Machine learning based low-rate DDoS attack detection for SDN enabled IoT networks

Haosu Cheng

Key Laboratory of Aerospace Network Security,
Ministry of Industry and Information Technology,
School of Electronic and Information Engineering,
Beihang University,
Beijing, 100191, China
Email: icq73@qq.com

Jianwei Liu, Tongge Xu, Bohan Ren and Jian Mao*

Key Laboratory of Aerospace Network Security,
Ministry of Industry and Information Technology,
School of Cyber Science and Technology,
Beihang University,
Beijing, 100191, China
Email: liujianwei@buaa.edu.cn
Email: xutg@buaa.edu.cn
Email: 19971999rrr@sina.com
Email: maojian@buaa.edu.cn
*Corresponding author

Wei Zhang

Experimental Teaching Center,
ShanDong University of Finance and Economics,
Jinan, 250014, China
Email: 20053716@sdufe.edu.cn

Abstract: The software-defined network (SDN) enabled internet of things (IoT) architecture is deployed in many industrial systems. The ability of SDN to intelligently route traffic and use underutilised network resources, enables IoT networks to cope with data onslaught smoothly. SDN also eliminates bottlenecks and helps to process IoT data efficiently without placing a larger strain on the network. The SDN-based IoT network is vulnerable to DDoS attack in a sophisticated usage environment. The SDN-based IoT network behaviours are different from traditional networks, which makes the detection of low-traffic DDoS attacks more difficult. In this paper, we propose a learning-based detection approach that deploys learning algorithms and utilizes stateful and stateless features from Openflow packages to identify attack traffics in SDN control and data planes. Our prototype approach and experiment results show that our system identified the low-rate DDoS attack traffic accurately with relatively low system performance overheads.

Keywords: IoT; internet of things; software-defined networking; industrial system; low-rate distributed denial-of-service; machine learning.

Reference to this paper should be made as follows: Cheng, H., Liu, J., Xu, T., Ren, B., Mao, J. and Zhang, W. (2020) 'Machine learning based low-rate DDoS attack detection for SDN enabled IoT networks', *Int. J. Sensor Networks*, Vol. 34, No. 1, pp.56–69.

Biographical notes: Haosu Cheng received his BS in Information Technology from the Fontys University of Applied Sciences, Eindhoven, The Netherlands, in 2007, and the MSc in Software Engineering of Distributed Systems from the Royal Institute of Technology, Stockholm, Sweden, in 2010, respectively. He is currently pursuing the PhD with the Laboratory of Information and Network Security, School of Electronics and Information Engineering, Beihang University, Beijing, China. His current research interests include Internet of Things, the security of software-defined networking, and network and information security.

Jianwei Liu received his BS and MSc in Electronic and Information from Shandong University, Shandong, China, in 1985 and 1988, respectively, and the PhD in Communication and Electronic

System from Xidian University, Shanxi, China, in 1998. He is currently a Professor with School of Cyber Science and Technology, Beihang University, Beijing, China. His current research interests include wireless communication network, cryptography, and network and information security.

Tongge Xu received his BS and the MSc from Beihang University, Beijing, Chian. He is an Associate Professor with the School of Cyber Science and Technology, Beihang University, Beijing, China. His current research interests include network management and traffic/protocol analysis technology, UNIX/Linux system development, large information system design and development technology, public opinion big data analysis and mining.

Bohan Ren received his BS in Information Security from Nanjing University of Posts and Telecommunications, Jiangsu, China, in 2019. He is pursuing the MSc with School of Cyber Science and Technology Beihang University, Beijing, China. His current research interests include machine learning, cryptography and convex optimisation.

Jian Mao received her BS and PhD from Xidian University, Shanxi, China. She is an Associate Professor with the School of Cyber Science and Technology, Beihang University, Beijing, China. Her research interests include cloud security, web security, and mobile security.

Wei Zhang received his MSc in Computer Science from the Tongji University, Shanghai, China,in 2011. He is currently a Research Assistant with the Experimental Teaching Center, Shandong University of Finance and Economics, China. His current research interests include cloud computing and computer networks.

1 Introduction

Internet of things (IoT) and software-defined network (SDN) are two emerging technologies. The IoT aims to connect objects over the Internet, and the SDN provides orchestration for network management by decoupling the control plane and the data plane. An example of a SDN enabled IoT system is a vehicular network (Cai et al., 2019). There are billions of connected objects (e.g., vehicles and road side units) in an IoT network, which makes control and management become a complicated task. The SDN provides flexibility and programmability in the IoT network without troubling underlying architecture of existing implementations.

The SDN enabled IoT network includes three domains, the device domain, the network domain and the application domain. In the device domain, an IoT device connects with an IoT gateway via the short-range wired and/or wireless personal area network (PAN) technologies such as WiFi, Bluetooth and ZigBee etc. (Schumny, 2004; Chang, 2017; Iii et al., 2016). The SDN control plane logically controls the network in a centralised way. The SDN data plane is employed to conduct decisions made by the control plane in both network and device domains. The application domain includes the SDN application plane, which implements control functions invoking software-based logic, the IoT applications and server/cloud infrastructures.

