, protocol proportions, packet length, and port binding services. By comparing these aspects, we can illustrate the degree of similarity between the original data and the regenerated data for attack scenarios. The specific comparisons are as follows:

(1) IP address distribution and usage frequency.

With the SDN-based network attack scenario recovery method proposed in this paper, the similarity of attack scenario recovery can be improved as much as possible. The model can fully represent the original data by adding probabilistic events, including the small sample of events present in the original data. In the raw data of this paper, each host has multiple different spoofed IP identities, and according to the statistics all attackers used a total of 28 IP addresses, but the number of packets sent by the attackers using different IPs varies. The IPs, 192.168.50.1, 192.168.50.6, 192.168.50.7, 172.217.11.2, and 172.217.9.226, are used with the highest frequency, and the remaining IPs are used with very low frequency. The statistics were also calculated in this paper, and the frequency graph of forged IPs is shown in Figure 7. The content in brackets after the IP address represents the number of times the IP appears. In the attack scenario recovery, to more clearly illustrate, the small sample events have been added. First, the main attack event IP is 172.16.0.5 for the statistics, accounting for 99.1818% of the sample and 99.2407% of the regenerated data. After that, the IP frequencies of the regenerated packets and the sample packets are compared, and the comparison graph is shown in Figure 8.

Figure 7. Frequency Diagram of Pseudo IP Addresses.

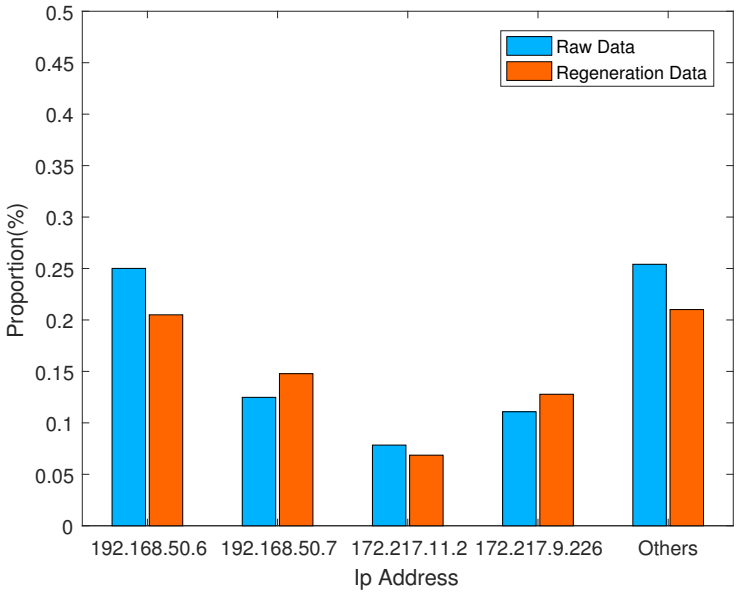

Figure 8. Probability Comparison Chart of Attack Packets and Sample Packets.

We can clearly see from the comparison graph in Figure 8 that there is a high degree of consistency between the original and regenerated data in terms of IP address usage frequency. Even for small sample events with very low IP usage, they can be accurately simulated due to the added probability model. Using the Pearson correlation coefficient calculation, the similarity can reach up to 99%.

(2) Similarity calculated by Pearson algorithm.

According to the network attack scenario recovery method proposed in this paper, the number of packets sent by the regenerated data and the original data in time are counted and the similarity of the attacks is compared. In this paper, we choose the time period from 0 to 5.5 s and count the total number of packets sent by the original data and the regenerated data every 0.5 s, respectively. From 0 to 2 s, the number of packets increases at a fast rate and there is a certain gap between the number of packets sent by the regenerated data and the original data. After that, as the number of sent packets slowly increases, the graph of the number of packets sent by the regenerated data floats around the graph of the number of packets sent by the original data, but the two rates are basically the same and the graphs of the changes of the two curves basically overlap together. The statistical results of the total number of packets sent during the time period from 0 to 5.5 s are shown in Figure 9.

