Jia *et al. Cybersecurity* (2022) 5:27
https://doi.org/10.1186/s42400-022-00128-7

Cybersecurity

RESEARCH Open Access

A lightweight DDoS detection scheme under SDN context

Kun Jia[1,2], Chaoge Liu[3], Qixu Liu[3], Junnan Wang[2,3], Jiazhi Liu[1,2] and Feng Liu[1*]

Abstract

Software-defined networking (SDN), a novel network paradigm, separates the control plane and data plane into different network equipment to realize the flexible control of network traffic. Its excellent programmability and global view present many new opportunities. DDoS detection under the SDN context is an important and challenging research field. Some previous works attempted to collect and analyze statistics related to flows, usually recorded in switches, to address DDoS threats. In contrast, other works applied machine learning-based solutions to identify DDoS and achieved promising results. Generally, most previous works need to periodically request flow rules or packets to obtain flow statistics or features to detect stealthy exceptions. Nevertheless, the request for flow rules is very time-consuming and CPU-consuming; moreover may congest the communication channel between the controller and the switches. Therefore, we present FORT, a lightweight DDoS detection scheme, which spreads the rule-based detection algorithm at edge switches and determines whether to start it by periodically retrieving the ports state. A time-series algorithm, ARIMA, is utilized to determine the port statistics adaptively, and an SVM algorithm is applied to detect whether a DDoS attack does occur. Representative experiments demonstrate that FORT can significantly reduce the controller load and provide a reliable detection accuracy. Referring to the false alarm rate of 1.24% in the comparison scheme, the false alarm rate of this scheme is only 0.039%, which significantly reduces the probability of false alarm. Besides, by introducing the alarm mechanism, this scheme can reduce the load of the southbound channel by more than 60% in the normal state.

Keywords: DDoS, Port monitoring, Flow rule, Software defined network, MiniNet, RYU controller, Detection

Introduction

Software-defined networking (SDN) is an innovative paradigm that draws lots of attention from both academia and industry (Yan et al. 2015). It decouples the data and control plane into separated devices to define traffic management regulation in the programmable form at the controller side. SDN can realize various applications such as route management, visualization, and traffic monitoring based on its global network view. Hence it is widely used in IoT (Wu et al. 2018; Balasubramanian et al. 2021; Yin et al. 2018), cloud (Leng et al. 2019; Wang et al. 2015) and some other crucial networks. SDN brings considerable benefits in practice and presents many new opportunities for security research, such as DDoS detection under SDN Context. According to Ubale and Jain (2020), DDoS is one of the most critical security issues in the current SDN research field. It also concludes that SDN has very effective characteristics, including separation of the control plane from the data plane, global view of the network, the programmability of network, traffic analysis based on software, and dynamic network update policy, that benefit for defeating DDoS attacks. Compared with DDoS detection based on a traditional network that usually can only find exceptions at separate endpoints, DDoS detection based on SDN can utilize its global view to cooperate with all switches to detect attacks. We can track the attack clues from the attack source to the attack

*Correspondence: liufeng@iie.ac.cn

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Beijing, China
Full list of author information is available at the end of the article

target and synthesize their results, making our detection work more credible.

DDoS is a cyberattack where malicious users send flooding fake messages to a target with high frequency. As a result, it causes legitimate users could not access the service as usual. The report of NSFOCUS (2021) shows that the volume of DDoS attacks has greatly increased recently. With increasing concerns on DDoS, DDoS detection using SDN has attracted the attention of many researchers (Jianfeng et al. 2020). Various methods have been proposed subsequently. Some researchers focus on statistics -based methods that collect and analyze statistics related to attack-free flows. If incoming flows do not comply with the behavioral profile of previous traffic, it is determined as malicious. These methods usually perform efficiently in detecting and mitigating DDoS attacks. However, some challenges might be faced on account that the specified thresholds usually cannot be adaptively adjusted (Eliyan and Di Pietro 2021).

As an alternative to the above statistics -based methods, some research focuses on machine learning-based algorithms to identify DDoS attacks. Support vector machines (SVM), neural networks (NN), and self-organizing maps (SOM) are the most frequently used algorithms, and they usually perform well in accuracy and false-positive rate. However, these machine learning-based approaches may introduce challenges in concurrently managing the SDN controller, and the cited machine learning algorithms, hence likely imposing some latency in handling the legitimate communications (Eliyan and Di Pietro 2021).

In addition to the challenges mentioned above, statistics -based methods and machine learning-based methods may also face the challenge of high resource consumption because both patterns of methods may require the periodical request of flow rules or packets. The periodical request means more precise and real-time monitoring, while it increases the load of bandwidth between the controller and switches, controller memory, and controller CPU. In addition, most detection methods rely on fine-grained routing management (Zhao et al. 2020), hence the number of flow rules requested will be huge, causing the process of transmitting and handling to be very inefficient. Although some work studied how to spread the load of measurement across the whole network (Sekar et al. 2008; Chang et al. 2011; Zhang 2013d), they usually require switches with programmable capabilities.

This paper aims to address the problems of high resource consumption and inefficiency of existing DDoS detection methods in SDN. Thus, we propose a novel DDoS detection scheme named FORT. Unlike existing detection methods, FORT determines whether to trigger flow rule-based detection by observing the port statistic. Once the observed indicators exceed specified thresholds, FORT starts to locate the abnormal port. It is worth noting that FORT implements situational awareness from both the attack source and the attack target, so the indicators need to be distinguished against different perspectives. However, according to the value of the indicators, it is easy to determine whether the abnormal port is the attack source or the attack target. After that, FORT only needs to awaken related switches and request the traffic related to the suspicious target or source. Generally, the main contributions of this paper are summarized as follows:

1. We propose a lightweight detection scheme that does not rely on periodical request of flow rules or packets as existing researches do. Specifically, FORT spreads the rule-based detection algorithm at edge switches and determines whether to start it by periodically retrieving the ports state. Compared with the request of flow rules, retrieving ports state costs much less resource and time. And spreading the detection algorithm avoids the inefficiency of retrieving flow rules;
2. We give a quantitative description of the indicators of the port statistics through a time series algorithm, ARIMA. It helps the controller identify abnormal port and whether it belongs to the attack source or the attack target. Then, the controller can trigger the detection process of the switch to which the port belongs;
3. We propose a lightweight detection algorithm that detects attack events at the victim side and locates attack sources. Since we treat transmitted and received flow rules separately, it is trustworthy that our detection algorithms can detect attacks with good performance. Experiments show that the detection accuracy for multiple attack types are above 97.26% from the victim side when idle-timeout equals 5, and *FPR* are only 0.039% if we comprehensively consider the anomaly from the perspective of attacker side and victim side;

The rest of this paper is organized as follows. "Background and related work" section introduces DDoS detection's background and related work under the SDN context. Then, "Motivation and system design" section describes our motivation and introduces the system design in detail. Subsequently, we validate the availability and performance of FORT in "Performance evaluation" section. Finally, we conclude this paper in "Conclusion" section.

Background and related work

In this section, we first give a simple description of SDN and then review related work of DDoS detection under the SDN context.

Background

Typical SDN consists of a controller, some switches, and users as Fig. 1. The communication channel between the controller and switches is called the southbound channel. Its communication mode is specified by OpenFlow protocol (2021). OpenFlow allows a controller to instruct routing, request the port and flow statistics , and handle new-arrival packets. When a new packet of a flow arrives at a switch, this switch will trigger a packet-in packet and request the controller to install the matching rules. Then, the remaining packets of the flow will be directly handled by the rule. In this way, the controller owns global management of network traffic.

There are also two ways that the controller could proactively monitor traffic statistics, port statistics request and flow statistics request, which is introduced in detail as the followings:

1. port statistics request: is a way to get grim statistics . Each port of switches will measure packets and bytes transmitted and received. Due to the simplicity of stored information and the small number of ports, port statistics request is usually very efficient;
2. flow statistics request: is a way to get precise statistics . Each flow rule is defined with elements such as IP source, IP protocol, and TCP destination port. Furthermore, the flow rule counts the number of packets and bytes that match it. The time to request the flow rules is positively correlated with the number of flow rules.