Although this layered network architecture provides more flexible configuration and capabilities than traditional networks, it increases the likelihood of an attack. Mainly, the SDN control and data plane are the most vulnerable parts to attacks (Dayal et al., 2016; Spooner and Zhu, 2016). The network could attack by a compromised switch/host via using this vulnerability. The study (Antikainen et al., 2014) shows that the malicious nodes launch a network attack via a compromised Openflow switch from the SDN data plane. Dhawan et al. (2015) discussed several attacks on the SDN controller plane including DDoS attack, side-channel attack and A fake topology attack. Many security challenges in SDN are carried out by recent studies (Kuerban et al., 2016; Spooner and Zhu, 2016; Ahmad et al., 2015). These security challenges include the DDoS attack, network blocking, side-channel attack, black hole attack, data modification, switch data leakage and other common attacks from the traditional networks. On the one hand, SDN supports centralised control that makes it is easier to detect and respond to the DDoS attacks. On the other hand, Data plane defects also provide a favourable environment and conditions for traditional DDoS attacks.

Recently, the DDoS attack becomes a big threat in SDN-based IoT networks (Cai and He, 2019). The SDN-based IoT network has many potential nodes that can launch a DDoS attack, and the traditional DDoS attack on the client-side will still cause harm to it (Antikainen et al., 2014; Dhawan et al., 2015). Besides, attacks will appear at multiple levels such as the high-rate DDoS attack in the control plane and low-rate DDoS attack in the data plane (Antikainen et al., 2014; Dhawan et al., 2015; Qiao et al., 2016; Cai et al., 2018). The attackers launch a high-rate DDoS attack on the SDN control layer by sending a large amount of useless data to consume the controller and the network resources. The controller that runs out of resources will cause the whole SDN network crash down (Dhawan et al., 2015). Nevertheless, high-rate DDoS attack on the controller has distinct traffic characteristics, so it is relatively easy to detect and prevent this kind of attack. The low-rate DDoS attack targets the data layer with a meagre attack traffic rate. It is easy to hide the attack traffic in the regular traffic and hard to be detected (Ping et al., 2016).

In this paper, we propose a machine learning-based low-rate DDoS attack detection system (MLDD), which is based on two groups of features extracting from the Openflow package. It can identify the low-rate DDoS attack traffic from heterogeneous IoT network in the SDN control plane

and data plane. The low-rate DDoS attack is difficult to identify in the SDN data layer, and traditional raw IP packet processing filtering may occupy many system resources. Besides, in the case of the DDoS attack, the SDN switches generate a large number of Openflow packets due to the mismatch of flow rules or the unreachable packets. To solve the above problems, we respectively adopt the stateless and stateful features of Openflow packets in the controller and switch to build the system model. We design and implement the experimental environment for the system model testing. It includes the low-rate DDoS attack module based on IoT devices, the physical and virtual heterogeneous SDN network, and the data flow capture and feature extraction model. Furthermore, we verify the prediction results from different machine learning methods and the distribution of different features in the raw data. We compare the classification results that use stateful features (with or without raw IP packet-based features) and stateless features respectively. We also compared the results with traditional IP packet classification solution for the DDoS attack in IoT networks.

Contributions. In summary, we make the following contributions in this paper:

- We introduce the MLDD system model for SDN IoT network, and it uses stateful and stateless features from Openflow package to identify attack traffic in the SDN control plane and data plane.
- We design and implement the experimental environment that included a low-rate DDoS attack module, the heterogeneous SDN-based IoT network, the Openflow package capture and feature extraction model.
- We discuss and compare the classification results of different machine learning algorithms of using different feature models and put forward improvement suggestions based on the distribution of feature models in the raw dataset.

Paper organisation. The rest of this paper is organised as follows: Section 2 discusses related work; Section 3 presents the background of the low-rate DDoS attack; Section 4 describes the MLDD system model; Section 5 evaluates our approach and analyses experimental results; Section 6 concludes the paper.

2 Related work

Detection of DDoS attack is crucial for public infrastructure network, especially in IoT applications (Cai and Zheng, 2018), energy-constrained networks (Gao et al., 2019), and 5G wireless networks (Huo et al., 2019). Researchers in academia and industry are working on detection strategies to cope with DDoS attacks in these scenarios. The recent work in this area can divide into two categories: threshold-value based detection methods and feature-based detection methods. The threshold-value based detection methods are based on detecting one or more traffic indicators, such as packet delay, traffic rate and maximum entropy. An attack may occur when those collected traffic metrics in real-time reach a predetermined threshold value. The period of a burst is used as a threshold value to detect low-rate DDoS attack typically (Sun et al., 2004). The feature-based detection methods apply a classifier to classify anomaly and regular traffic. Generally, such methods adopt statistical analysis or machine learning to construct a classification model that contains the characteristics of attack behaviour for attack detection. For example, Doshi et al. used machine learning methods to train classifiers to detect DDoS attacks in IoT networks based on the behavioural characteristics of IoT data flow (Doshi et al., 2018).

Zhou et al. (2017) proposed an expectation of packet size-based approach to distinguish the low-rate DDoS attacks from the legitimate traffic by measuring the distribution difference of the packet size. Hoque et al. (2016) introduced the FFSc a statistical measure for multivariate data analysis to detect DDoS attack from normal traffic. It uses the entropy of source IPs, variation of source IPs and packet rate, three features of network traffic to analyse the behaviour of network for detection. Chistokhodova and Sidorov (2018) analysed several types of DDoS attacks and detecting methods with their modifications, such as low-rate HTTP-flood, SYN-flood and TCP-flood etc., and proposed to use a classifier with memory, as well as additional information to improve the existing detection methods. Chen et al. (2019) introduced a DDoS attack detection method based on the network abnormal behaviour in a big data environment. It defines the network abnormal feature value NAFV to reflect the IP changes, and identify the abnormal network flow caused by attacks.

