mathematics

MDPI

Article

An Intrusion Detection System Based on Genetic Algorithm for Software-Defined Networks †

Xuejian Zhao [1,2], Huiying Su [1] and Zhixin Sun [1,2,*]

[1] Technology and Application Engineering Center of Postal Big Data, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
[2] Key Lab of Broadband Wireless Communication and Sensor Network Technology of Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
* Correspondence: sunzx@njupt.edu.cn
† This paper is an extended version of our paper published in 2020 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, China, 18–20 December 2020; pp. 309–313.

Abstract: A SDN (Software-Defined Network) separates the control layer from the data layer to realize centralized network control and improve the scalability and the programmability. SDN also faces a series of security threats. An intrusion detection system (IDS) is an effective means of protecting communication networks against traffic attacks. In this paper, a novel IDS model for SDN is proposed to collect and analyze the traffic which is generally at the control plane. Moreover, network congestion will occur when the amount of data transferred reaches the data processing capacity of the IDS. The suggested IDS model addresses this problem with a probability-based traffic sampling method in which the genetic algorithm (GA) is used to approach the sampling probability of each sampling point. According to the simulation results, the suggested IDS model based on GA is capable of enhancing the detection efficiency in SDNs.

Keywords: IDS; SDN; traffic sampling; genetic algorithm

MSC: 68M10

Citation: Zhao, X.; Su, H.; Sun, Z. An Intrusion Detection System Based on Genetic Algorithm for Software-Defined Networks. *Mathematics* **2022**, *10*, 3941. https://doi.org/10.3390/math10213941

Academic Editor: Catalin Stoean

Received: 22 September 2022
Accepted: 17 October 2022
Published: 24 October 2022

1. Introduction

The network architecture of the SDN is programmable compared with traditional networks. It separates the controls into a logic control plane [1,2]. Nevertheless, for the SDN, it is still susceptible to attacks from network. One of the efficient countermeasures is the deployment of an intrusion detection system (IDS) for the SDN [1,3]. In this paper, an IDS which is based on GA is proposed to prevent SDN from malicious data traffic.

The GA is an evolutionary algorithm. It simulates the phenomena of replication, crossover and mutation in natural selection, and it is insensitive to local optima, which makes it possible to find global optima [4]. The GA has been proposed for solving different optimization problems and has obvious advantages on optimization problems according to the experimental results [5].

An IDS is a security scheme that tracks and examines network traffic in order to find intrusions [6]. However, it is incredibly challenging to detect malicious packets in a network owing to the rapid growth of data traffic and the continually expanding network scale. Consequently, it might be difficult to locate the detection points in an IDS [7]. IDS hardware resources, such as CPU processing power, memory access speed and storage capacity, are typically constrained [8]. To improve the detection efficiency, multiple IDSs must be deployed to check a vast number of data packets, especially in large-scale network systems. The IDS we suggest is deployed on the control plane of an SDN, and by mirroring fully-configured SDN controllers, it collects and analyzes traffic from the switch [1].

In order to identify suspicious packets accurately, the IDS module typically classifies the collected traffic using classification algorithms. The authors of [9] created a Deep Neural Network model for suspicious traffic detection in SDN. In order to identify network risks for the IDS, an enhanced behavior-based support vector machine and learning algorithm utilized in the security monitoring system (SMS) was proposed in [10]. However, most of the current research focuses on packet sampling and detection at a specific SDN point. In fact, the SDN is massive and has a huge amount of data flow. Anomalies can be quickly identified with IDS modules deployed on the SDN control plane, allowing for quick and decisive action to be taken to ensure network security.

Not all packets of the network traffic can be examined due to the limited storage capacity of the IDS. Therefore, some malicious packets might escape from detection [11]. In order to address this problem, some studies suggested an IDS model for capturing partial traffic in applications. Using center metrics in SDNs, Yoon et al. [12] suggested a scalable flow sampling method. This approach uses per-flow and per-switch sampling to probabilistically capture packets on the switch, and it calculates the traffic sampling sites of switches by applying center metrics in graph theory. However, sampling might cause some valuable information to be lost. The proposed scheme offers a sample rate adjustment strategy to resolve this problem and establish the best sampling rate for switching. Due to the IDS's limited detection capacity, the technique determines the best sampling rate for each switch. Therefore, despite the IDS's limited detecting capabilities, the suggested approach fully utilizes them. Results from simulations show that our suggested IDS model can accurately detect malicious traffic with limited processing capacity. Additionally, simulation results demonstrate the suggested method is superior compared to other similar strategies.

The contributions of this work are summarized as follows: (i) A new IDS model has been proposed which can detect suspicious packets on numerous switches in an SDN; (ii) A scheme based on genetic algorithm to approach the optimal sampling rate of each switch has been proposed; (iii) The effectiveness of the suggested IDS model and the efficiency of the optimal sampling rate scheme have been precisely verified by experiments.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of previous related work. In Section 3, we present the preliminaries and methodology of the solution. In Sections 4 and 5, the proposed IDS model and the scheme to compute the optimal sampling rate are presented in detail. In Section 6, the evaluation and analysis of the simulation results of the proposed IDS model are presented. In Section 7, we conclude this paper and discuss ideas for future work.

2. Related Works

Numerous research works of intrusion detection technology concentrate on how to effectively manage large numbers of traffic samples related to conventional IP networks [13–16]. Silva [13] showed the value of network traffic sampling in his research. Additionally, a mechanism for calculating each network device's weight according to the memory usage, CPU load and data volume has been proposed. An adaptive feature recognition sampling technique was put out by Bartos [14] as a solution to the issue that sampling network traffic can decrease the precision of subsequent anomaly detection. The simulation results showed that traffic sampling can improve the efficiency of anomaly detection and minimize information loss throughout the sample process. In order to increase the detection of network attacks and balance IDS loads, Ha et al. [15] suggested a clustering-based flow grouping method that distributes flows according to routing information and flow data rate. In contrast to the approach based on traditional clustering, Ahmed [16] proposed a network traffic summary technique which may further create statistics for data mining in high-dimensional complicated network traffic data sets.

SDN security technologies have been the subject of numerous research studies [17–21]. Only a small number of them, however, focused on the attack detection of SDN. The fusion of Statistical Fingerprint IDS and SDN architecture can prevent the growth of malicious traffic in the network [20]. In order to identify the patterns in the data coming through the

firewall, the authors of [21] employed association rules and constructed a packet filtering firewall on an SDN controller called Floodlight. The authors of [22] presented an overview of several kinds of intrusion detection systems and the new technology of SDN. To minimize the effects of an unbanlanced data flow, the authors of [23] construct an adaptive IDS model of SDN based on online ensemble learning algorithm. Some studies have concentrated on a specific probability of data flow sampling in light of the IDS's comparatively poor detection capacities.