In addition to traffic monitoring capability, SDN also supports topology monitoring capability, which maintains the topology of users, links, and switches. For example, it could discover new hosts and locate their sites via packet-in messages embedded with an ARP payload. Furthermore, the link information is maintained via LLDP packets. In these ways, the controller owns the global view of the SDN network.

Related work

Distributed denial of service (DDoS) is a severe threat to network security. Therefore, it is crucial to detect and mitigate DDoS in a timely considering that DDoS has destructive power on network infrastructures. Maninder et al. and Eliyan et al. presented a classification of DDoS detection under SDN context, namely statistics -based, machine learning-based methods (Singh and Bhandari 2020; Eliyan and Di Pietro 2021). Moreover, they also categorize the DDoS methods by packet-based, flow-based, and packet-in based which are the three different ways to inspect the network traffic.

Statistics-based methods

Statistics-based methods typically collect and analyze statistics related to attack-free flows. Some benchmarks or indicators are converted from these statistics and used as traffic profiles. If incoming flows do not

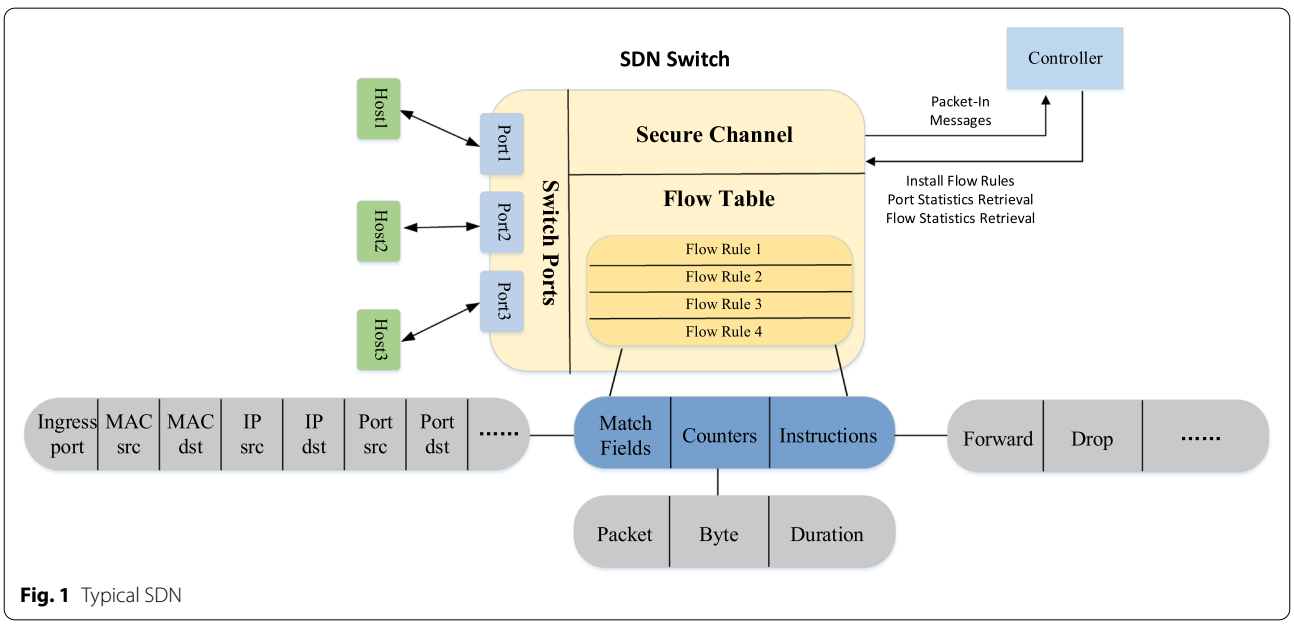

Fig. 1 Typical SDN

comply with the benchmark profile, they will be determined malicious.

Dao et al. (2015) proposed one such statistics -based method using the analysis result of a real-life dataset. They have implemented a table and placed it inside the controller to track the source IP of incoming packets. For each unique IP address, the characteristics of DDoS addresses are to initiate less than k connections (in a constant duration) and to generate less than n packets per connection, where k is the average connections of "frequent" users and n is the minimum number of packets per connection. Likewise, Kalkan et al. (2018) proposed a joint entropy-based security scheme (JESS) to detect DDoS attacks. Specifically, the packet header information is collected to obtain the joint entropy benchmark. JESS decides whether DDoS happened by comparing the difference of joint entropy in the nominal period and suspicious period.

The above research mechanisms are based on packet inspection. This type of mechanism is necessary to check all packets for precise detection, even when the network is not attacked, which is very inefficient for real-time detection.

A solution to that is not inspecting packets but the statistical information stored in flow rules. Wang et al. (2015) proposed to deploy their method at the edge switches and designed an entropy-based algorithm to perform lightweight anomaly detection. However, edge switches could not acquire a global view like controller, which means they hardly detect attack through another path. Furthermore, retrieving flow rule statistics periodically also cannot avoid inefficiency altogether. To improve it, Giotis et al. (2014) combined flow rule request and sFlow together to reduce processing overhead in native flow rule statistics collection. Nevertheless, sampling using sFlow commonly imposes a high false-positive rate.

Another idea is inspecting the packet-in message. For example, You et al. (2017), extract three types of entropy from the packet-in messages and perform real-time attack detection through confidence interval. Nevertheless, packet-in messages only carry the information of new-arrival packets, which cannot sufficiently reflect the overall traffic state.

Generally, the statistics -based methods have provided reliable solutions for DDoS detection in SDN. However, most of them need to request packets, flow rules, or packet-in messages periodically, even in a normal state, which costs many resources. Besides, statistics -based methods usually rely on fixed thresholds set empirically and do not meet the requirement of adaptation.

Machine learning-based methods

Machine learning-based methods usually learn from a large set of data collected in advance to identify an attack and regular patterns. Then trained model will be used in practice and make decisions requiring no explicit instructions.

Typically, Braga et al. (2010) proposed a lightweight approach, which requests flow rules statistics periodically and converts them to six prominent features, then the SOM algorithm is used to classify these samples. Likewise, Ye et al. (2018) extracted 6-tuple features from flow tables, and then the DDoS attack model was built by combining the SVM classification algorithms. Finally, Hu et al. (2017) acquire network traffic information through the SDN controller and sFlow agents. Then an entropy-based method is used to measure network features, and the SVM classifier is applied to identify network anomalies.

As mentioned before, these methods all require periodical flow rules requests, which is inefficient. Thus, machine learning-based methods also try to detect using packet-in messages. For example, Mehr and Ramamurthy (2019) measure the distribution of the source IP address, source port, destination IP address, destination port, and train the model with the SVM algorithm.

Differently, Xu and Liu (2016) tried to locate the victim and attacker. In their method, spatial adaptation is adopted to manage different flows. Precisely, coarse-grained rules to monitor expected flows and fine-grained rules to monitor traffic with a high likelihood of under DDoS attack. Yang's method inspires that the detection algorithm could only focus on the traffic going to the suspicious victim.

Generally, machine learning-based methods successfully avoid empirical thresholds yet introduce new challenges. First, there are difficulty in concurrently managing the SDN controller and the cited machine learning algorithms, hence likely imposing some latency in handling the legitimate communications (Eliyan and Di Pietro 2021). Second, flow rule-based methods still require frequent requests, and the alternative methods, for example, packet-in-based methods, usually perform not well in detecting DDoS with real IP since it only requests the new-arrival packets.

Summary

In response to the harm of DDoS, effective detection methods are essential for protecting the network infrastructures. Previous work of statistics -based methods usually is faced with the challenge of specified thresholds. However, machine learning-based methods impose latency in handling legitimate traffic. In addition, from the perspective of feature source, detection methods under SDN context can be categorized as packet-based methods, flow rule-based methods, and packet-in-based

methods. Packet-based methods usually ask for too much resource; packet-in-based methods can only roughly reflect the current flow state, and yet flow rule-based methods costs medium resources and can provide a precise reflection of flow state, relatively.

Thus, FORT chooses the flow rule-based strategy to ensure reliable detection efficiency and accuracy. To further avoid its inefficiency in retrieving flow rules, we explore how to process flow statistics at the local switch. However, even if processed locally periodically, it will bring many unnecessary burdens to the switch. Therefore, a trigger mechanism, which instructs which switch and when to start the process, needs to be established.