Jang et al. (2014) proposed an approach to classify traffic attacks by visualising IP, Ports and their variance. The method uses an artificial neural network to classify normal and attack data. Toklu and Simsek (2018) proposed a two-layer filtering method that is able to detect high and low-rate DDoS attacks at the same time. The detection method uses the discrete Fourier transform (DFT) to analyse package arrival time to victim router. Xin et al. (2017) proposed an interest flooding attacks (IFA) detection scheme based on cumulative entropy in NDN. Cao et al. (2017) proposed MPTCP-La/E^2, a solution to detect DDoS attacks in a multipath TCP mobile cloud system. Behal and Kumar (2017) introduced a method to detect DDoS attack, which applies the information theory-based generalised entropy (GE) and generalised information distance (GID) metrics for DDoS attacks detecting. Ping et al. (2016) presented a DDoS attack detection method, which based on classifying the flow events associated with an interface and using the sequential probability ratio test (SPRT) to make a decision. Mehmood et al. (2018) proposed a naive Bayesian classification algorithm applied to DDoS attack intrusion detection in IoT networks. Mahjabin et al. (2017) provided a systematic introduction of DDoS attack, including different attacking types, some protection techniques and mitigation techniques on traditional systems. Mahjabin et al. (2019) presented two mitigation methods allowing a quick escape route of the normal traffic from the

Table 1 Summary of representative works

Work	*Target scenario*	*Core method*	*Technical features*	*Applicability and effect*
Our work	DDoS attack in SDN-based IoT networks	Learning-based DDoS attack Detection	Supports switch and controller node deployment in real-time detection, multiple features learning based on Openflow protocol with low system resource overhead.	Suitable for high or low-rate DDoS attack detection in SDN, high detection accuracy (up to 1)
Sun et al. (2004)	DDoS attack in TCP/IP-based traditional network	Dynamic time warping method based DDoS attack detection	Supports router node deployment, supports the defence method to isolate legitimate traffic	Suitable for low-rate DDoS attack detection in traditional TCP/IP network, defence mechanism can improve link capacity after attack (68% in Single TCP flow, 85% in Multiple flows)
Doshi et al. (2018)	DDoS attack in TCP/IP-based IoT networks	Learning-based DDoS attack detection	Supports route node deployment and real-time detection, multiple features learning based on TCP protocol	Suitable for DDoS attack detection in IoT network, high detection accuracy (up to 0.999)
Zhou et al. (2017)	DDoS attack in TCP/IP-based traditional networks	Packet-size distribution based DDoS attack detection	Supports independent of network topology, arrival patterns, and pulse patterns of attack packets	Suitable for high or low-rate DDoS attack detection in traditional TCP network, the detection accuracy up to 0.987
Hoque et al. (2016)	DDoS attack in TCP/IP-based traditional networks	Statistical measure based DDoS attack detection	Support threshold setting to improve detection accuracy and feature acquisition preprocessing	0.98 and 0.99 detection accuracy in CAIDA and DARPA dataset
Ping et al. (2016)	DDoS attack against SDN controllers	Sequential probability ratio test	Supports quick detection with a small number of flows supports multiple types of flood attack detection against SDN controllers.	The probability of detecting the attack data flow and compromised interface in the DARPA dataset is 0.33 and 0.6, respectively.
Mehmood et al. (2018)	DDoS attack in TCP/IP-based IoT networks	Naive Bayes classification algorithm	multi-agent-based IDS especially for the IoT environment, low implementation and execution cost	The detection probability is up to 0.78 in NS2.35 simulation system
Mahjabin et al. (2019)	DDoS attack in TCP/IP-based traditional networks	Distributed deployment and DNS caching	A light-weight load distributed method suitable for different types of organisations low implementation and execution cost	service can be accessed even when the DNS servers are down. packet drop rate is 0 in the simulation system.

DNS flood attack. Table 1 Summaries representative related works.

Unlike the work mentioned above, we are interested in exploring the low-rate DDoS attack detection method in the new SDN-based IoT network environment (Openflow package-based detection). Besides, the behaviour characteristics of traditional network services are also different from those of IoT and SDN. For example, IoT services have strong time rules and relatively fixed encapsulated data size, while SDN has the characteristics of controlling information and separating data forwarding. However, attackers can take advantage of these new features to launch low-rate DDoS attacks that hide in normal data flows and challenging to detect with traditional methods. For example, attackers use compromised cameras to launch attacks, and the attack stream hides behind the outgoing video streams. This paper adopts new features of SDN-based IoT network to extract two types of features according to the attack characteristics and improve the detection accuracy according to the correlation between the features. We focus on detecting network anomaly in a semi-physical simulation network environment. Our scheme can use machine learning ability to adapt to different attack variants, use network centralised control ability to provide global and local situation awareness ability, and use SDN centralised control ability to provide early warning service for low-rate DDoS attacks.

3 Low rate DDoS attacks

In this section, we discuss the related attacks and the new challenges involved in SDN enabled IoT networks.