To support system packet sampling, probabilistic packet sampling and other diverse sampling approaches in an SDN, an extended OpenFlow called FleXam was suggested in [24]. In [25], the authors suggested a low-latency, sampling-based network measurement platform called OpenSample, which implements quick traffic statistics measurement in SDN using the probabilistic packet sampling of the sFlow protocol. According to [11], an optimization problem was formulated to determine an appropriate sample rate for each switch because the processing capacity of an IDS server is considered significantly smaller than the entire quantity of traffic in a large-scale network system. Utilizing mirroring, the IDS server samples network traffic at the best sampling rate.

IDS servers are often installed at the edge of a network [1,12]. By mirroring the fully configured SDN controller, an IDS server can sample traffic from any switch and then process all of the switch packets. The strategy employed flow sampling techniques to cut down on the duplication of traffic and minimize network overhead. However, there will be an increase in network overhead and possibly even network congestion due to the transmission of sampled traffic and the transfer of the control information. Additionally, delays in feedback may occur due to the time needed for information transmission. In [26], a functional modular control plane architecture model was suggested. This architectural model increases the flexibility and scalability of a single centralized controller (e.g., NOX) by decoupling the control plane. In [27], Hu proposed a distributed architecture for the SDN control plane. To demonstrate the scalability of the control plane, the authors of [28] suggested adding some control features to the data plane. A hierarchical control plane architecture featuring peer-to-peer communication among logically scattered controllers was created by the developers of [29]. An Intrusion Detection and Prevention System (IDPS) was designed and put into use by the authors of [30] using SDN. The suggested IDPS is a software application that monitors malicious activity or security policy violations on networks and systems and then takes action to stop them. A comparison study of several IDS systems based on a deep learning model and machine learning methodologies is explored in the article, and future perspectives for SDN security are detailed in [31], which also provides an overview of the security solutions currently available for the SDN. To increase the overall accuracy of intrusion detection in SDNs, the authors of [32] proposed a five-level hybrid classification system combining the k-nearest neighbor approach (kNN), the extreme learning machine (ELM), and the hierarchical extreme learning machine (HELM). The authors of [33] noted that the visibility and flexibility of managed, centralized, and regulated software defined networks has increased. However, these advantages also result in a more vulnerable environment and some serious challenges. This study demonstrates the use of tree-based machine learning algorithms for traffic monitoring to detect malicious behaviors in the SDN controller.

In this paper, we propose to install an IDS module on an SDN's control plane. Our scheme utilizes the flexibility and programmable network management of SDN in comparison to previous alternatives. Furthermore, adequate processing capacity is offered by the scalable control plane of SDN.

3. Preliminaries and Methodology of the Solution

3.1. Preliminaries

Compared to traditional networks, SDN features a centralized control system which manages a tremendous quantity of data traffic. The application plane, control plane, and data plane are the three tiers of the SDN architecture [1,34]. Numerous applications

make up the application plane of the SDN. The control plane is in charge of centrally supervising the devices of the data plane and managing the network's overall information [35]. Using OpenFlow, the control plane keeps monitoring on the data plane [36]. Additionally, the data plane provides the function of data forwarding. The operational effectiveness of many Internet applications and service systems can be increased via SDNs. Compared to the conventional routing table, each switch's flow table on the data plane for data processing is more complicated, since the SDN controller would need to give routing instructions for all of the network traffic which is forwarded.

3.2. Methodology of the Solution

The network congestion will occur when the overall amount of sampled traffic exceeds its processing capacity. In order to address this problem, we proposed an IDS for SDN based on a genetic algorithm and a scheme that is based on the sum of false-negative rates to choose the optimal sampling rate of the IDS.

The studied network architecture is made up of OpenFlow-based switches and an SDN controller. Based on OpenFlow which is on the control plane, the SDN controller is linked to the switches. Through OpenFlow, the controller monitors traffic and manages each switch's flow forwarding table.

The control plane is abstracted into a *master module* and a number of *sub-function modules* without losing generality. Each sub-function module is primarily managed and controlled by the master module. According to their types, messages from the data plane are passed to the appropriate sub-function modules. The sub-function modules can be flexibly enlarged internally while being logically centralized, which further improves the control plane's adaptability, reliability and overall performance.

The control plane of an SDN is where the suggested IDS module is installed. All packets on all switches are sampled by the SDN controller, which then sends the samples to the IDS module for additional analysis. The IDS module will trigger and feed back to the master module a security alert if it discovers a suspicious packet. In order to secure the network, the master module reconfigures it based on the switches' present status and the findings of the detection.

GA is suitable for solving complex optimization problems. The IDS module can accurately detect malicious packets and immediately deliver the alert to the SDN controller when using GA to solve the optimal sampling probability of switches.

4. A GA-Based IDS Model

Some malicious packets may be mistakenly detected by improper traffic sampling. To reduce this proportion of false positives, the IDS model requires a sampling probability method. Therefore, we present the suggested GA-based IDS model based on the procedures proposed in [1] in this section.

4.1. Overview of the Proposed IDS Model

In order to prevent the congestion of network when the amount of data transferred reaches the data processing capacity of the IDS, we provide an algorithm for managing the sampling rates of switches which will minimize malicious packet loss rate while ensuring the overall number of flows sampled stays below the IDS module's maximum detectability.

Based on OpenFlow, we presume that there exist g flows and s switches. The packet transmission rate of the i_{th} flow is specified as o_i. The malicious sending rate of the i_{th} flow is given as γ_i $(0 \leq \gamma_i \leq o_i)$. o_i and γ_i are units of packets per second (pps). There is no malicious packets when $\gamma_i = 0$. All packets are harmful when $\gamma_i = o_i$. The following definition applies to the packet sending rate vector:

$$\vec{o} = [o_1, o_2, \ldots, o_g]. \quad (1)$$

The following is the defined vector of the malicious sending rate:

$$\vec{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_g]. \tag{2}$$

The routing information matrix $A(k \times s)$ is created by an SDN controller which continually monitors the traffic status of each underlying switch. Each element of the matrix is described as follows:

$$A(a_{ij}) = \begin{cases} 0, & i_{th} \text{ do not pass } j_{th} \text{ switch,} \\ 1, & i_{th} \text{ pass } j_{th} \text{ switch.} \end{cases} \tag{3}$$

According to the packet sending rate and the routing information matrix of each flow, the packet sending rate across the j_{th} switch is defined as b_j (pps). The following formula is used to calculate the packet sending rate vector of n switches $\vec{b} = [b_1, b_2, \ldots, b_n]$:

$$\vec{b} = A \times \vec{o}. \tag{4}$$

For the purposes of lessening the load pressure on the IDS module and the network overhead imposed by mirroring, the IDS model samples packets sent from switches and then transmits the samples to the IDS module on the control plane. The definition of the sample probability vector for switches is $\vec{x} = [x_1, x_2 \ldots x_j, \ldots]$, where x_j is the sampling probability of packets passing through the j_{th} switch. All packets passing through the j_{th} switch will be sampled and sent to the IDS module if $x_j = 1$. The sampling probability vector of the switches is described as follows:

$$\vec{x} = [x_1, x_2, \ldots, x_s], 0 \leq x_j \leq 1 \tag{5}$$

It is vital to select a sampling probability wisely in order to reduce the total false-negative rates. The optimal sampling probability vector, i.e., $\vec{x}^*$, is solved by the master module.