Motivation and system design

In this section, we describe our DDoS detection scheme in detail. We first discuss the motivation for this paper, followed by the system design and details.

Motivation

Since we do not want to request flow rules frequently, deploying the detection algorithm locally at the edge switches is necessary. However, if each switch periodically retrieves flow rules for detection, it will cause much unnecessary waste of resources. Hence, a lightweight trigger mechanism is necessary. For example, the controller could awaken the related switch and start the subsequent detection process when the indicators are abnormal. Luckily, SDN supports the fast and accurate request of port statistics . Each switch port in SDN is equipped with a meter, which stores the received and transmitted packets and bytes. These records can roughly reflect the current flow condition. Most importantly, retrieving the port statistics is very efficient because the information saved is rough, and the number of ports is relatively tiny. Therefore, the port monitoring procedure can quickly and precisely identify a suspicious traffic anomaly and distinguish whether the abnormal port belongs to the attack source or the attack target.

Once indicators of port statistics exceed specified thresholds, subsequent DDoS detection is triggered. Controllers will instantly inform the switch related to the abnormal port, and the switch will retrieve flow rules and extract corresponding features. These features are then used as inputs to the final detection algorithms. Because the characteristics obtained by coarse-grained monitoring are very rough, there may be unexpected alarms for events such as a flash event, so fine-grained monitoring and detection is essential (Wang et al. 2018).

It is worth noting that in the procedure of port monitoring, indicators for judging abnormal flow are expected to be adaptive. Thus, we use the time series model to predict the future state value. Then, by comparing the difference between the predicted value and the actual value, we can check the traffic anomaly. Furthermore, since the time series model uses the correlation between the current state and historical states, the value of the next stage always maintains stationary with the values of the previous stages; hence it can dynamically adapt to the slight flow fluctuation.

System design

Generally, FORT is constituted with two procedures: port monitoring, DDoS detection as Fig. 2. Specifically, a port statistics collector starts to work periodically in the port monitoring procedure, then volume and asymmetrical indicators will be extracted. An ARIMA algorithm is subsequently used to determine anomaly and whether the abnormal port belongs to the attack source or target.

Once the anomaly is confirmed, the controller will inform related switches and initiate its DDoS detection procedure. Subsequently, the switch collects suspicious flow rules corresponding with the abnormal port and extracts distinguished features. Then, based on the previous determination of attack source or target, these features will be sent to the corresponding algorithms. We will describe these procedures in detail in the following.

Port monitoring

This procedure consists of a periodical port statistics collection, where FORT requests the necessary information to create a traffic profile. Mainly, port statistics are gathered with a message Onp_Port_Stats and then delivered to get the indicators of volume and asymmetry. As we know, a typical DDoS commonly shares two types of anomaly: volume anomaly and asymmetry anomaly. The former refers to the abnormal increase of network traffic, and the latter refers to the extreme imbalance of source IP, destination IP, and volume in upstream and downstream traffic. We describe how they are obtained in the following parts.

Port statistics collection

Port statistics can be more quickly and efficiently obtained than flow table statistics for two reasons. First, switches usually have fewer ports than rules, especially in a fine-grained managed network where flow rules are exact-match defined. Second, the statistics recorded by ports are much less than the statistics recorded by flow rules. Thus, FORT chooses to request port statistics to avoid time-consuming and CPU-consuming.

Two types of rwa data are collected, including the number of received packets and transmitted packets. As mentioned earlier, FORT will extract features from the

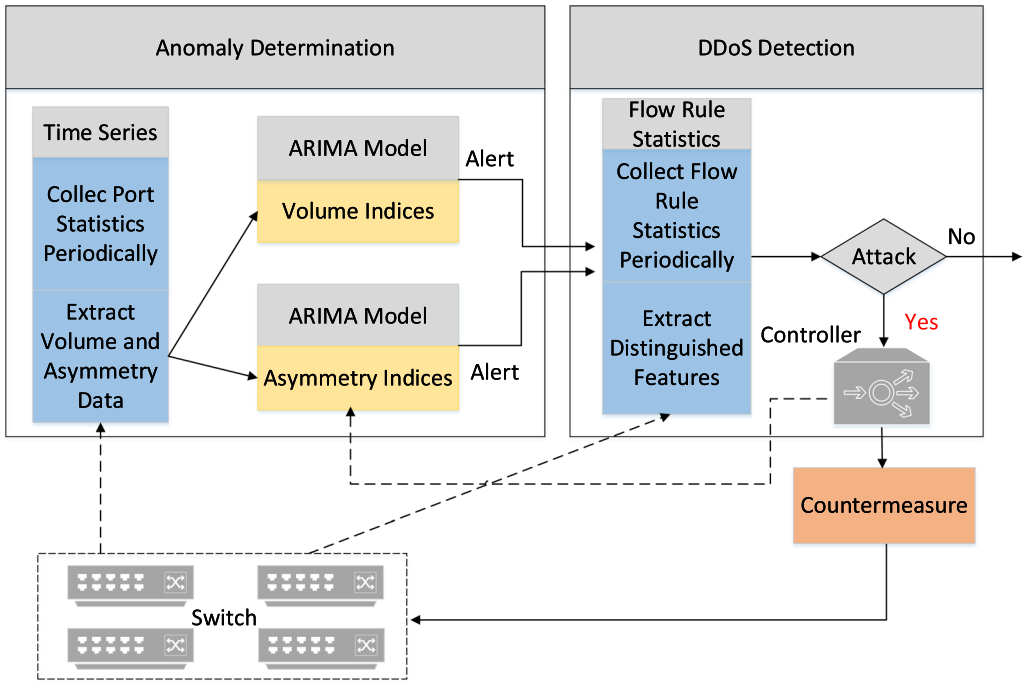

Fig. 2 System design

perspectives of volume and asymmetry. Since the normal network interaction usually fluctuates with time, it is impossible to specify a fixed threshold for the scale of network traffic. In addition, intuitively, we believe that the current traffic is related to the traffic in the previous periods. Therefore, we introduce a time series prediction algorithm, ARIMA, to determine the volume and asymmetrical anomaly. The components in detail will be described in section "ARIMA" .

ARIMA

The ARIMA algorithm is a trend prediction algorithm based on time series analysis. It is to establish the relationship between current data and historical values and realize a prediction of a future state. The ARIMA model must meet the requirements of stationarity, and autocorrelation (Perraudin and Vandergheynst 2017). The work of Box et al. (2015) and Cortez et al. (2006) has proven that time series analysis is available for traffic forecast, and Fouladi et al. (2020) show that it is effective in SDN and DDoS detection.

The ARIMA algorithm is characterized by a three-tuple $< p, d, q >$, where p is the number of auto-regressive (AR) terms, d is the order of differences required for stationarity, and q is the number of moving average (MA) or lagged forecast errors terms. Let X_t be a stationary data series, and a typical ARIMA model is represented as Eq. 1:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t - \sum_{i=1}^{q} \theta_i \epsilon_{t-i}, \tag{1}$$

where ϵ_t denotes the forecast errors in the AR model, which is introduced in ARIMA to eliminate random fluctuations, c is a constant, ϕ_i and θ_i are the coefficient waiting to be trained. If the time-series data X_t is not stationary, a difference method should be applied to reduce the fluctuation of original data. Expressed in math, the first-order difference is $\Gamma_t = X_t - X_{t-1}$. The second-order difference is the same operation as the first-order difference, while the data it processes is Γ_t. ADF unit root test is generally used to check the stationarity of time series data. Generally, when the *t-statistics* is smaller than the critical value 1% of any confidence, time-series data is supposed to be stationary.

Since the controller could save the historical data of different ports, we can easily use the ARIMA algorithm to predict the traffic in the next stage. However, if a DDoS occurs, the network traffic to a victim will increase sharply, making the ARIMA model prediction invalid, i.e., the prediction error will be highly exaggerated. Based on this, FORT can precisely detect the volume and asymmetry anomaly.