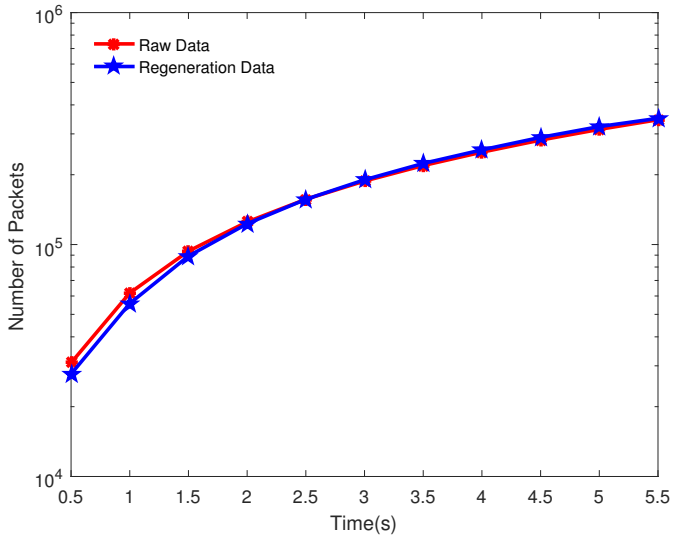

Figure 9. Comparison Chart of The Total Sum of Packets Sent on The Time Series.

According to the similarity calculation method mentioned in Section 4.1 using Pearson correlation coefficient: this paper has done three sets of experiments on three different Pcap packets, the first of which is the experimental result in Figure 9. The correlation coefficients were calculated for the original data and the regenerated data from the three sets of experimental results, where X is the original data, Y is the regenerated data, μ_X is the mean of X, μ_Y is the mean of Y, σ_X is the standard deviation of the original data X, σ_Y is the standard deviation of the regenerated data Y, and $\rho_{X,Y}$ is the correlation coefficient. In this paper, three different groups of experiments were performed. The statistics are shown in Table 3.

Table 3. Statistical Table of Correlation Coefficients.

Group	Group 1	Group 2	Group 3
μ_X	172,198.0833	182,011.3	141,554.8
μ_Y	173,701.0833	200,759.3	145,845.8
σ_X	113,199.1511	100,338.73414	78,006.45636
σ_Y	117,431.6822	117,964.93129	77,420.57464
$\rho_{X,Y}$	0.9998	0.9978	0.99

Based on the comparison described above, the Pearson correlation was calculated for the regenerated and original data for all three experiments. The three experiments showed high consistency and achieved a very high degree of similarity. As can be seen from Table 3, the method is stable, and the regenerated data traffic is essentially the same as the original traffic in terms of sending time attributes.

(3) Protocol proportions.

To demonstrate the similarity in the recovery of the network attack scenarios, the experiments were also compared statistically in terms of packet protocols, packet lengths, port binding services for sending packets, and port binding services for receiving packets. In this experiment, the protocols in the sample are mainly UDP protocols, and a few are ICMP protocols and TCP protocols, along with some other protocols, but the other protocols are almost a very small part. The results of the statistical comparison are shown in Table 4. By comparing the difference between both is not more than 0.2%. Based on the protocol distribution in Table 4 and comparison using Pearson correlation, the similarity of protocols can reach up to 98%.

Table 4. Network Packet Protocol Proportion Information.

Protocol	Raw Data	Regeneration Data
UDP	99.2815%	99.0780%
TCP	0.7028%	0.8908%
ICMP	0.0155%	0.0269%
Others	0.0002%	0.0043%

(4) Packet length.

In the sample data protocols are mainly UDP protocols, a small number of TCP protocols and ICMP protocols, and some other protocols because different protocols send packets of different lengths. The length of packets used to send in the UDP protocol is mainly 524, and there are a few other lengths of UDP protocol packets. The TCP and ICMP protocols themselves account for a smaller proportion, and the packet length distribution is more dispersed, so the proportion of the total number of packets is even smaller. The results of comparing the regenerated data with the native data are shown in Table 5. Using the Pearson correlation to compare the distribution of packet lengths in Table 5, the similarity can reach up to 98%.

Table 5. Packet Length Proportion Information Table.

Packet Length	Raw Data	Regeneration Data
$UDP.length == 524$	99.1404%	99.0768%
$UDP.length! = 524$	0.1411%	0.0012%
$TCP.length > 45$	0.3423%	0.5344%
$TCP.length \leq 45$	0.3605%	0.3564%
Others	0.0157%	0.0312%

(5) Port-binding services.