4.2. The Process For Anomaly Detecting of the GA-Based IDS

The process for anomaly detecting of the GA-based IDS can be summed up as follows:

1. The SDN controller gathers data on the state of switches placed on the data plane, computing data such as the number of flows g, switches n, packet sending rate vector $\vec{o}$, routing information matrix A, and initialized malicious sending rate of each flow η_i.
2. The SDN controller samples traffic from all switches and directs the sampled traffic to the IDS module by solving the optimal sampling probability vector $\vec{x}^*$ with GA in the master module.
3. The master module is alerted by the IDS module when flows contain malicious packets, enabling it to quickly fend off attacks. In the mean time, the IDS module updates the master module with its estimates of $\vec{\gamma}$.
4. The master module updates $\vec{x}^*$ using the new value of $\vec{\gamma}$; then, it transmits the traffic sampled to the IDS module for another time. Then, it continues by repeating the previous steps.

5. The Optimal Sampling Vector

5.1. Problem Formulation of the Optimal Sampling Vector

With the sampling technique, the accuracy of classifier and sampling can both impact the ability of the network to resist malicious attacks. For simplicity, a false-negative rate is used in the rest of the work to measure sample accuracy.

Define p_i as the false-negative rate of the i_{th} flow; then, the false-negative rate vectors of g flows are:

$$\vec{p} = [p_1, p_2, \ldots, p_g], 0 \leq p_i \leq 1 \tag{6}$$

If malicious packets of a flow have not been sampled from the sender to the receiver, IDS will not be able to detect them in this flow. This flow will be classified as normal flow by mistake, and the controller will not implement necessary measures to resist it. Let $p_{i,j}$ denote the probability that the j switch misses malicious packets while sampling the i_{th} flow and $\alpha(i)$ denote the set of switches which the i_{th} flow passes through. Therefore, p_i can also be defined as the product of $p_{i,j}$ of all switches in $\alpha(i)$:

$$p_i = \prod_{j \in \alpha(i)} p_{i,j} \tag{7}$$

It should be noted that Equation (7) refers to the probability that each intermediate switch cannot catch malicious packets. $min(p_i)$ is the best to achieve a higher attack detection efficiency. The calculation of $p_{i,j}$ will be described in the following.

Assuming $x_j \cdot b_j$ packets are sampled from b_j packets per second (both b_j and $x_j \cdot b_j$ are integers), then the false-negative rate $p_{i,j}$ can be calculated as follows:

$$p_{i,j} = \begin{cases} 0, & if\ b_j - \gamma_i < x_j \cdot b_j, \\ \dfrac{\binom{b_j - \gamma_i}{x_j \cdot b_j}}{\binom{b_j}{x_j \cdot b_j}} = \frac{(b_j-\gamma_i)!(b_j-x_j \cdot b_j)!}{b_j!(b_j-\gamma_i-x_j \cdot b_j)!}, & otherwise. \end{cases} \tag{8}$$

If the number of normal packets in the i_{th} flow is less than the number of sampled packets ($b_j - \gamma_i - x_j \cdot b_j < 0$), all packets sampled contain at least one malicious packet, so $p_{i,j}$=0. If the i_{th} flow does not pass through the j_{th} switch, the sampling of the jth switch does not affect the false-negative rate of the i_{th} flow, so $p_{i,j} = 1$.

For better illustration, Equation (8) is further relaxed by introducing the Gamma function [37]. The gamma function is defined by $\Gamma(t) = \int_0^\infty x^{t-1} \cdot e^{-x} dx$, which is related to factorial as:

$$\Gamma(t) = (n-1)!. \tag{9}$$

Then, Equation (8) is rewritten as:

$$p_{i,j} = \begin{cases} 0, & if\ b_j - \gamma_i < x_j \cdot b_j, \\ \frac{\Gamma(b_j-\gamma_i+1) \cdot \Gamma(b_j-x_j \cdot b_j+1)}{\Gamma(b_j+1) \cdot \Gamma(b_j-\gamma_i-x_j \cdot b_j+1)}, & otherwise. \end{cases} \tag{10}$$

The optimization problem to find the $\vec{x}^*$ is formulated as follows:

$$\mathcal{P}1: \quad \min_x \sum_i p_i \tag{11}$$

s.t.,

$$\sum_{j=1}^{s} x_j \cdot b_j \leq C, \tag{12}$$

$$0 \leq x_j \leq 1, \qquad j = 1, 2, \ldots, s. \tag{13}$$

The objective function of $\mathcal{P}1$ is the sum of the false-negative rates of all flows. Further, based on Equation (10), we can rewrite Equation (11) as follows:

$$\begin{aligned} \textstyle\sum_i p_i &= \textstyle\sum_i (\prod_j p_{ij}) \\ &= \textstyle\sum_i (\prod_j \frac{\Gamma(b_j-\gamma_i+1) \cdot \Gamma(b_j-x_j \cdot b_j+1)}{\Gamma(b_j+1) \cdot \Gamma(b_j-\gamma_i-x_j \cdot b_j+1)}). \end{aligned} \tag{14}$$

Solving $\mathcal{P}1$ will approach to the $\vec{x}^*$ that can minimize the global false-negative rate of an IDS. Due to the limited capacity of IDS, the total amount of traffic sampled cannot exceed the detection capability of IDS. The detection capability of IDS can be simply defined as the

maximum number of traffic that IDS can handle correctly without any significant decrease in detection performance. Let C denote the IDS capacity. Equation (12) is used to constrain the relationship between C and $\vec{x}$. Equation (13) is used to limit the sampling probability for each switch to a positive number between 0 and 1.

5.2. GA-Based Approach to the Optimal $\vec{x}^*$

In order to solve $\mathcal{P}1$, we proposed to apply GA for computational efficiency. In particular, the fitness function in the GA approach is the objective function shown in Equation (11). The proposed approach has three steps, Initial solution population, Selection operator, Crossover operator and mutation operator.