Indicators selection

To do a reliable alarm, we first need to select proper indicators. As previously stated, FORT will capture the flow

volume and the asymmetry indicators. $\{R_{t,i}\}$ denotes the time series of the number of received packets, $\{T_{t,i}\}$ denotes the time series of the number of transmitted packets and $\{\lambda_{t,i}\}$ denotes the time series of $\{\frac{T_{t,i}}{R_{t,i}}\}$ at port i, then the indicators can be expressed as the followings:

1. The normalized error of received packets ($\Delta R_{t,i}$): describes the normalized error between the actual received packets and predicted received packets. Considering that the scale of traffic varies greatly in different scenarios, this value will be normalized:

$$\Delta R_{t,i} = \frac{R_{t,i} - \hat{R}_{t,i}}{\hat{R}_{t,i}}, \tag{2}$$

2. The normalized error of transimitted packets ($\Delta T_{t,i}$): describes the normalized error between the actual transimitted packets and predicted transimitted packets. Considering that the scale of traffic varies greatly in different scenarios, this value will be normalized:

$$\Delta T_{t,i} = \frac{T_{t,i} - \hat{T}_{t,i}}{\hat{T}_{t,i}}, \tag{3}$$

3. The error of asymetrical ratio ($\Delta \lambda_{t,i}$): describes the error between the actual ratio and the predicted ratio. Here, ratios mean the ratio of traffic transmitted and received by the ports. In order to better quantify the ratio difference when it is less than 1, a logarithmic function is used:

$$\Delta \lambda_{t,i} = \ln \lambda_{t,i} - \ln \hat{\lambda}_{t,i}. \tag{4}$$

When a DDoS attack happens, compared to the incoming packets, packets originating from a victim would be much less (Xu and Liu 2016), thus from the perspective of edge switches at the victim side, received packets would be much less than transmitted packets, i.e., the ratio $\lambda_{t,i}$ increases unpredictably. Further, we can draw a conclusion that $\Delta R_{t,i}$ and $\Delta \lambda_{t,i}$ will suddenly become larger. Once they exceed defined thresholds ΔR_0 and $\Delta \lambda_{upper}$ respectively, FORT determines that there is a traffic anomaly and concludes the abnormal port is located near the victim. The same analysis can be used for the attacker side, but $\Delta T_{t,i}$ and $\Delta \lambda_{t,i}$ need to be noticed this time. If $\Delta T_{t,i}$ exceeds ΔT_0 and $\Delta \lambda_{t,i}$ is below the lower bound $\Delta \lambda_{lower}$, we determine something abnormal has happened. The above thresholds can be selected according to the requirement of the scenarios. If DDoS needs to be strictly supervised, the thresholds should be strictly defined; the threshold can be loosely defined if the vendor aims to reduce the false alarm rate first.

DDoS detection

In the procedure of DDoS detection, features will be extracted from flow table statistics. Since only traffic related to the suspicious victim or attacker is necessary for detection, we only need to request flow rules related to the abnormal port.

Features

From the perspective of a port such as p, we define transmitted flows and received flows. That is, transmitted flows of p means flows sent from the port p, and received flows of p means the flows received by the port p. Furthermore, the flow rules associated with transmitted flows are called transmitted rules $\mathbb{T}$ and the flow rules associated with received flows are called received rules $\mathbb{R}$. Their source headers ($IP\_src, Port\_src, Protocol$) are denoted by T_{src} or R_{src}. T_{dst} or R_{dst} have similar definitions, yet it is worth noting that only flow rules that match more than two packets will be recorded in T_{dst} or R_{dst}. Because according to Dao et al. (2015), most of the normal traffic includes at least 3 packets. Ten representative features are extracted from these transmitted rules and received rules of the abnormal port captured at the port monitoring procedure. They are expressed as follows:

- Median Packets per Transmitted Flow (*MPTF*):
  One of the main characteristics of DDoS attacks is IP spoofing, which prevents the task of tracing the attacker's source. This method is widely used in SYN flood, UDP flood, and ACK flood, so for the port near the victim side, usually median packets per transmitted flow will decrease sharply in a DDoS; In contrast, the median number of packets per transmitted flow in a normal state is larger than 5 according to Dao et al. (2015); Yet for the port near attacker side, the conclusion is not tenable. In order to acquire this feature, we sort the flows in ascending order based on the number of packets per flow. Then the median is computed;
- Median Bytes per Transmitted Flow (*MBTF*):
  Similar to the median of packets per flow, for the port near the victim side, the median of bytes per transmitted flow will decrease sharply owing to IP spoofing technique; Research shows over 90% of attack packets are under 100 bytes, while normal packets vary between 100 and 1,200 bytes (Doshi et al. 2018). Nevertheless, the conclusion is not tenable for the port near the attacker side. The calculation process is similar to that of *MPTF*;

- Median Duration per Transmitted Flow (*MDTF*):
  Flooding DDoS usually creates many rules, and since the attacker has no motivation to establish regular interactive communication with the victim, the duration of transmitted flows is usually relatively short for the port near the victim side. However, the conclusion is not tenable for the port near the attacker's side. The calculation process is similar to that of *MPTF*;
- Median Packets per Received Flow (*MPRF*):
  Similar analysis as *MPTF*;
- Median Bytes per Received Flow (*MBRF*):
  Similar analysis as *MBTF*;
- Median Duration per Received Flow (*MDRF*):
  Similar analysis as *MDTF*;
- Ratio of Transmitted Flow (*RTF*):
  This feature reflects the proportion of transmitted flows to all flows. As it is abnormally high or low, DDoS is likely to occur. Commonly, for the port near the victim side, this value closes to 1, yet for the port near the attacker side, this value closes to 0;
- Ratio of Transmitted Packets (*RTP*):
  This feature reflects the proportion of transmitted packets to all packets. As it is abnormally high or low, DDoS is likely to occur. Commonly, for the port near the victim side, this value closes to 1, yet for the port near the attacker side, this value closes to 0;
- Ratio of Successful Request (*RSR*):
  The terminal communicates with the outside through the bound port. The ratio of successful requests reflects the probability that the terminal successfully establishes a connection with the outside, which is a very suitable indicator of DDoS. Due to the source IP forgery technology, most of the requests from the attacker will not get a response. Thus, this ratio will be minimal for the port near the attacker side. However, it may not happen for a port near the victim's side or in a normal state. The value is computed as Eq. 5:

$$RSR = \frac{|\mathbb{R}_{src} \bigcap \mathbb{T}_{dst}|}{|\mathbb{R}_{src}|} \tag{5}$$

- Ratio of Successful Access (*RSA*):
  Each switch port is bound to one or more terminals; the ratio of successful access reflects the probability that these terminals are successfully accessed from outside. Due to the limitation of victim connection resources and the source IP forgery technology, the accessibility in typical DDoS scenarios is relatively poor. Thus, this ratio will be minimal for the port near the victim side. Nevertheless, it may not happen for a port near the attacker's side or in a normal state. For an attack like SYN DDoS, illegal access may also trigger responsive packets. However, the connection cannot be completely established due to the source IP forgery technology, so the number of packets in the communication flow is minimal. When we calculate *RSA*, we do not involve these flows, which can correctly reflect the proportion of legitimate flows. The value is computed as Eq. 6:

$$RSA = \frac{|\mathbb{R}_{dst} \bigcap \mathbb{T}_{src}|}{|\mathbb{T}_{src}|} \tag{6}$$

Other works usually include the entropy of the source IP and the entropy of the destination IP or their variants, which FORT does not use on account that we only study the flow related with abnormal port; these characteristics are no longer applicable. For example, the entropy of the source IP will be relatively large, whether it is in the normal state or not.

Idle-timeout

Timeout is a crucial property defined in a flow entry. There are idle-timeout and hard-timeout. Idle-timeout refers to the idle connection state. If no packet matching the rule is received after the specified time, the rule will be recycled. Furthermore, hard-timeout refers to the most extended lifetime of a flow rule, regardless of whether matching packets are continuously received. Therefore, the timeout value positively correlates with the number of flow rules. Conversely, the packet-in frequency negatively correlates with the timeout value, considering that a small value means switches will frequently trigger packet-in messages. Due to the existence of idle-timeout, hard-timeout is usually set very large, so we only need to consider the setting of idle-timeout. To configure a "good" idle-timeout, we need to find a trade-off between packet-in frequency and the number of flow rules. In addition, the median duration per flow we extracted is also closely related to its value. In order to distinguish attack traffic from legitimate traffic, we tend to use a larger idle-timeout since a more significant value usually reflects the current flow state more stably.