3.1 DDoS

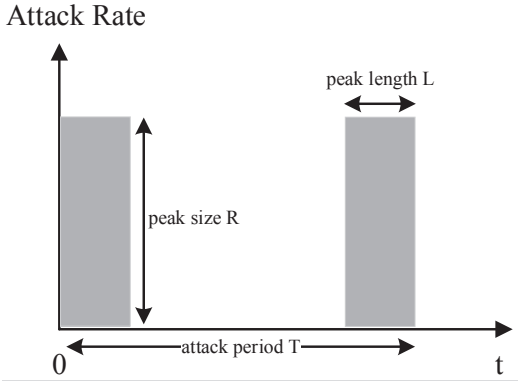
The distributed denial of service attacks the network services by sending large data streams from hundreds of thousands of infected hosts to a target host. Those DDoS attacks need a mass of resources to start a successful attack. It presents apparent statistical anomalies to network monitors under the attack. The DDoS defence strategies can detect and define the DDoS attack easily by using these obvious statistical anomalies. The low-rate DDoS attack can launch a successful attack by using a few attack resources (Kuzmanovic and Knightly, 2006). There are two types of low-rate DDoS attacks, shrew attack (Kuzmanovic and Knightly, 2006) and reduction of quality (RoQ) attack (Guirguis et al., 2004). We will mainly focus on the Shrew attack in this paper.

The Shrew attack is a DDoS attack on the transmission control protocol (TCP). It exploits a weakness in TCP's retransmission timeout (RTO) mechanism. It attacks legitimate TCP streams by periodically sending high rate packet flows in a low frequency that matches retransmission timeout. Each time the package senders recover from timeout, they will immediately encounter severe congestion on the bottleneck link and encounter timeout again. Therefore, legitimate data flows are extremely low or even nearly zero. IoT networks provide many potential attack nodes for DDoS attacks against specific hosts. The most typical of which is Mirai malware that uses numerous infected Web cameras for DDoS attacks (Kolias et al., 2017). IoT nodes send data packets with relatively fixed packet size and data transmission interval, and strong business continuity. IoT data packets generally encapsulate by TCP or UDP protocols. For example, Webcam's video data packets can range in size from tens to hundreds of kilobytes. The size of the attack packet can be kept very small and hidden in large data package streams. One of the attack features of DDoS is the ability to continuously send a large number of packets to the attacked host in a certain period. This pattern is highly similar to the way IoT terminals continuously feed service data to application servers, which makes it is challenging to detect DDoS attacks.

SDN has the characteristics of separation of forwarding and control. Packets sent by IoT nodes must match the flow table rules in the SDN switch so as to be forwarded correctly. Otherwise, mismatched packets will be re-packaged into Package-In messages through Openflow protocol and sent to the SDN controller for further processing. Besides, even the existing flow rules in the switch will be removed from time to time due to the limitation of hardware resources. It means that part of the attack data stream will always be re-encapsulated and sent to the SDN controller. Moreover, encapsulated or un-encapsulated DDoS attack data are always in the switch. It makes it is possible to obtain or detect DDoS attack data streams in both SDN controllers and switches. Various application protocols in the SDN-based IoT network are all based on TCP/UDP protocols, such as HTTP/HTTPS, Openflow, MQTT protocols, etc. It makes low-rate DDoS attacks that take advantage of the weakness of TCP protocol widely exist in SDN based networks. For instance, Antikainen et al. (2014), Dhawan et al. (2015) and Mohammadi et al. (2019) show how TCP SYC flooding attacks the SDN network. The Shrew attack model likes a square-wave in a time sequence with three parameters attack period T, peak size R and peak length L, which are indicated in the Figure 1.

The Shrew attack effect is not only defined by three attacking parameters $< T, L, R >$ but also influenced by network setting parameters. It includes the bottleneck buffer size B, the bottleneck bandwidth C, the number of TCP flows N and the delay of the TCP flow d. A successful Shrew attack requires that R is high enough to fill the bottleneck buffer and induce packet loss, L is long enough to invoke timeout, T is large enough to keep the average attack rate low. Three parameters are all tunable, and the peak size R should always be equal to the bottleneck bandwidth C. The victim would have time to react when $R < C$, and the attack resources would be wasted or be much easier to expose when $R > C$. For example, attackers must set the peak length L to a predefined value, which must be greater than or equal to the conditional minimum of burst length L_T^* for a successful attack (Luo et al., 2014). These attacking and network setting parameters will guide us to configure a successful Shrew DoS attack.

Figure 1 Square-wave of Shrew attack model



Typically, low-rate DDoS attacks can only be detected when the target object is being attacked in traditional networks. The attacking data flow goes through those traditional network nodes, which cannot provide early warning service to attacked targets. The network configuration parameters (e.g., $< B, C, N, d >$) are difficult to change for defending when under the attack. These are the results of lacking centralised control mechanism and inflexible architectures and also are the weaknesses exposed by the traditional network when facing low-rate DDoS attacks.

The SDN switch node can deploy a perceptive module to support a defence mechanism against DDoS attack flow in the forwarding path. The network configuration can be changed through the centralised control mechanism, such as network bandwidth, forwarding path and network delay to reduce or avoid the impact of DDoS attacks. The DDoS attack detection capability of the sensing module is the core function of the whole defence mechanism, and it drives us to explore a method suitable for detecting DDoS attack in SDN-based IoT environment.