1. The Initial solution population is used to initialize k solutions in order to form a solution population. The variable of the model is the sampling probability vector $\vec{x}$. We choose the initial solution population as $\overrightarrow{x^m} = rand(0,1) \cdot one(0,1) \quad m = 0, 1, 2, \ldots, k$. Each solution in the initial solution population is an n-dimensional equivalent vector whose values are probability values generated randomly.
2. The Selection operator is used to guarantee the correctness of the solution and satisfy the constraints in Equations (12) and (13), accelerating the convergence to the optimal solution. The selection operator used has two parts:
   - Judge whether a solution satisfies the constraints of Equations (12) and (13). If it is not satisfied, delete the solution in the population and then set a solution to join the population randomly.
   - Each round is sorted according to the value of fitness function (10), keeping the best individuals without crossover and mutation, and going to the next generation directly.
3. The Crossover operator and mutation operator is used to cross and mutate the rest of the individuals selectively, except that the best individual in each generation goes to the next generation of execution algorithms directly. r is generated randomly. If $r \leq p_c$, the chromosome needs crossover, thus forming a collection of chromosomes to be crossed gradually. If the number of chromosomes in the collection is even, chromosomes cross each other sequentially. If it is odd, the last chromosome in the collection goes into the next generation directly without crossover after the other chromosomes cross each other sequentially. The specific way of crossover is then given by:

$$\begin{cases} (\vec{x^0})' & = l_1 \cdot \vec{x^0} + l_2 \cdot \vec{x^1} + l_3 \cdot \vec{x}^*, \\ (\vec{x^1})' & = l_2 \cdot \vec{x^0} + l_1 \cdot \vec{x^1} + l_3 \cdot \vec{x}^*, \end{cases} \tag{15}$$

   where $l_1 + l_2 = 0.5$, $l_3 = 0.5$. The genes inherited from parents and the global optimal genes are half each. The mutation operator and crossover operator are similar. The best chromosome in each generation does not mutate, and the rest of the chromosomes produce a random number r. If $r \leq p_m$, choose it to mutate. The specific way of mutation is then given by:

$$(\vec{x})' = \vec{x} + \Theta, \tag{16}$$

   where $\Theta \sim N(0,1)$. Each mutation is generated randomly. After iterative iterations until the optimal solution no longer changes, the result obtained is the optimal sampling probability.

In order to find the optimal solution $\vec{x}^*$, a parameter γ_i must be computed first. When a flow begins to transmit data, we can only know how many packets the flow sends per second without knowing how many malicious packets it sends per second. Malicious packets are sent at different rates for different flows. The value of parameter γ_i will affect the choice of sampling probability, so we update γ_i continuously by taking feedback in the SDN controller so that γ_i will be corrected according to the feedback value until it approaches the exact value.

Assume $\gamma_i = \eta_i$ when the scheme is initialized. The SDN controller samples the i_{th} flow on each switch and directs the packets sampled to the IDS module. The IDS module detects that the flow sends r_i malicious packets per second, while the total number of packets sampled per second for this flow is y_i:

$$y_i = \sum_{j \in \alpha(i)} s_i \cdot x_j. \tag{17}$$

The IDS module feeds back the estimate γ_i of $\hat{\gamma}_i$ to the master module for the i_{th} flow:

$$\hat{\gamma}_i = \frac{r_i}{y_i} \cdot s_i. \tag{18}$$

However, we cannot replace the initial value with the estimated value completely, so the weight $\theta (0 \leq \theta \leq 1)$ is set. The updated γ_i can be calculated as follows:

$$\gamma_i = (1 - 0) \cdot \gamma_i + \theta \cdot \hat{\gamma}_i. \tag{19}$$

Ideally, γ_i will approach the optimal value after a few iterations. That concludes the approach.

The GA-based approach to the optimal $\vec{x}^*$ is summarized in Algorithm 1. In Algorithm 1, $FuncBest(\cdot)$ finds the optimal solution $\vec{p}_{best}$ and the value of fitness function M_{best} with Equation (11). $Judge(\cdot)$ is used to judge whether GA is converged. $Select(\cdot)$ picks the suitable solutions from the last previous generation. $Cross(Select(\cdot))$ is the crossover calculation function of GA and $Mu(\cdot)$ is the mutation calculation function of GA.

Algorithm 1 Approach to the optimal $\vec{x}^*$.

Input: $s, g, A, \vec{o}, k, p_c, p_m, \Theta, \vec{\gamma}$.

Output: $\vec{x}^*$.

```
Set the initial solution population P of GA
(p⃗_best, M_best) ← FuncBest(P)
M = M_best
while Judge(M) do
    P ← Cross(Select(P))
    P ← Mu(P)
    (p⃗_best, M_best) ← FuncBest(P)
    if M_best ≤ M then
        M ← M_best
    end if
end while
x⃗* ← p⃗_best
```

5.3. The GA-Based IDS Scheme

The proposed GA-based IDS scheme is summarized in Algorithm 2. The GA-based IDS solves the problem of overloading in the IDS module effectively. Using GA to solve the optimal sampling probability of switches, the IDS module can detect malicious packets accurately and send the warning to the SDN controller instantly without receiving all packets transmitted by switches. Meanwhile, the high-quality feedback mechanism of the

master module uses the feedback information of the IDS module to update $\vec{\gamma}$ continuously, which makes the sampling scheme more efficient and accurate.

Algorithm 2 The anomaly detection process of the GA-based IDS.

Input: $s, g, A, \vec{o}, \theta, \vec{\gamma}, \vec{\eta}$.

Output: Suspicious flows.

```
Initialize the γ⃗ with η⃗
while True do
    compute x⃗* and p⃗ with Algorithm 1
    if IDS needs to be stopped then
        break
    end if
    Sampling ← x⃗*
    classify the data according to the K-means clustering algorithm
    identify suspicious flows
    γ⃗ ← Equation (19)
end while
```

6. Experiment and Discussion

In this section, the suggested schemes are illustrated through simulations and tests. First, we demonstrate that using GA to optimize the IDS model can produce results that are more accurate and efficient than those produced by other algorithms. Then, we contrast the suggested IDS sampling plan with those already in place in SDN setups. The simulations and assessments are performed by Matlab 2014b on a Thinkpad E460 with an Intel I7@3.5GHz and 16GB RAM.

6.1. Network Settings

The networks used in the simulations and experiments are constructed randomly. To assess the suggested techniques, the intrusion flows' malicious sending rates $\vec{\gamma}$ are modified. In particular, detailed settings of the two tested networks (i.e., network A and network B) are listed in Table 1.

Table 1. The parameters of network settings.