Classifiers

As mentioned above, our classifiers module was implemented using the Support vector machine (SVM). SVM can find the most apparent hyperplane to classify the data in the high-dimensional plane. It is simple, usable, and robust, therefore, widely used in DDoS detection under SDN context (Eliyan and Di Pietro 2021). The features at different sides (victim and attacker) will be classified and tested with different classifiers in our implementation. Since some features have nonlinear correlation, our data are linearly nonseparable; thus, an "rbf" kernel function is finally chosen.

Performance evaluation

Several experiments are carried out to validate the feasibility and efficiency of FORT. First, we verify the availability of ARIMA. Then, packet numbers and asymmetry ratio are collected to compare their prediction values. Then we verify the impact of idle-timeout and compare the SDN resource consumption when configuring different timeout values. Finally, we replay real-life background traffic and generate multiple types of DDoS attacks in a simulated SDN scenario so as to judiciously examine the performance of our detection algorithm and compare our scheme with the existing scheme of Ye et al. (2018) which also adopt the SVM classifier.

Platform and dataset

Our SDN environment is based on Mininet, a commonly used network simulation platform, which supports network topology construction with a graphical interface. In addition, RYU, a commercial switch developed on python, is integrated as a remote SDN controller. The experiment platform is run on an Ubuntu PC with 8GB RAM and Intel Core i7-8550U 2.00 GHz CPU.

Figure 3 depicts the topology used for these experiments. The topology is the same version of Braga et al. (2010). Precisely, it consists of three networks where botnet lurks in Network 2, and the attack target is Network 1.

To simulate the regular network activity, the public datasets from MAWI (Working Group Traffic Archive) (2021) are acquired. MAWILab works on network traffic measurement, analysis, evaluation, and verification in the long term on the global Internet. Part of the data collected on June 20 and June 17, 2020, is utilized to generate background traffic in our experiments. Since we did not modify all the IP addresses in the dataset, the replayed traffic can still represent the normal traffic state. Moreover, a large-scale topology is successfully simulated in practice even if we built a small topology.

To simulate some types of DDoS, the public datasets from CIC DDoS (Sharafaldin et al. 2019) are acquired. The taxonomy of DDoS attacks is defined in terms of reflection-based and exploitation-based attacks. This dataset incorporated representative reflective-based attacks and exploitation-based attacks, such as PortMap, NetBIOS, LDAP, MSSQL, NTP, DNS, SNMP, SYN flood, UDP flood, and UDP-Lag. By configuring the parameters of TcpReplay, we can easily customize DDoS of different types and scales.

Availability of ARIMA

To verify the availability of ARIMA, we deploy a port statistics collector which requests port statistics every 3 s. Then, the ratio of packets transmitted and received and the number of packets by ports are calculated and used as the input of ARIMA. Three indicators over time are shown in Fig. 4, which are depicted by blue lines. It was

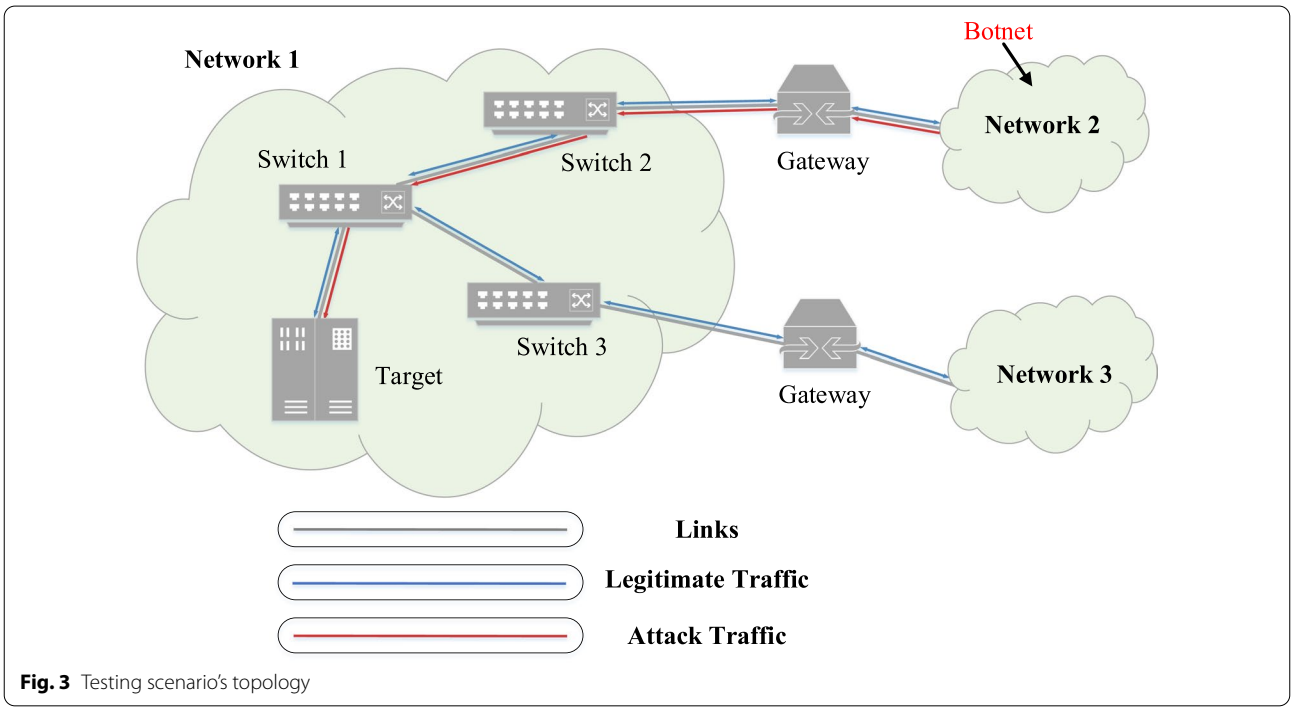

Fig. 3 Testing scenario's topology

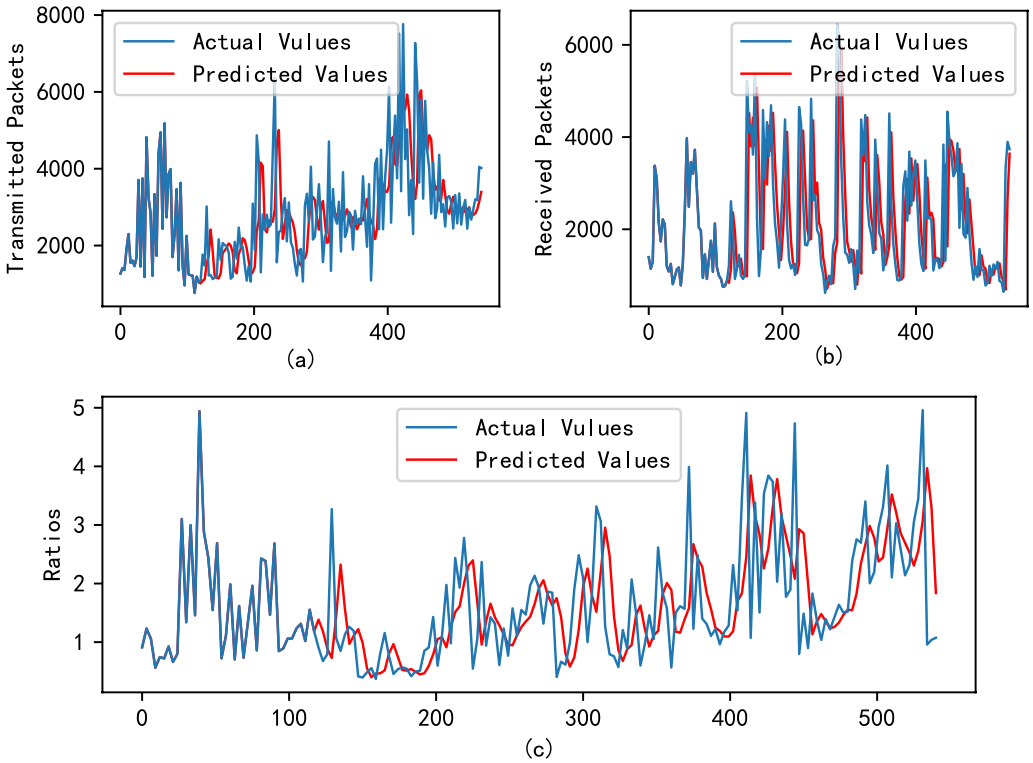

Fig. 4 Comparison between predicted values and actual values

observed that they are not stationary; thus, the first-order difference is required. We choose the "historical" 50 data as the training dataset. Every time the "current" data is predicted, the actual "current" data is added in the "historical" datasets until no data is left. The comparison of actual and predicted values are also shown in Fig. 4, where the red lines represent the predicted values.