4 MLDD: design

4.1 DDoS attack threat in the SDN data plane

As we discussed previously, the security threats exist in SDN-based IoT network, and the data plane is the most vulnerable to DDoS attacks, especially low-rate DDoS attacks (Spooner and Zhu, 2016). There are quite destructive to application services in the targets and can hide themselves in the normal data flow and difficult to detect (Antikainen et al., 2014; Toklu and Simsek, 2018). The data plane holds the SDN forwarding devices such as routers, switches, IoT-gateways and access points, and manages configurable device entity using Openflow protocol in control plane.

The SDN device can be (re)configured for different purposes, including traffic isolation, data-path modification and device virtualisation. The forwarding rules master the data forwarding path, and they store in the flow table of the switch. The controller is responsible for receiving data forwarding requests and issuing flow rules to switches. The SDN applications are responsible for initiating configuration instructions of the data layer via invoking embedded functions in the controller. The IoT gateways connect IoT devices with different protocols (WiFi, Bluetooth, ZigBee, etc.) and forward the collected data to other network devices such as switches and routers. Some IoT devices can transmit data directly to other networking devices such as WiFi-based cameras and sensors by connecting to a WiFi access point without an IoT gateway.

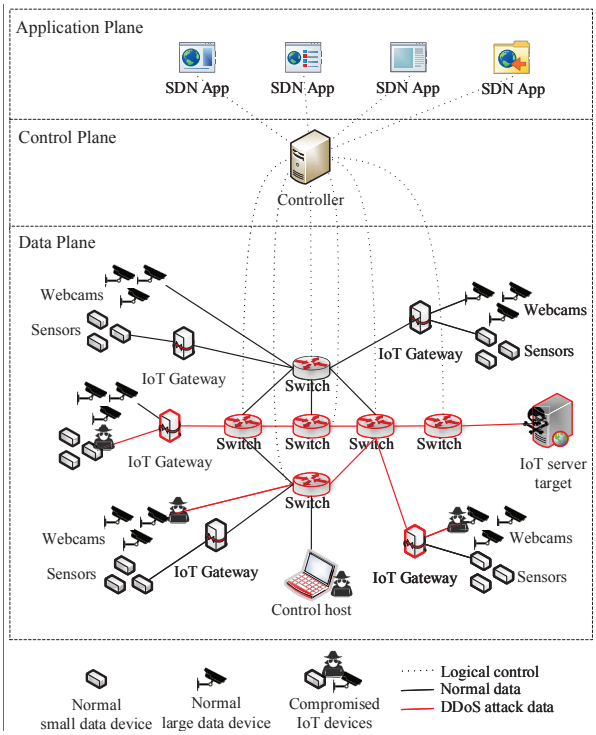
IoT devices can produce two different types of data: small data packets generated by conventional sensors and big data packets generated by things like cameras. These two types of data packets appear as regular data streams in the SDN data plane. Hackers can use these two types of devices to configure the different DDoS attack streams. For example, they can use webcams to launch high-rate DDoS attacks, use IoT sensors to launch low-rate DDoS attacks or use a combination of both to launch a mixed attack.

After a portion of the IoT device has been taken over by a hacker, the hacker can launch DDoS attack commands on the selected targets from any compromised nodes. The compromised device sends DDoS attack packets to the target using existing flow rules in the forwarding device of the SDN data plane. Alternatively, if the relevant flow rules do not exist or flow rules expire in the forwarding device. The forwarding device will send the flow rules requests with original attacking data to the controller for further processing. Therefore, the attack packets have to pass through the corresponding forwarding devices, or some of them appear in both switches and controller as the repackaged Openflow packages.

Generally, low-rate DDoS attack parameters such as the package size and sending interval time are pre-configured by the attacker before launching an attack (Luo et al., 2014). The pre-configured attack parameters ensure that the characteristics of the low-rate DDoS attack stream are not distinct compared to the regular data stream. It can hide in the regular data stream to achieve the goal that the attacking data stream is difficult to detect. When the low-rate DDoS attack packet reaches the target, it uses the RTO mechanism to generate more useless packets. They will quickly consume the processor time, memory space, network interface buffer and other resources of the target host. Legitimate data packets cannot be correctly received and processed due to severe resource consumption.

We consider a heterogeneous IoT network setup with traffic across Openflow and non-Openflow entities. We assume a trusted controller that ensures the correct network functions, but we do not trust either the switches or IoT nodes. It implies that switch can cheat about everything to the controller, as the switches connect with controller over separate TCP (or TLS enabled) channel. Nevertheless, we do assume that most of the switches are trustful in the network. The compromised IoT devices can launch a DDoS attack via receiving an attack command from a latent control host. We assume the IoT server is the potential targets of strategic value in the network as Figure 2 shown. The IoT server receives both legitimate data streams from regular devices and attacking data streams from compromised devices. The compromised device could be conventional sensors or webcams, and they connect with or without an IoT gateway on the data plane.