Due to the presence of several different protocols in this sample packet, there are different services bound to the same port. For example, on port 1483 both packets of the TCP protocol and packets of the UDP protocol are sent. As the source and destination ports are spread out, the number of different services bound to the port is counted in the statistics. The statistics show that the number of different services bound to different ports is 51,350 for the original data and 52,245 for the regenerated data. In conclusion, the difference between the two is negligible.

Through the comparison from five different perspectives above, it can be concluded that the regenerated network traffic for attack scenarios shows high similarity with the original data traffic.

According to the content mentioned in Section 4.4, the experiment can not only reproduce the UDP flood attack but also control the strength of the attack on the network. For network defenders, it would be significant to be able to identify other problems in the network by modulating the intensity of the attack on top of reproducing it. Therefore, based on this experiment, this paper regenerates traffic that increases and decreases the intensity of the network attack under the same network topology. The data generated by increasing and decreasing intensity were also compared with the sample data, and the results of the comparison are shown in Figure 10.

At the same time, network topology nodes can be added to the experiment, and the addition of network topology nodes allows the experimenter to perform some other types of attacks or experiments on these nodes. In this paper, we address the addition of network topology nodes and still perform attacks on the attacking hosts in the experiment without changing the original attack parameters. The initial state of the experiment is six hosts, and the number of packets is observed from 0 to 5.5 s by gradually increasing the number of hosts. The number of packets sent by the attacker in the experiment increases with the

number of hosts. The number of packets sent by the attacker varies with the number of hosts versus time, as shown in Figure 11.

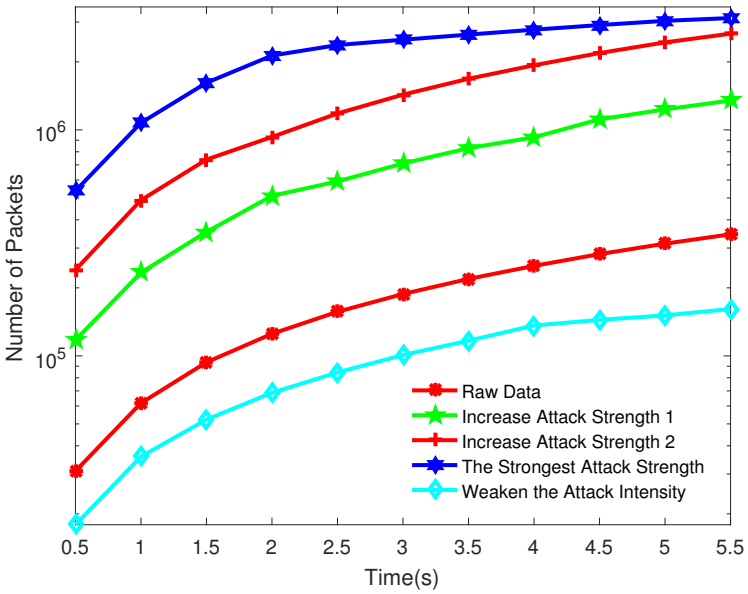

Figure 10. Frequency Diagram of Pseudo-IP Addresses.

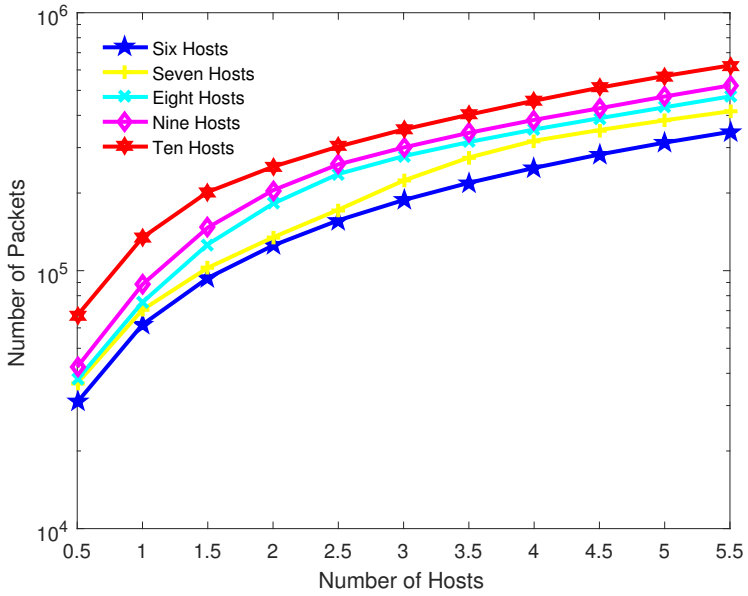

Figure 11. Packet change graph after increasing the number of hosts.