As can be seen from the figure, the trend of the predicted line is very consistent with that of the actual line. Thus, we believe ARIMA could make a good prediction of the "current" condition from the historical record. Furthermore, the predicted error could be used as an indicator to determine anomalous traffic.

Impact of timeout

To verify the impact of the timeout, we count the number of packet-in messages and flow rules consumed every 5 s with five timeouts: 1 s, 5 s,10 s,15 s, and 30 s. Results are given in Fig. 5.

Figure 5a depicts the number of packet-in messages over time with different idle-timeout values. We can see that the five lines are jagged and almost inseparable, and even the fluctuation trend is almost synchronous. It means, when idle-timeout is above 1, the number of packet-in messages is not affected by idle-timeout. This is because the interval between adjacent packets of the same flow is less than 1 s. The same analysis process could be made about the Fig. 5b. However, unlike the number of packet-in messages, the flow rules show an obvious hierarchy. The line with a high idle-timeout is obviously above the low idle-timeout. It proves that idle-timeout greatly impacts the number of flow rules. According to the above conclusion, we tend to configure a small idle-timeout to save the resource. However, it cannot be ignored that idle timeout may also affect the detection effect. Intuitively, a larger value may perform better because it could precisely reflect the traffic state for a longer time. It will be verified at the next stage.

Performance of detection

Then, the performance of our detection mechanism was evaluated through a simulated DDoS attack. As with conventional DDoS attacks, a victim is selected in advance, and we generate DDoS traffic through a famous tool, hping3 (Sanfilippo 2021) or replaying public datasets. Table 1 presents the generated attacks types and their fundamental attributes, including attack duration, packet size, and attack rate in the training procedure, where $n\times$ in the attribute represents n times of regular traffic. According to the statistics

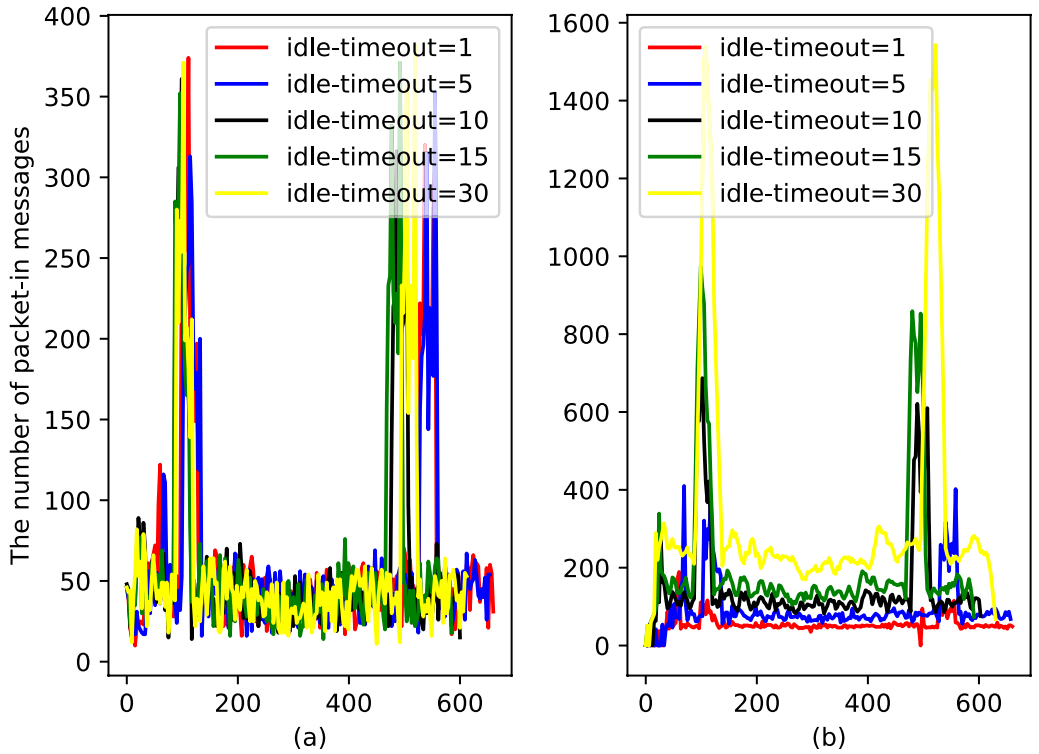

Fig. 5 The number of flow rules and packet-in messages with different idle-timeout values

Table 1 Attack types and attributes

Attack types	Attributes
SYN Flood DDoS	(generator: hping3, packets size: 200, attack rate: 1 ×)
SYN Flood DDoS	(generator: hping3, packets size: 400, attack rate: 1 ×)
UDP Flood DDoS	(generator: hping3, packets size: 1024, attack rate: 2 ×)
DNS Amplification DDoS	(generator: tcpreplay, attack rate: 3 ×)
SSDP Amplication DDoS	(generator: tcpreplay, attack rate: 3 ×)
ACK Flood DDoS	(generator: hping3, packets size: 200, attack rate: 1 ×)
ICMP Flood DDoS	(generator: hping3, packets size: 32, attack rate: 2 ×)

of the selected victim, the benchmark packet rate is 1000 pkts/s.

In our experiment, we refer to the work of Kalkan et al. (2018) and set the attack rate to 1-3 times that of regular traffic.

In order to obtain training and test data, we capture traffic both in the legitimate phase and attack phase. As a result, we collected 14795 samples in the attack phase and 12682 samples in the legitimate phase for the training dataset. In addition, we collect samples according to the attack types and attack rates for the testing dataset.

It is worth noting that when capturing training and testing samples, the time interval for the detection loop was set to 3 s, which is the same as the work of Braga et al. (2010). However, the detection loop will only work in typical conditions after a port reports an anomalous alarm. This design is to minimize access to flow rule statistics to decrease the burden of the controller.

Performance metrics

The performance of our detection algorithm could be evaluated through false-positive rate (*FPR*), accuracy (*ACC*), recall rate(*RR*), F1 score and time efficiency based metrics, computed using Eqs. 7, 8 and 9 respectively, where *TP* is the number of attack logs classified as attacks, and *TN* is the number of legitimate logs classified

as legitimate. Conversely, *FP* is the number of legitimate logs classified as attacks. Furthermore, *FN* is the number of attack logs classified as legitimate.

FPR is a critical indicator because it represents the proportion of legitimate traffic identified as malicious traffic and discarded:

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

ACC is a typical metric to determine whether the classification results are reliable:

$$ACC = \frac{(TP + TN)}{TP + TN + FP + FN} \tag{8}$$

RR aims to measure the probability that an attack is correctly identified. Since FORT proposes to monitor the state of ports selectively, it is essential to check if some attacks are missed:

$$RR = \frac{TP}{TP + FN} \tag{9}$$

F1 score, which is defined as the harmonic average of precision rate and recall rate, aims to measure the availability of the classification model:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{10}$$

Experiments results

This part describes the performance results according to the metrics defined above. In order to reach a convincing conclusion, we first elaborate a discussion on the results of different idle-timeout values, and the detection results for different attack types are taken out separately for comparison. Then we extend our analysis with different attack rates. Finally, we discussed the *FPR* and the necessity of distinguishing transmitted and received traffic.

Different idle-time values We configure the idle-timeout with different values, and a set of experiments are taken. First, typical DDoS types are chosen to verify the accuracy of our algorithm. Ye et al. (2018) also use SVM as their classifier, but they select different features. We reproduced their work and compared the results with FORT as follows.