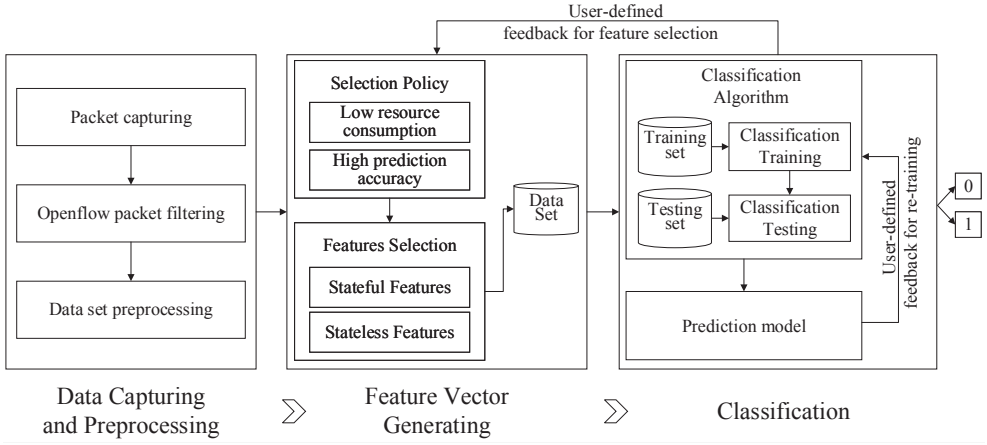
Figure 2 DDoS attack in the SDN data plane (see online version for colours)



Finally, we trust the SDN applications that run as modules of the controller binary, and they can be trusted as long as the controller itself is trusted. The application plane is above the control layer and logically connects to it through REST-APIs such as the HTTP GET/PUT/POST/DELETE methods. It interacts with the control plane through the data plane and the TCP protocol. The application plane sends service requests to the control plane through REST-APIs. The control layer sends relevant application services to the corresponding forwarding devices through the form of flow rules to complete the interaction. Mostly, the interactions between the application plane and control plane, the control plane and data plane all go through the data plane, so our threat model focuses on the data plane.

The assumptions above imply that Openflow communication from controller to switches is trusted, but Openflow communication from switches to switches is untrusted. The communication between switches and IoT devices is untrusted, and the switch or IoT devices can transmit illegal data packets in some cases, such as during the DDoS attack. Moreover, various IoT and forwarding devices in the data plane are natural targets and attacking platforms for DDoS attacks in SDN network.

Figure 3 IoT DDoS attack detection approach



4.2 Approach review

The DDoS Detection framework can be deployed in the critical node of the SDN network such as switch or controller side. It is a learning-based system, consisting of three phases, Data capture and preprocessing, Feature vectors generating and Classification. In the first phase, it captures the SDN network flow data from the network interface, filters the Openflow stream data from captured network raw data and normalises the dataset for the next phase. In the second phase, two groups of vectors (stateless and stateful features) are generated based on the preprocessed dataset. The generated feature vectors dataset is divided into the training set and the testing set. The last part uses the different binary classification algorithms such as random forests, supports vector machines, and k-nearest neighbour with the training and testing data set to train the predicting model for DDoS attack detection. Figure 3 shows the detection approach described above.

4.2.1 Data capturing and preprocessing

The packet capturing is the first step in the framework for IoT DDoS detection processing. It captures the data packages of the IoT network via physical or virtual network interfaces at SDN switch and controller devices. The data package contains the data information from different protocol layers, such as application layer with Openflow protocol, transport layer with TCP/UDP protocol, network layer with IP protocol. Our primary concern is the application layer with Openflow protocol at SDN switch and controller side.

After the packet capturing step, we filter out the required packets (e.g., Openflow or TCP) from a large number of captured packets. The filtered packet can be grouped by switch device and time window. Packet form different switches separate by source and destination IP, MAC address and port. Packets are further divided into non-overlapping time windows (e.g., 10 s) by timestamps recorded at capturing devices such as SDN switch or controller.

4.2.2 Feature vector generating

The feature vectors are generated based on the previous step. The feature values can be divided into two groups, stateless and stateful feature. The stateless feature is only related to the properties of the individual packet and has nothing to do with the properties of the data stream, while the stateful feature is related not only to the properties of the individual packet but also to the properties of the data stream. Both group features are based on domain knowledge of IoT device behaviour in SDN network. The stateless features are extracted from package header fields directly, such as arriving time, IP addresses, ports, package type, and so on. They consume very few hardware resources to process in extraction. The stateful features are aggregated packet data streams information during a short period, such as the network bandwidth, the change in the number of packages over time, the speed of change and the acceleration. They require more hardware resources to obtain and compute. The generated feature vectors also contain a tag item that is attacked or not. For instance, our feature vector model contains ten stateless features and eleven stateful features. Table 2 lists all features that are involved in our model. They are critical to detect the DDoS attack stream in the system.

Table 2 The features set

Stateless	$Time, IP_s, IP_d, PT_s, PT_d, Type, S_i, \Delta T, \frac{d\Delta T}{dt}, \frac{d^2\Delta T}{dt^2}$
Stateful	$\Delta N, \frac{d\Delta N}{dt}, \frac{d^2\Delta N}{dt^2}, N_{of}, S_{of}, bw, \frac{dbw}{dt}, \frac{d^2bw}{dt^2}, \Delta N_{of}, \frac{d\Delta N_{of}}{dt}, \frac{d^2\Delta N_{of}}{dt^2}$

Stateless features are obtained from flow-independent characteristics of individual Openflow packets. There are 10 Stateless features in our feature model, including package arrival time $Time$, source and destination IP address IP_s, IP_d, source and destination ports PT_s, PT_d, type of Openflow packages $Type$, the packet size S_i, the Openflow package interval, ΔT, $\frac{d\Delta T}{dt}$, $\frac{d^2\Delta T}{dt^2}$ (the magnitude and speed of the change in ΔT). $Time$, IP_s, IP_d PT_s and PT_d are the most commonly used packet features that can be used to identify the direction and destination of the packet's flow over time. They exist in both IP and Openflow package. Arrival time helps packets to be grouped according to time and get other time-based features. The IP address and port can help to identify the source and destination of the data stream.