Parameters	Network A	Network B
# Switches	30	100
# Links	200	600
Data rates	12–256 Mbps	12–256 Mbps
Malicious flows	3	3

6.2. Approach to the Optimal Sampling Probability $\vec{x}^*$

The proposed Algorithm 1 is used to find the optimal sampling probability $\vec{x}^*$. In this section, we first test and find the optimal settings for running GA in Algorithm 1. Then, we compare the proposed Algorithm 1 with other widely-adopted approaches, i.e., Artificial Neural Network (ANN) and Particle Swarm Optimization (PSO) for both network A and network B.

There are two parameters to be set for the GA to approach the optimal $\vec{x}^*$: crossover probability p_c and mutation probability p_m. In order to find the best parameter settings so

that the optimal sampling probability can be found quickly, we test several pairs of p_c and p_m randomly to solve $\vec{\gamma}$ and compare the result with the true value. Table 2 shows some of the parameter pairs that return the best results for network A and network B, respectively. As we can see, the parameter pair that returns the best results for both network A and B is $p_c = 0.4$, $p_c = 0.6$. Without loss of generality, $p_c = 0.4$, $p_c = 0.6$ are used to run Algorithm 1 for further analysis in this work.

Table 2. The false-negative rates with different p_c and p_m.

p_c	p_m	Network A	Network B
0.3	0.7	0.60	0.70
0.4	0.6	0.27	0.32
0.5	0.5	0.37	0.46
0.6	0.4	0.43	0.44
0.8	0.2	0.50	0.55

Next, we demonstrate the proposed Algorithm 1 by comparing it with ANN-based and PSO-based approaches. In particular, the malicious sending rates of three attack flows are set to 12 pps, 18 pps and 25 pps in each round of comparison. As shown in Figure 1, the x-axis indicates the number of iterations, and the y-axis indicates the mean of false-negative rate of three attack flows. With both tested networks, all three approaches are able to converge with enough iterations. In comparison, our proposed Algorithm 1 is able to achieve a much lower capture-failure rate among the three approaches in both network settings. In particular, our proposed GA-based Algorithm 1 converges faster than the PSO-based approach. Although Algorithm 1 converges a bit slower than the ANN-based approach, it is just a few iterations behind, with a much better result. Now that we have verified that our proposed Algorithm 1 is efficient in finding $\vec{x}^*$, we will then evaluate the proposed IDS model.

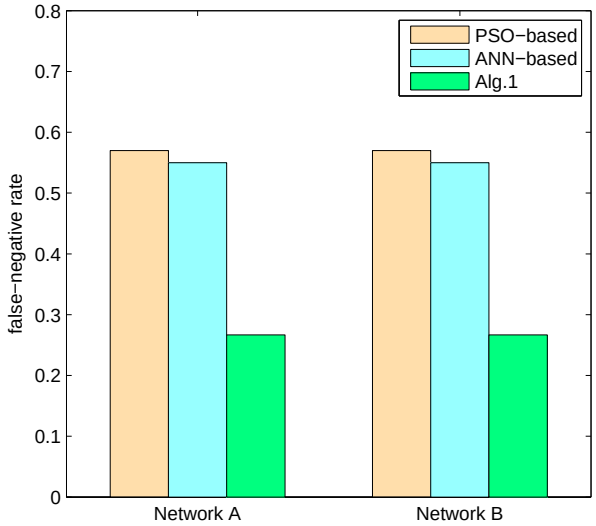

Figure 1. Comparison of Algorithm 1.

6.3. Evaluation of the Sampling Scheme

In this subsection, we evaluate the proposed sampling scheme, i.e., Algorithm 2, by comparing it with two existing schemes. In particular, we compared it with the probabilistic sampling scheme (noted as Alg. P) and the optimal sampling scheme (noted as Alg. S) proposed in [11] with the same simulation settings. Without loss of generality, we use the same network settings as the previous evaluations. Assume that three flows (out of 200 flows) contain malicious packets with a ratio of 2%.

Figure 2a,b show the mean of false-negative rates with respect to the IDS capacities achieved by the three schemes. In particular, Figure 2a,b are the results for network A and network B, respectively. We can see that our proposed Algorithm 2 outperforms the other two schemes in both network settings. Algorithm 2 has the fastest convergence rate to achieve the lowest capture-failure rate.

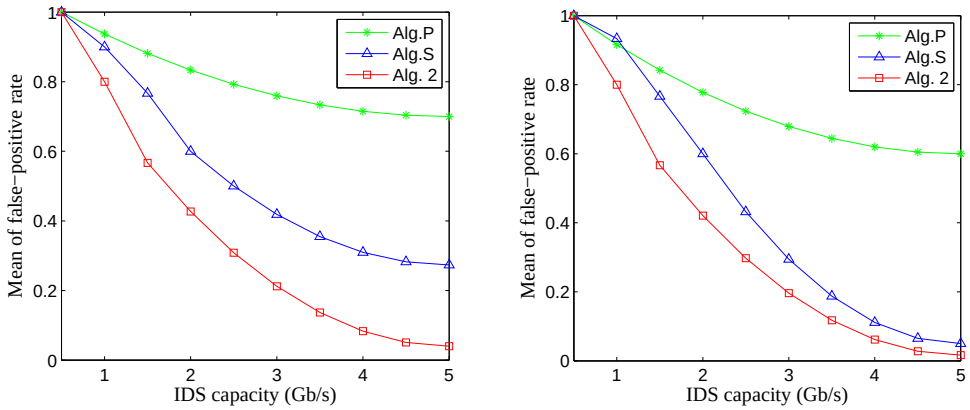

Figure 2. Mean false-negative rate with respect to IDS capacity: (**a**) shows the mean false-negative rates for network A, and (**b**) shows the mean false-negative rates for network B.

Next, we show the mean false-negative rate with respect to the ratio of malicious packets. As shown in Figure 3a,b, the proposed Algorithm 2 has the fastest convergence rate among the three approaches. In practice, the proposed Algorithm 2 may require 20% or fewer iterations than Alg. S, while Alg. P with a much slower convergence is left out of the competition.