Different from Ye et al. (2018), we detect DDoS from both the attacker side and victim side. Thus accuracy rates and F1 scores are also acquired from both sides. From the results in Table 2, we find that idle-timeout could affect detection effectiveness. Obviously, when idle-timeout equals 1 s, the detection accuracy and F1 score is worst in FORT and Ye's scheme. Through the analysis of MAWI data, we find that even the normal network flow includes many scanning packets or error packets. These packets will seriously affect the effectiveness of detection. If the idle-timeout is too small, the flow rule cannot reflect the traffic state for a long time. If the influence of scanning packets and error packets is superimposed, it will significantly interfere with the detection results. Also, for this reason, FORT performs better than Ye et al. (2018). Because we exclude the flow containing only one or two packets when extracting *RSA* and *RSR* features, we can more accurately reflect the asymmetry of traffic.

Different attack rates We generate typical DDoS with different attack rates, and the flow rule is defined with a fixed idle-timeout value. The detection results of these DDoS are shown in Table 3:

From the results in Table 3, we can find that the detection accuracy tends to be improved with the increase of attack rate, which is consistent with our intuition since attacks with higher rates usually lead to more apparent anomalies both on the attacker side and the victim side. It is worth noting that an SSDP amplification attack is not easy to detect when the attack rate is relatively low. After analyzing the CIC DDoS 2019 dataset, it is found that only one host launches the SSDP amplification

Table 2 Performance with different idle-timeout values

Idle-timeout	SYN Flood DDoS			UDP Flood DDoS			SSDP Amplification DDoS		
	1	5	10	1	5	10	1	5	10
ACC,F1 (FORT,Victim)	95.2%, 96.1%	97.26%, 97.17%	100%, 100%	97.64%, 97.23%	100%, 100%	100%, 100%	95.2%, 94.93%	97.9%, 97.83%	98.43%, 98.27%
ACC,F1 (FORT,Attacker)	95.31%, 95.12%	97.54%, 98.11%	98.72%, 98.65%	98.5%, 98.61%	100%, 100%	100%, 100%	94.76%, 95.12%	98.46%, 98.51%	98.75%, 98.53%
ACC,F1 (Ye et al. 2018)	93.42%, 94.12%	96.32%, 96.23%	98.72%, 97.65%	98.43%, 98.46%	100%, 100%	100%, 100%	91.54%, 93.23%	99.48%, 98.71%	96.23%, 96.11%

Table 3 Performance with different attack rates

Attack-Rate	SYN Flood DDoS			UDP Flood DDoS			SSDP Amplification DDoS		
	1×	2×	3×	1×	2×	3×	1×	2×	3×
ACC,F1 (FORT,Victim)	97.2%, 97.31%	97.26%, 97.17%	97.83%, 97.95%	98.75%, 98.23%	100%, 100%	100%, 100%	88.43%, 90.76%	97.9%, 97.83%	98.63%, 98.86%
ACC,F1 (FORT,Attacker)	97.2%, 96.89%	97.54%, 98.11%	98.41%, 98.27%	98.65%, 98.73%	100%, 100%	100%, 100%	87.16%, 90.58%	98.46%, 98.51%	98.21%, 98.11%
ACC,F1 (Ye et al. 2018)	95.26%, 95.23%	96.32%, 96.23%	96.4%, 96.78%	98.13%, 97.98%	100%, 100%	100%, 100%	76.03%, 80.18%	99.48%, 98.71%	97.53%, 97.41%

Idle-timeout = 5

attack, and the strategy of source IP randomization has not been applied. Moreover, compared with the attack generated by hping3, the flow in CIC DDoS 2019 usually contains multiple packets. Therefore, the interactive features such as *RSA*, *RSR*, and the pair-flow ratio in Ye et al. (2018) are no longer representative when the attack rate is low. As the attack rate increases, the detection accuracy and F1 score improves significantly. In most situations, FORT performs better than Ye et al. (2018) for two reasons. First, the feature used by FORT to reflect asymmetry is more precise since we exclude the flow containing only one or two packets when computing *RSA* and *RSR*; Second, we separate the transmitted and received rules so that we can get indicators *RTF* and *RTP*. With the increase of attack traffic, *RTF* and *RTP* will increase abnormally. This is an obvious sign of DDoS.

False positive rate (FPR) The false-positive rate is a crucial evaluation criterion for DDoS detection since a high *FPR* may trigger a mistake mitigation process, which influences the regular interaction between servers and clients. To acquire the *FPR*, another part of the legitimate dataset is replayed, and we only collect the legitimate samples. To save the resource of SDN, the idle-timeout is set as 5 s. The corresponding *FPR* are 0.23% and 0.17% respectively from the victim and attacker sides' classifiers. Moreover, it is surprising that if we comprehensively consider the results of classifiers for the victim side and attacker side, that is, when the attacker and victim determine the exception at the same time, we think an attack occurs, then the *FPR* is decreased to only 0.039%, which is a negligible probability. In comparison, the *FPR* of Ye's method is 1.24%. Through the analysis of features, it is found that all features related to proportion are usually stable, while indicators related to quantity sometimes vary greatly. For example, the speed of source IP and flow rules selected by Ye et al. (2018). Therefore, the occurrence of flash events increases its false-positive rate.

Recall rate (RR) The recall rate is crucial since FORT proposes to monitor the state of ports selectively. The recall rate in the ARIMA procedure matters because FORT relies on monitoring the state of ports to decrease the load. A high *RR* means that most of the attacks are identified.

To acquire the *RR*, we launched DDoS attacks periodically and monitored the port at the victim side in real-time. Specifically, each attack lasts for 5 s, and a new round of attacks will be launched 10 s after the end of the attack. The attack traffic is twice the regular traffic. Finally, we obtained 3000 samples. The normalized error of the actual value and the predicted value is highly correlated with *RR* as depicted in , so we defined multiple thresholds. The results of *RR* are shown in Table 4.

As we think, the recall rate is highly correlated with the threshold. Therefore, in order to improve the recall rate, we should set a relatively loose value. Specifically, $\Delta T_{t,i}$ and $\Delta \lambda_{t,i}$ should be relatively small, but this will lead to a high false-positive rate. Generally, a comprehensive trade-off is needed to select the threshold.

Table 4 Recall Rate with different thresholds

$\Delta R_{t,i}$	1	1	1	1	1	1
$\Delta T_{t,i}$	1	1.2	1.4	1.6	1.8	2
$\Delta \lambda_{t,i}$	0.2	0.4	0.6	0.8	1.0	1.1
RR	1	1	1	0.93	0.83	0.76

Table 5 Time efficiency with different attack rates

Attack-Rate	SYN Flood DDoS			UDP Flood DDoS			SSDP Amplification DDoS		
	1×	2×	3×	1×	2×	3×	1×	2×	3×
FORT	3.85 s	4.13 s	4.32 s	3.92 s	4.23 s	4.28 s	3.26 s	3.94 s	4.32 s
(Ye et al. 2018)	3.54 s	4.04 s	4.21 s	3.56 s	4.13 s	4.31 s	3.07 s	3.67 s	4.19 s

Idle-timeout = 5

The time efficiency of FORT Time efficiency is an essential concern in the existing methods based on flow rule statistics. Therefore, our method adopts an early warning through port monitoring and only requests flow rules related to the abnormal port to reduce the detection cost. It is found that the time cost from port statistics collection to ARIMA detection is only 0.7 s. While Ye et al. (2018) performs periodic SVM detection, it takes 1.5 s. Therefore, in the normal state, FORT can greatly reduce the time-consuming.

If DDoS attacks the network, further attack detection is required in addition to port monitoring. It is necessary to obtain the flow rules of the related switch; From the result in Table 5, if we launch a 2× SYN DDoS attack, FORT takes 4.13 s to finish the complete detection. Comparatively, the time cost of Ye et al. (2018) has also increased since the bandwidth is congested and the number of flow rules surges; thus, it averagely takes 4.04 s. From the results, the time-consuming difference between the two methods is not much. Considering that the time of the network in the normal state is much longer than that of being attacked, FORT is very efficient.