The network attack scenario recovery method proposed in this paper not only has a high reproduction rate and scalability, but also the attack scenario recovery is particularly fast. According to the relevant calculation methods mentioned in Section 4.3, The total time T spent in network attack scenario recovery is mainly concentrated in three aspects: data preprocessing T_1, network topology creation T_2, and attack command generation and attack execution T_3. For different numbers of datasets, time statistics are performed separately from these three aspects. As the number of packets increases, the time increases but is still fast. The statistical results are shown in Table 6.

Table 6. Timetable Required for Network Attack Recovery.

Time (s)	500	1000	5000	10,000	100,000
T_1	3.57	8.86	19.68	27.83	51.96
T_2	1.46	1.61	1.55	1.50	1.53
T_3	2.23	2.44	2.68	2.98	3.23
T	7.26	12.91	23.91	32.31	56.72

There exist many network topology reconstruction models with network attack scenario reconstruction models. However, with the proposed QoS requirements in the network, the simple, fixed network topology model is not a good simulation of the real network, and it is necessary to add connection properties to the existing connections. For example, QoS parameters such as bandwidth, latency, cost, and packet loss rate are added to the network model to better simulate the real network. Currently, the network topology scenario reconstruction mainstream is expressed in the form of language and diagram and does not really create the topology. The network topology model proposed in this paper not only adds QoS parameters but also creates the real network topology.

In the process of network attack recovery, the proposed network attack scenario recovery method in this paper is qualitatively evaluated against existing network attack scenario recovery methods. Four main aspects are compared: whether the network topology is created in the network attack scenario recovery; whether the real network traffic is regenerated; whether the intensity of the attack is controllable during the network attack; and the recovery emphasis of the network attack scenario. The comparison results are shown in Table 7.

Table 7. Qualitative Analysis Table.

Methods	Create Topology	Regenerate Real Traffic	Control the Intensity of the Attack	Restore Capability
H. Liu [7]	Yes	No	No	Strong
Y. Djemaiel [13]	No	No	No	Medium
Y. Zhang [14]	No	No	No	Strong
H. Wu [10]	Yes	No	No	Strong
W. Wang [15]	No	No	No	Weak
T. Guo [16]	No	No	No	Weak
Our method	Yes	Yes	Yes	Strong

5.6. Extra Costs and Limitations Discussion

In this section, we will mainly discuss the possible extra costs and limitations that may be involved in our research plan. We will analyze the potential limitations of our study and the associated extra expenses and propose corresponding solutions to these issues.

5.6.1. Extra Costs

In this experiment, we utilized a publicly available dataset that had already been labeled for network attack scene recovery, so there were no significant additional costs incurred. However, in real-world network attack scenarios, the time span for packet collection can be quite long. If we obtain a raw dataset, additional costs will be incurred. We would need to spend considerable time or technical resources to calibrate the attack process and extract the proper data. The accuracy of our network attack scene recovery is dependent on the accuracy of data calibration. Only with more accurate calibration data can we recover scenes closer to reality.

5.6.2. Limitations

In this work, although the SDN-based network attack scene restoration method proposed in this paper has achieved some results, due to the complexity and diversity of

network attacks, if data packets containing unknown network protocols are mixed in during scene reconstruction, these unknown network protocol packets still cannot be revived.

6. Conclusions

This paper proposes an SDN-based network attack scene recovery method. This method is able to regenerate the original data and integrate other network attacks into the reconstructed scene, generating new blended data. The first step of this method is to parse Pcap data of network attacks obtained from public sources. The second step is to generate the network topology by utilizing the network attack scenario topology reconfiguration model and generate corresponding attack scripts using the network attack scenario recovery probability model and attack sequence algorithm and finally, to regenerate traffic on the network topology. The experimental results show that the proposed method is closer to the actual network attack scenario, with a higher similarity of the attack scenario. Additionally, the proposed method not only enables the recovery of network attack scenarios but can also be extended to other experiments. For example, based on the restored network attack scenarios, the network topology nodes can be modified and the strength of the attack can be intensified or weakened and combined with other types of attacks.