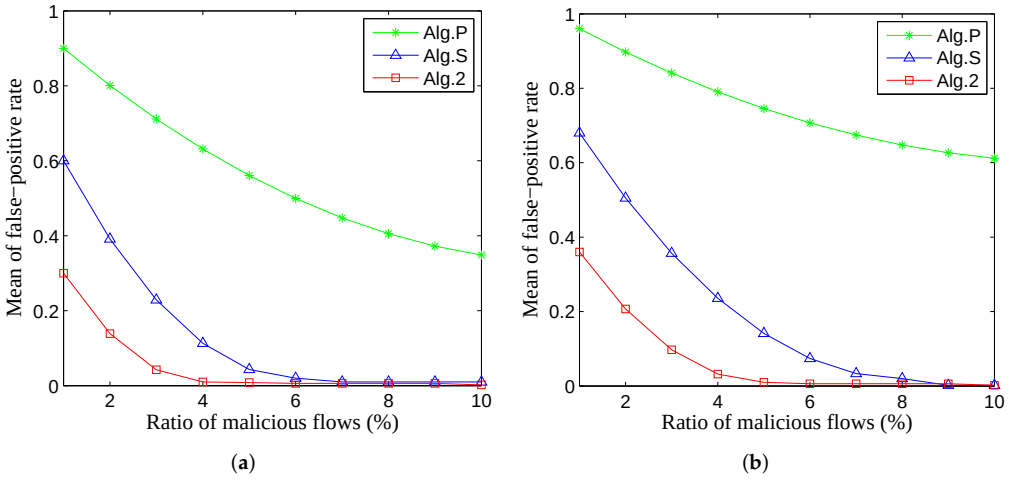

Figure 3. Mean false-negative rate with respect to the ratio of malicious flows: (**a**) shows the mean false-negative rate w.r.t the ratio of malicious flows for network A, and (**b**) shows the mean false-negative rate w.r.t the ratio of malicious flows for network B.

6.4. Evaluation of the IDS Model

In this part, we assess the suggested IDS model using the suggested sample method (i.e., Algorithm 2), and the ideal parameters (i.e., $\vec{x}^*$) calculated from the prior assessment. The initial value of the malicious sending rate of each flow η_i is set to 2% of o_i. The malicious sending rate for the three attack flows are each initialized as 12 pps, 18 pps and 25 pps. In order to provide a clearer demonstration, the malicious rate are updated to 40 pps, 56 pps, and 60 pps at the experiment's 50th iteration.

As shown in Figure 4a–d, in each iteration, the false-negative rates for three flows are calculated, and $\vec{\gamma}$ is updated in accordance with Equation (19). As seen in Figure 4a,b, after several updates to $\vec{\gamma}$, the proposed IDS model can detect the majority of malicious packets in network A. As the model iterates, the accuracy rises. Additionally, in the initial few iterations, the convergence is faster with a higher θ. The same results apply to network B, as shown in Figure 4c,d. The accuracy of anomalous traffic detection will therefore be improved by updating $\vec{\gamma}$ in the traffic sampling process. If the malicious packet rate is high enough with an optimized $\vec{\gamma}$, the missing rate might be nearly 0. Additionally, the findings from Figure 4a–d show that the convergence rate declines with a smaller θ. Additionally, the fact that γ_i approaches the exact value in continuous feedback is what causes the convergence.

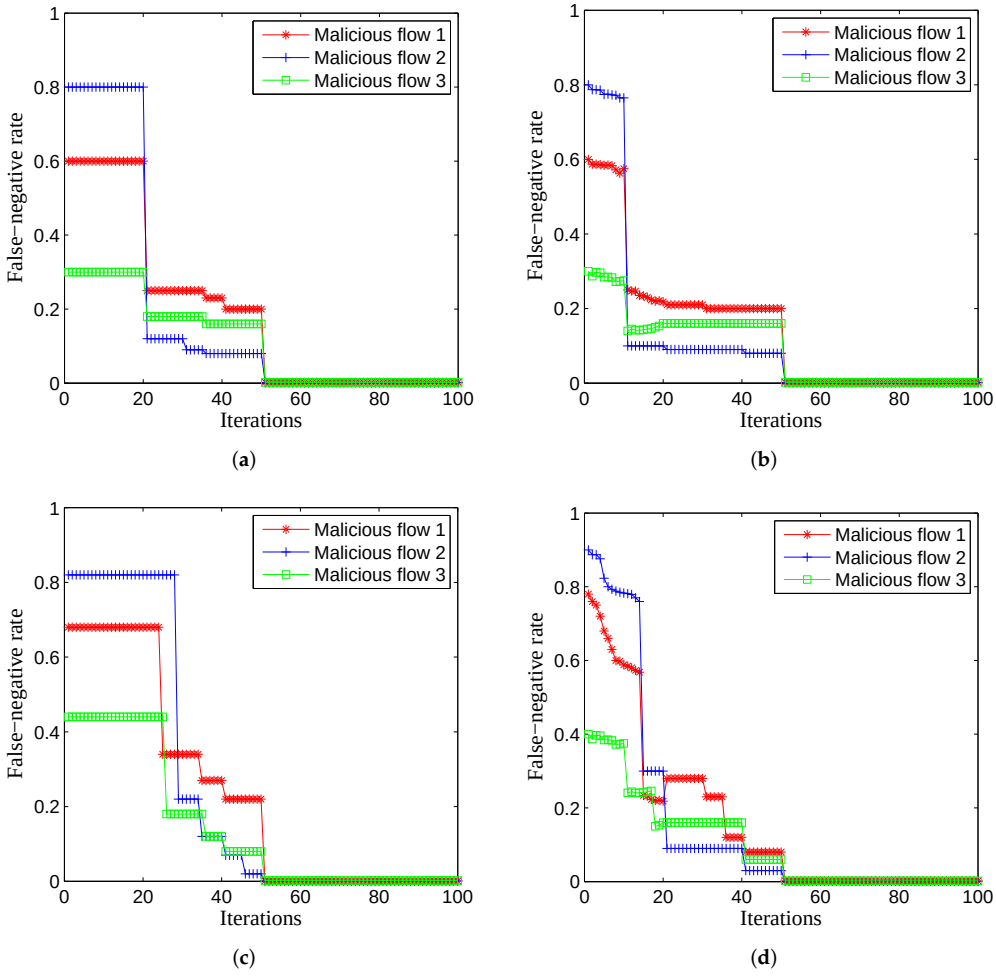

Figure 4. Comparison of false-negative rate with respect to different θ: (**a**) shows the false-negative rate for network A ($\theta = 0.2$), (**b**) shows the false-negative rate for network A ($\theta = 0.8$), (**c**) shows the false-negative rate for network B ($\theta = 0.2$), and (**d**) shows the false-negative rate for network B ($\theta = 0.8$).

The change of γ_i is further demonstrated in Figure 5a–d. As it demonstrates, γ_i soon approaches the actual value. After the 50th iteration, we update the actual value of γ_i to show that our suggested IDS model can adjust to rapid network changes.

Based on the false-negative rates, the experimental results show that compared to other widely adopted methods, the proposed IDS model and the the optimal sampling rate scheme are capable of lowering the overhead on the IDS module and enhancing the detection efficiency in SDNs under medium network loads.