Normally, the low-rate DDoS attack package size is tiny. They are typically smaller than 100 bytes, and the normal IP package size is between 100 and 1200 bytes in the raw IP package (Doshi et al., 2018). For example, a TCP SYN attack package size is 74 bytes, and it after being repackaged into Openflow Package-In message is 164 bytes in our experiment. It will result in a significant reduction in the accuracy of attack detection mainly based on the size of the attack packet. Figures 6(a) and 7(a) indicate the distribution of Openflow packets of different sizes in controller and switch. A DDoS attack, such as TCP SYN Flood, is trying to open as many connection requests as possible with the victim to exhaust the victim host's resources. Thus, an attacker wants to keep the package size as small as possible in order to maximise the number of connection requests per second. In other words, if the number of tiny packets increases in a given period, the probability of attack will increase.

Based on the forwarding mechanism of SDN, different Openflow packet types are distributed differently in normal flow and attack flow. Flow rule mismatch or flow table reset can result in a large number of package-in, package-out, and package-mod Openflow packets between the switch and the controller during the attack. The data flows between normal nodes are relatively stable without apparent burstiness. The data exchange between controller and switch is relatively stable under normal conditions. Figures 6(b) and 7(b) indicate the distribution of Openflow packets of different types in controller and switch.

The limited burstiness makes normal traffic stream stable. Most Openflow packets are sent within appreciable time, and the interval between them is relatively stable in the normal stream. This highly cyclical network activity is evident in IoT networks. In contrast, the time interval between packets in a DDoS attack stream is approximately zero and high first and second derivatives of packet intervals.

All stateless feature vectors are available from the current and next Openflow packages. These features do not need to split the incoming traffic stream by IP source to generate. Therefore, Stateless features are lightweight for processing.

The stateful features indicate how network traffic states change over time. Hence, we split the network traffic (Openflow package) by a time window, e.g., 10 seconds in our case. The time windows will serve as a time-series representation of the devices' changing network behaviour. There are 11 stateful features in our feature model, including the data bandwidth currently used by the device bw, $\frac{dbw}{dt}$, $\frac{d^2bw}{dt^2}$ (the magnitude and speed of the change in bw), the number of arrived package (all kinds of packages, including IP and Openflow packages) per second ΔN, $\frac{d\Delta N}{dt}$, $\frac{d^2\Delta N}{dt^2}$ (the magnitude and speed of the change in ΔN), the number of Openflow package changed in the time window ΔN_{of}, $\frac{d\Delta N_{of}}{dt}$, $\frac{d^2\Delta N_{of}}{dt^2}$ (the magnitude and speed of the change in ΔN_{of}), the number of Openflow package in the time window N_{of} and the size of Openflow package in the time window S_{of}. The bandwidth contains evidence that describes the current network usage of IoT devices. For example, the ordinary working IoT devices consume relatively stable network bandwidth, while DDoS data streams consume as much bandwidth as possible. The victim's available bandwidth will be decreased when attacking occurs, and $\frac{dbw}{dt}$ and $\frac{d^2bw}{dt^2}$ are high during the attack.

In some cases, IoT nodes are numerous, and they send many packages with big data periodically. For example, Webcams provide video data that take up much bandwidth, and the amount of bandwidth variation is not significant when attacked. Because the attack data can hide behind the legitimate big data when launching a low-rate DDoS attack. The magnitude and speed of the bandwidth change brought by the low-rate DDoS attack data are not apparent. The problem mentioned above can be solved by using the amount of arrived package at some time (e.g., ΔN and ΔN_{of}), as well as the magnitude and speed of the changes in the number of packages (e.g., $\frac{d\Delta N}{dt}$, $\frac{d^2\Delta N}{dt^2}$ and $\frac{d\Delta N_{of}}{dt}$, $\frac{d^2\Delta N_{of}}{dt^2}$). Although the bandwidth occupied by a low-rate DDoS attack stream can hide in legitimate stream with big data, the number of arrived independent attack packets cannot hide. Statistics of all kinds of data packets on the SDN switch or control node would be a very resource-intensive work. When the data packets arrive at the SDN switch, a large number of Openflow packets will be generated and interact with the controller due to mismatching flow rules, flow table update and other reasons. In general, if the flow rules exist, the switch will directly forward the data packets according to the flow rules without generating any extra Openflow data packets (except the heartbeat data packets sent regularly between the switch and the controller). However, the number of Openflow packets is much smaller than the number of raw IP packets that arrived. The system resources overheads of using Openflow-based statistics features such as ΔN_{of}, $\frac{d\Delta N_{of}}{dt}$ and $\frac{d^2\Delta N_{of}}{dt^2}$ are much lower than those of using raw IP-based statistics features directly.

The number of the arrived package per second, the number and the size of the Openflow package in the time window are all time-related features. They are used to describe the different behaviour patterns of the DDoS attack flow and the normal flow in different periods, especially the abnormal behaviour patterns between the switch and control. The stateful features require aggregating statistics over many packets in a time window. Those stateful features need extra system overhead to generate.