Regarding the limitations mentioned in Section 5.6.2, for future work, we aim to investigate how to construct unknown protocol data packets and revive them, in hopes of automating adaptation and resurrection of various types of data packets. Additionally, we will enhance the accuracy and reliability of our model by introducing more data sources, optimizing algorithm design, and exploring new data analysis and modeling methods, among other approaches.

Author Contributions: Conceptualization, Y.W. and T.Z.; methodology, Y.W. and T.Z.; software, Y.W. and J.D.; validation, Y.W., J.D. and T.Z.; formal analysis, Y.W. and J.D.; investigation, Y.W. and J.D.; writing—original draft preparation, Y.W., J.D., Y.X. and X.H.; writing—review and editing, Y.W. and J.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research work is supported by the National Natural Science Founds of China (62072368, U20B2050) and the Key Research and Development Program of Shaanxi Province (2021ZDLGY05-09, 2022GY-040).

Data Availability Statement: Sample data sets used in the experiment site: https://www.unb.ca/cic/datasets/index.html; accessed on 2 March 2023, this study generated data sets, please contact dingjunxia@stu.xaut.edu.cn.

Acknowledgments: The authors would like to thank the editors and the reviewers for their valuable suggestions and comments.

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

References

1. Shen, Z.-Y.; Su, M.-W.; Cai, Y.-Z.; Tasi, M.-H. Mitigating SYN Flooding and UDP Flooding in P4-based SDN. In Proceedings of the 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS), Online Event, 8–10 September 2021; pp. 374–377.
2. Mladenov, B. Studying the DDoS Attack Effect over SDN Controller Southbound Channel. In Proceedings of the 2019 X National Conference with International Participation (ELECTRONICA), Sofia, Bulgaria, 16–17 May 2019; pp. 1–4.
3. Runze, C.; Fangming, R.; Yidan, L.; Lan, Y.; Yanli, C. A Simple DDoS Defense Method Based SDN. In Proceedings of the 2021 IEEE 15th International Conference on Anti-Counterfeiting, Security, and Identification (ASID), Xiamen, China, 29–31 October 2021; pp. 88–92.
4. Csikor, L.; Szalay, M.; Rétvári, G.; Pongrácz, G.; Pezaros, D.P.; Toka, L. Transition to SDN is HARMLESS: Hybrid Architecture for Migrating Legacy Ethernet Switches to SDN. *IEEE/ACM Trans. Netw.* **2020**, *28*, 275–288. [CrossRef]
5. Gedia, D.; Perigo, L. Performance Evaluation of SDN-VNF in Virtual Machine and Container. In Proceedings of the 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 27–29 November 2018; pp. 1–7.
6. Amin, R.; Reisslein, M.; Shah, N. Hybrid SDN Networks: A Survey of Existing Approaches. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3259–3306. [CrossRef]