The necessity of distinguishing transmitted and received traffic In contrast to Ye et al. (2018) and Braga et al. (2010), we distinguish transmitted and received traffic and rules, which enables us to obtain features such as *RTF* and *RTP*. To prove the necessity of this distinguishing, we delete them and replace *RSA*, and *RSR* with the percentage of pair-flows defined in Ye et al. (2018) and Braga et al. (2010), which denotes the ratio of successful sessions. The idle-timeout is set as 5 s. We noticed the *ACC* for DNS amplification DDoS decreased to 47.25% and 47.83, and the *FPR* for benign samples decreased to 9.4% and 9.1%. The detection for DNS amplification DDoS failed almost completely. After analyzing the data set, it is found that only relatively few flows are generated in it. Each flow includes many packets; thus, it can also successfully lead to bandwidth flooding. There will be no significant increase in pair-flows since only a few attack flows are recorded in the related switch. If we do not distinguish transmitted and received rules, it will be challenging to fully acquire the anomaly of asymmetry.

Conclusion

In this paper, we studied how to use the function of SDN to detect DDoS. Our method proposed a flow rule-based detection. Periodical port monitoring is introduced to avoid the frequent request of flow rules. It observes the port statistics based on the ARIMA algorithm; the subsequent DDoS detection module works once indicators exceed specified thresholds. Since the states of the attacker and victim sides are different, our method can perceive attacks from the perspective of both. We demonstrated that the ARIMA is available in SDN scenarios and our detection methods are highly reliable through simulation tests. The detection of multiple attack types and untrained new attack types proves it. Further, we plan to distinguish attack traffic and legitimate traffic in the future to defend the DDoS and protect legitimate communication concurrently.

Acknowledgements
I would like to express my gratitude to all mates who support ideas and advice during the writing of this thesis and thank the anonymous reviewers for their valuable comments.

Author contributions
KJ proposed the lightweight scheme with careful experiments and wrote the manuscript. JW and JL joined the discussion of the work and provided suggestions on algorithms and data processing. CL, QL and FL reviewed the manuscript and gave suggestions on the revision of the details of the article. All authors read and approved the final manuscript.

Funding
This work was supported by the National Key R &D Program of China with No. 2018YFC0806900 , Beijing Municipal Science & Technology Commission with Project No. Z191100007119009, NSFC No. 61671448 and NSFC No.61902397.

Availability of data and materials
The data and materials of this study are all from the public dataset and they are available from the corresponding author upon reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]State Key Laboratory of Information Security, Institute of Information Engineering, Beijing, China. [2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China. [3]The Sixth Research Office, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China.

Received: 31 August 2021 Accepted: 4 July 2022
Published online: 03 October 2022

References

Balasubramanian V, Aloqaily M, Reisslein M (2021) An SDN architecture for time sensitive industrial IoT. Comput Netw 186:107739
Box GE, Jenkins GM, Reinsel GC, Ljung GM (2015) Time series analysis: forecasting and control. Wiley, pp 325–329
Braga R, Mota E, Passito A (2010) Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: IEEE local computer network conference. IEEE, pp 408–415
Chang C-W, Huang G, Lin B, Chuah C-N (2011) Leisure: a framework for load-balanced network-wide traffic measurement. In: 2011 ACM/IEEE seventh symposium on architectures for networking and communications systems. IEEE, pp 250–260
Cortez P, Rio M, Rocha M, Sousa P (2006) Internet traffic forecasting using neural networks. In: The 2006 IEEE international joint conference on neural network proceedings. IEEE, pp 2635–2642
Dao N-N, Park J, Park M, Cho S (2015) A feasible method to combat against DDoS attack in SDN network. In: 2015 international conference on information networking (ICOIN). IEEE, pp 309–311
Doshi R, Apthorpe N, Feamster N (2018) Machine learning DDoS detection for consumer internet of things devices. In: 2018 IEEE security and privacy workshops (SPW). IEEE, pp 29–35
Eliyan LF, Di Pietro R (2021) DoS and DDoS attacks in software defined networks: A survey of existing solutions and research challenges. Futur Gener Comput Syst 122:149–171
Fouladi RF, Ermiş O, Anarim E (2020) A DDoS attack detection and defense scheme using time-series analysis for SDN. J Inf Secur Appl 54:102587
Foundation ON (2021) OpenFlow specification. https://www.opennetworking.org (August)
Giotis K, Argyropoulos C, Androulidakis G, Kalogeras D, Maglaris V (2014) Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. Comput Netw 62:122–136
Group TMW (2021) MAWI Working Group Traffic Archive. http://mawi.wide.ad.jp/mawi/ (August)
Hu D, Hong P, Chen Y (2017) FADM: DDoS flooding attack detection and mitigation system in software-defined networking. In: GLOBECOM 2017-2017 IEEE global communications conference. IEEE, pp 1–7
Kalkan K, Altay L, Gür G, Alagöz F (2018) JESS: joint entropy-based DDoS defense scheme in SDN. IEEE J Sel Areas Commun 36(10):2358–2372
Leng X, Hou K, Chen Y, Bu K, Song L, Li Y (2019) A lightweight policy enforcement system for resource protection and management in the SDN-based cloud. Comput Netw 161:68–81
Mehr SY, Ramamurthy B (2019) An SVM based DDoS attack detection method for Ryu SDN controller. In: Proceedings of the 15th international conference on emerging networking experiments and technologies, pp 72–73
NSFOCUS Telecom C (2021) 2020 DDoS report. https://www.nsfocus.com.cn (August)
Perraudin N, Vandergheynst P (2017) Stationary signal processing on graphs. IEEE Trans Signal Process 65(13):3462–3477
Sanfilippo S (2021) hping3 tool. http://www.hping.org/hping3.html (August)
Sekar V, Reiter MK, Willinger W, Zhang H, Kompella RR, Andersen DG (2008) csamp: a system for network-wide flow monitoring
Sharafaldin I, Lashkari AH, Hakak S, Ghorbani AA (2019) Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In: 2019 international Carnahan conference on security technology (ICCST)
Singh MP, Bhandari A (2020) New-flow based DDoS attacks in SDN: taxonomy, rationales, and research challenges. Comput Commun 154:509–527
Ubale T, Jain AK (2020) Survey on DDoS attack techniques and solutions in software-defined network. Handbook of computer networks and cyber security. Springer, Cham, pp 389–419
Wang B, Zheng Y, Lou W, Hou YT (2015) DDoS attack protection in the era of cloud computing and software-defined networking. Comput Netw 81:308–319
Wang R, Jia Z, Ju L (2015) An entropy-based distributed DDoS detection mechanism in software-defined networking. In: 2015 IEEE Trustcom/BigDataSE/ISPA, vol 1. IEEE, pp 310–317
Wang H, Yang G, Chinprutthiwong P, Xu L, Zhang Y, Gu G (2018) Towards fine-grained network security forensics and diagnosis in the SDN era. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security, pp 3–16
Wu D, Nie X, Asmare E, Arkhipov DI, Qin Z, Li R, McCann JA, Li K (2018) Towards distributed SDN: mobility management and flow scheduling in software defined urban IoT. IEEE Trans Parallel Distrib Syst 31(6):1400–1418
Xu J, Wang L, Xu Z (2020) Survey on resource consumption attacks and defenses in software-defined networking. J Cyber Secur 5(4):72–95
Xu Y, Liu Y (2016) DDoS attack detection under SDN context. In: IEEE INFOCOM 2016-the 35th annual IEEE international conference on computer communications. IEEE, pp 1–9
Yan Q, Yu FR, Gong Q, Li J (2015) Software-defined networking (SDN) and distributed denial of service (DDOS) attacks in cloud computing environments: a survey, some research issues, and challenges. IEEE Commun Surv Tutor 18(1):602–622
Ye J, Cheng X, Zhu J, Feng L, Song L (2018) A DDoS attack detection method based on SVM in software defined network. Secur Commun Netw 2018:9804061
Yin D, Zhang L, Yang K (2018) A DDoS attack detection and mitigation with software-defined internet of things framework. IEEE Access 6:24694–24705
You X, Feng Y, Sakurai K (2017) Packet in message based DDoS attack detection in SDN network using OpenFlow. In: 2017 fifth international symposium on computing and networking (CANDAR). IEEE, pp 522–528
Zhang Y (2013) An adaptive flow counting method for anomaly detection in SDN. In: Proceedings of the ninth ACM conference on emerging networking experiments and technologies, pp 25–30
Zhao G, Xu H, Fan J, Huang L, Qiao C (2020) Achieving fine-grained flow management through hybrid rule placement in SDNS. IEEE Trans Parallel Distrib Syst 32(3):728–742

Publisher's Note