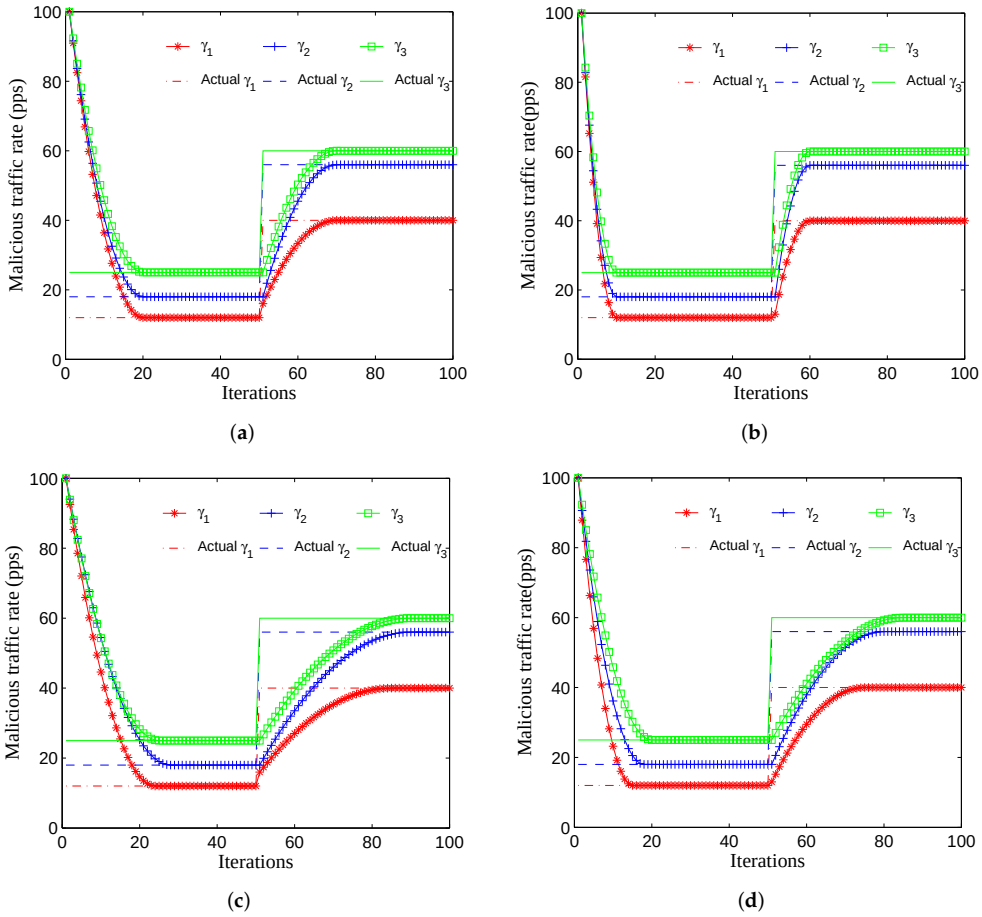

Figure 5. Comparison of rate $\vec{\gamma}$ with respect to different θ: (**a**) shows the change of γ_i for network A ($\theta = 0.2$), (**b**) shows the change of γ_i for network A ($\theta = 0.8$), (**c**) shows the the change of γ_i for network B ($\theta = 0.2$), and (**d**) shows the the change of γ_i for network B ($\theta = 0.8$).

7. Conclusions

In this paper, we suggested an IDS for SDN based on a genetic algorithm to detect suspicious packets. The suggested IDS is specifically installed on an SDN's control plane. Compared to the traditonal IDS server settings, it lowers network overhead. As a result, the suggested system only requires a little amount of resources to effectively detect malicious traffic.

In addition, we suggested a method for determining the ideal sampling rate for the IDS which is based on the total of false-negative rates.

The simulation results showed that the suggested IDS model may significantly increase its intrusion detection efficiency under medium network loads. By developing this model and the sampling scheme, we aim to enhance the detection efficiency in SDNs. In the future, we will keep working to enhance the system in order to cope with medium to heavy network loads.

Author Contributions: Conceptualization, X.Z.; Methodology, X.Z.; Supervision, Z.S.; Writing—original draft, H.S.; Writing—review & editing, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been performed in the Project "Research on Anomaly Network Traffic Detection Methods for Software Defined Networks" supported by National Natural Science Foundation of China (No. 61672299), and partly supported by the China Postdoctoral Science Foundation funded project (No. 2018M640509).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon reasonable request from the corresponding author.