7. Liu, H.; An, L.; Ren, J.; Wang, B. An Interactive Traffic Replay Method in a Scaled-Down Environment. *IEEE Access* **2019**, *7*, 149373–149386. [CrossRef]
8. Li, Y.; Miao, R.; Alizadeh, M. DETER: Deterministic TCP replay for performance diagnosis. In Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA, USA, 26–28 February 2019; pp. 437–452.
9. Li, L.; Hao, Z.; Zhang, Y.; Liu, Y.; Li, D. Modeling for Traffic Replay in Virtual Network. In Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications, IEEE 16th International Conference on Smart City, IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Exeter, UK, 28–30 June 2018; pp. 495–502.
10. Wu, H.; Liu, H.; Wang, B.; Xin, G. Accurate traffic replay based on interactive sequence and timestamp. In Proceedings of the 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), Guangzhou, China, 6–8 May 2017; pp. 1107–1110.
11. Mao, B.; Liu, J.; Lai, Y.; Sun, M. MIF: A multi-step attack scenario reconstruction and attack chains extraction method based on multi-information fusion. *Comput. Netw.* **2021**, *198*, 108340. [CrossRef]
12. Wei, Y.; Wu, F. Research on Network Topology Model of Tactical Communication System. In Proceedings of the 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Chongqing, China, 11–13 December 2020; pp. 808–811.
13. Djemaiel, Y.; Fessi, B.A.; Boudriga, N. Using Temporal Conceptual Graphs and Neural Networks for Big Data-Based Attack Scenarios Reconstruction. In Proceedings of the 2019 IEEE International Conferance on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Xiamen, China, 16–18 December 2019; pp. 991–998.
14. Zhang, Y.; Zhao, S.; Zhang, J. RTMA: Real Time Mining Algorithm for Multi-Step Attack Scenarios Reconstruction. In Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications, IEEE 17th International Conference on Smart City, IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China, 10–12 August 2019; pp. 2103–2110.
15. Wang, W.; Du, X.; Ren, Z.; Shan, D. Reconstructing attack scenarios based on causal knowledge and spatio-temporal correlation for cloud platforms. *Comput. Sci.* **2021**, *48*, 317–323.
16. Guo, T. *Research on Attack Scene Reconstruction Algorithm Based on Correlation Analysis*; Beijing University of Posts and Telecommunications: Beijing, China, 2018.
17. Huang, Y.; Sun, Y.; Lin, K.; Xie, B.; Fan, J.; Ma, Y. An Effective Reconstruction Method of the APT Attack Based on Hidden Markov Model. *J. Circuits Syst. Comput.* **2021**, *31*, 2250108. [CrossRef]
18. Rusek, K.; Suárez-Varela, J.; Almasan, P.; Barlet-Ros, P.; Cabellos-Aparicio, A. RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 2260–2270. [CrossRef]
19. Hajizadeh, M.; Phan, T.V.; Bauschert, T. Probability Analysis of Successful Cyber Attacks in SDN-based Networks. In Proceedings of the 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 27–29 November 2018; pp. 1–6.
20. Sun, W.; Li, Y.; Guan, S. An Improved Method of DDoS Attack Detection for Controller of SDN. In Proceedings of the 2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET), Beijing, China, 16–18 August 2019; pp. 249–253.
21. Naing, M.T.; Khaing, T.T.; Maw, A.H. Evaluation of TCP and UDP Traffic over Software-Defined Networking. In Proceedings of the 2019 International Conference on Advanced Information Technologies (ICAIT), Yangon, Myanmar, 6–7 November 2019; pp. 7–12.
22. Pérez-Díaz, J.A.; Valdovinos, I.A.; Choo, K.-K.R.; Zhu, D. A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning. *IEEE Access* **2020**, *8*, 155859–155872. [CrossRef]
23. Gill, S.; Lee, B.; Qiao, Y. Containerchain: A Blockchain System Emulator based on Mininet and Containers. In Proceedings of the 2021 IEEE International Conference on Blockchain (Blockchain), Melbourne, Australia, 6–8 December 2021; pp. 1–7.
24. Zulu, L.L.; Ogudo, K.A.; Umenne, P.O. Simulating Software Defined Networking Using Mininet to Optimize Host Communication in a Realistic Programmable Network. In Proceedings of the 2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD), Durban, South Africa, 6–8 August 2018; pp. 1–6.
25. Lee, S.; Ali, J.; Roh, B.-H. Performance Comparison of Software Defined Networking Simulators for Tactical Network: Mininet vs. OPNET. In Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 18–21 February 2019; pp. 197–202.
26. Tivig, P.T.; Borcoci, E.; Brumaru, A.; Ciobanu, A.-I.-E. Layer 3 Forwarder Application - Implementation Experiments Based on Ryu SDN Controller. In Proceedings of the 2021 International Symposium on Networks, Computers and Communications (ISNCC), Dubai, United Arab Emirates, 1–3 June 2021; pp. 1–6.
27. Chouhan, R.K.; Atulkar, M.; Nagwani, N.K. Performance Comparison of Ryu and Floodlight Controllers in Different SDN Topologies. In Proceedings of the 2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE), Bangalore, India, 19–20 March 2019; pp. 188–191.

28. Elsayed, M.S.; Le-Khac, N.-A.; Dev, S.; Jurcut, A.D. DDoSNet: A Deep-Learning Model for Detecting Network Attacks. In Proceedings of the 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Cork, Ireland, 31 August–3 September 2020; pp. 391–396.
29. Nguyen, M.H.; Lai, Y.-K.; Chang, K.-P. An Entropy-based DDoS attack Detection and Classification with Hierarchical Temporal Memory. In Proceedings of the 2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Tokyo, Japan, 14–17 December 2021; pp. 1942–1948.