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

References

1. Zhao, X.; Chen, S.; Yu, Y.; Sun, Z. Genetic Algorithm based Intrusion Detection System for Software-Defined Network Architecture. In Proceedings of the 2020 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, China, 18–20 December 2020; pp. 309–313. [CrossRef]
2. Janz, C.; Ong, L.; Sethuraman, K.; Shukla, V. Emerging transport SDN architecture and use cases. *IEEE Commun. Mag.* **2016**, *54*, 116–121. [CrossRef]
3. Freet, D.; Agrawal, R. A virtual machine platform and methodology for network data analysis with IDS and security visualization. In Proceedings of the SoutheastCon 2017, Concord, NC, USA, 30 March–2 April 2017; pp. 1–8.
4. Xue, Y.; Wang, Y.; Liang, J.; Slowik, A. A Self-Adaptive Mutation Neural Architecture Search Algorithm Based on Blocks. *IEEE Comput. Intell. Mag.* **2021**, *16*, 67–78. [CrossRef]
5. Xue, Y.; Xue, B.; Zhang, M. Self-Adaptive Particle Swarm Optimization for Large-Scale Feature Selection in Classification. *ACM Trans. Knowl. Discov. Data (TKDD)* **2019**, *13*, 1–27. [CrossRef]
6. Yang, Y.; McLaughlin, K.; Sezer, S.; Littler T.; Im, E.G.; Pranggono, B.; Wang, H. F. Multiattribute SCADA-Specific Intrusion Detection System for Power Networks. *IEEE Trans. Power Deliv.* **2014**, *29*, 1092–1102. [CrossRef]
7. Sun, T.; Zhang, J.; Yang, Y. Review on the development and future trend of the intrusion detection system (IDS). In Proceedings of the 2016 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 21–22 October 2016.
8. Seeber, S.; Stiemert, L.; Rodosek, G.D. Towards an SDN-enabled IDS environment. In Proceedings of the Communications & Network Security, Florence, Italy, 28–30 September 2015.
9. Tang, T.A.; Mhamdi, L.; Mclernon, D.; Zaidi, S.; Ghogho, M. Deep learning approach for Network Intrusion Detection in Software Defined Networking. In Proceedings of the International Conference on Wireless Networks & Mobile Communications (WINCOM), Fez, Morocco, 26–29 October 2016. [CrossRef]
10. Ping, W.; Chao, K.M.; Lin, H.C.; Lin, W.H.; Lo, C.C. An Efficient Flow Control Approach for SDN-Based Network Threat Detection and Migration Using Support Vector Machine. In Proceedings of the IEEE International Conference on E-business Engineering, Macau, China, 4–6 November 2017.
11. Ha, T.; Kim, S.; An, N.; Narantuya, J.; Jeong, C.; Kim, J.W.; Lim, H. Suspicious Traffic Sampling for Intrusion Detection in Software-Defined Networks. *Comput. Netw.* **2016**, *109*, 172–182. [CrossRef]
12. Yoon, S.; Ha, T.; Kim, S.; Lim, H. Scalable Traffic Sampling Using Centrality Measure on Software-Defined Networks. *IEEE Commun. Mag.* **2017**, *55*, 43–49. [CrossRef]
13. Silva, J.M.C. Computational weight of network traffic sampling techniques. In Proceedings of the 2014 (ISCC) IEEE Symposium on Computers and Communication, Funchal, Portugal, 23–26 June 2014.
14. Bartos, K.; Rehak, M. Towards Efficient Flow Sampling Technique for Anomaly Detection. *Int. Workshop Traffic Monit. Anal.* **2012**, *7189*, 93–106.
15. Ha, T.; Yoon, S.; Risdianto, A.C.; Kim, J.W.; Lim, H. Suspicious Flow Forwarding for Multiple Intrusion Detection Systems on Software-Defined Networks. *IEEE Netw.* **2016**, *30*, 22–27. [CrossRef]
16. Ahmed, M.; Mahmood, A.N.; Maher, M.J. An Efficient Technique for Network Traffic Summarization using Multiview Clustering and Statistical Sampling. *ICST Trans. Scalable Inf. Syst.* **2015**, *15*, e4. [CrossRef]
17. Chukwu, J.; Osamudiamen, O.; Matrawy, A. IDSaaS in SDN: Intrusion Detection System as a service in software defined networks. In Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS), Philadelphia, PA, USA, 17–19 October 2016.
18. Rengaraju, P.; Ramanan, V.R.; Lung, C.H. Detection and prevention of DoS attacks in Software-Defined Cloud networks. In Proceedings of the 2017 IEEE Conference on Dependable and Secure Computing, Taipei, China, 7–10 August 2017.
19. Dotcenko, S.; Vladyko, A.; Letenko, I. A fuzzy logic-based information security management for software-defined networks. In Proceedings of the International Conference on Advanced Communication Technology, Pyeongchang, Korea, 16–19 February 2014.
20. Boero, L.; Marchese, M.; Zappatore, S. Support Vector Machine Meets Software Defined Networking in IDS Domain. In Proceedings of the 2017 29th International Teletraffic Congress (ITC 29), Genoa, Italy, 4–8 September 2017.
21. Sayeed, M.A.; Sayeed, M.A.; Saxena, S. Intrusion detection system based on Software Defined Network firewall. In Proceedings of the International Conference on Next Generation Computing Technologies, Dehradun, India, 14–16 October 2016.
22. Prathibha, S.; Bino, J.; Ahammed, M.T.; Das, C.; Oion, S.R.; Ghosh, S.; Afroj, M. Detection Methods for Software Defined Networking Intrusions (SDN). In Proceedings of the 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 28–29 January 2022; pp. 1–6. [CrossRef]
23. Lin, Z.; Hongle, D. Research on SDN intrusion detection based on online ensemble learning algorithm. In Proceedings of the 2020 International Conference on Networking and Network Applications (NaNA), Haikou, China, 10–13 December 2020; pp. 114–118. [CrossRef]

24. Shirali-Shahreza, S.; Ganjali, Y. FleXam: Flexible sampling extension for monitoring and security applications in openflow. *ACM Spec. Interest Group Data Commun.* **2013**, 167–168. [CrossRef]
25. Suh, J.; Kwon, T.T.; Dixon, C.; Felter, W.; Carter, J. OpenSample: A Low-Latency, Sampling-Based Measurement Platform for Commodity SDN. In Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems (ICDCS), Madrid, Spain, 30 June–3 July 2014.
26. Karakus, M.; Durresi, A. A Scalability Metric for Control Planes in Software Defined Networks (SDNs). In Proceedings of the 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), Crans-Montana, Switzerland, 23–25 March 2016.
27. Hu, J.; Lin, C.; Li, X.; Huang, J. Scalability of control planes for Software defined networks: Modeling and evaluation. In Proceedings of the 2014 IEEE 22nd International Symposium of Quality of Service (IWQoS) Hong Kong, China, 26–27 May 2014.
28. Zuo, Q.; Chen, M.; Ding, K.; Xu, B. On generality of the data plane and scalability of the control plane in software-defined networking. *China Commun.* **2014**, *11*, 55–64. [CrossRef]
29. Singh, K.; Guleria, A.; Bassiouni, M. A Scalable Peer-to-Peer Control Plane Architecture for Software Defined Networks. In Proceedings of the 15th IEEE International Symposium on Network Computing and Applications (NCA 2016), Cambridge, MA, USA, 31 October–2 November 2016.
30. Birkinshaw, C.; Rouka, E.; Vassilakis, V.G. Implementing an Intrusion Detection and Prevention System Using Software-Defined Networking: Defending Against Port-Scanning and Denial-of-Service Attacks. *J. Netw. Comput. Appl.* **2019**, *136*, 71–85. [CrossRef]
31. Hande, Y.; Muddana, A. A Survey on Intrusion Detection System for Software Defined Networks (SDN). *Int. J. Bus. Data Commun. Netw.* **2020**, *16*, 28–47. [CrossRef]
32. Latah, M.; Toker, L. An Efficient Flow-based Multi-level Hybrid Intrusion Detection System for Software-Defined Networks. *CCF Trans. Netw.* **2018**, *3*, 261–271. [CrossRef]
33. Alenazi, M.J.F. Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks. *Future Internet* **2021**, *13*, 111. [CrossRef]
34. Bao, H.; Pham-Quoc, C.; Thinh, T.N.; Thoai, N. A Secured OpenFlow-Based Switch Architecture. In Proceedings of the International Conference on Advanced Computing & Applications, Can Tho, Vietnam, 23–25 November 2017.
35. Fan, Y.; Liao, Q.; He, Q. Research and Comparative Analysis of Performance Test on SDN Controller. In Proceedings of the 2016 First IEEE International Conference on Computer Communication and the Internet, Wuhan, China, 13–15 October 2016.
36. Azzouni, A.; Braham, O.; Trang, N.; Pujolle, G.; Boutaba, R. Fingerprinting OpenFlow Controllers: The First Step to Attack an SDN Control Plane. In Proceedings of the Global Communications Conference, Washington, DC, USA, 4–8 December 2017.
37. Schoenecker, S.; Luginbuhl, T. Characteristic Functions of the Product of Two Gaussian Random Variables and the Product of a Gaussian and a Gamma Random Variable. *IEEE Signal Process. Lett.* **2016**, *23*, 644–647. [CrossRef]