4.2.3 Classification

The learning-based classification and detection module have two working phases, which are the learning phase and detecting phase. Each of these work phases uses part of the dataset prepared in the previous step. There is no overlap between the two datasets, and the usual quantitative ratio is 80% for the training dataset and 20% for the testing dataset. The module uses the binary classification algorithms such as random forests, supports vector machines, and K-nearest neighbours, etc., to train and test through those two data sets. The module will work in the detection phase after the training. The detection model generated by the learning phase can directly participate in the classification and detection of DDoS attacks in detecting phase. The

module supports periodic learning and testing in a preset manner to continuously improve the accuracy of the detection and adapt to changes in the operating environment, such as network topology or network configuration parameters change.

5 Evaluation

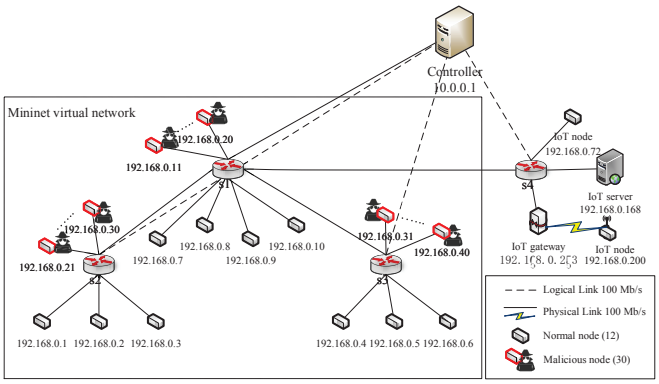
5.1 Experimental settings

The specific configuration and parameters of this experiment are introduced in this section, including network topology, IoT server, IoT node and DDoS attack configuration.

5.1.1 Network topology

The experimental network is an IoT heterogeneous network, and its topology is indicated in Figure 4. It contains three virtual switches, one physical switch, both virtual and physical switch connects the SDN controller that installs the Floodlight (Floodlight, n.d.) application and sets IP address to 10.0.0.1. Forty virtual nodes connect to switch S1, S2 and S3 respectively in total, ten nodes are normal (IP address from 192.168.0.1 to 192.168.0.10), and the rest of the nodes are attacking nodes (IP address from 192.168.0.11 to 192.168.0.40). One physical SDN switch (Raspberry Pi3) with four ports connects to the virtual switch S1, an IoT node, an IoT server and an IoT WiFi gateway (TP-Link WiFi-AP) respectively, while the wireless IoT node hardware is the Raspberry Pi ZeroW based WiFi device. The link bandwidth between the node and the switch is 100Mb/s, including both physical and virtual nodes and switches. Finally, the controller provides the SDN switch with the flow table strategy issuing and management mechanism, and it is deployed on the same PC with the Mininet virtual network at the same time. Table 3 indicates all physical devices configuration in details.

Figure 4 The experimental network topology (see online version for colours)



5.1.2 IoT server and node

The IoT Server (Broker) and nodes communicate with each other via the MQTT (Mqt, n.d.) protocol. The MQTT protocol is an application layer protocol built on top of the TCP protocol. It is a simple publish/subscribe and lightweight messaging protocol to communicate between IoT devices and server. The physical and virtual IoT Nodes send some fixed messages such as the current WiFi single strength, time, temperature, etc., to IoT server. We implement the MQTT communication mechanism, and the message content carried is in the Python code, as shown in the following code list.

```
com <- "mqtt message publication commmand"
host <- "ip + port + username + password"
topic <- "topic id"
node <- "node id"

while 0<1
 do t <- current time
 m <- "message+node+t"
 /*system executes command*/
 call execute ("com+host+topic+ m")
  /*sleeep for 2 seconds*/
 call sleep (2000)
end
```

The above code is deployed in physical and virtual hosts. It transmits around 114 Bytes message in every two or three seconds from each normal host. The physical host with IP address 192.168.0.168 is the IoT server, which installs Apache-Apollo (ActiveMQ for now) (Apollo, n.d.) application to work as an IoT server. The Apache-Apollo is configured based on the scale of our network. The maximum number of MQTT messages received at the same time is 100 (MQTT message buffer size). The messages receiving port is 61613, the web administration port is 80, and the rest of the settings remain default. Moreover, The TCP socket max buffer size of IoT Server is 6 Mbytes, the average delay from a virtual node to server is 0.226 ms.

5.1.3 DDoS attack configuration

In this section, we launch the DDoS attack from thirty virtual nodes to a physical IoT server host. The network topology and settings required for an attack are discussed in the previous section. According to the mathematical model of DDoS attack introduced in the previous chapter, we need to set attack parameters based on network configuration to ensure the successful implementation of the DDoS attack.

The network configuration parameters include $< B, C, N, d >$, i.e., a buffer size B = 6 Mbytes, a bottleneck link of capacity C = 100 mb/s, the number of TCP flows N = 30 and the one-way average delay of TCP flows d = 0.226 ms (it is the actual time and negligible), the minRTO is set to its default value 1 s, the TCP receiver would delay at most 10 ms to send ACK to TCP sender (D = 10 ms). For our attack, scenario parameter b will be set to 2, when b = 2 means an application only has data transmission in one direction (from IoT nodes to IoT server).

The burst length L will be set to $B/C + 3d$ (around 0.5 s), it should be larger than $L_{min} = B/C + 3d - Nb/C$ for the success of the attacks. For every L, we set the period T to 5 s

Table 3 Physical device configuration

Device	*IoT node*	*IoT node*	*IoT server*	*Controller*
CPU	Quad-core ARM Cortex-A53	Single-core ARM11	Quad-core ARM Cortex-A53	Quel core Intel I7-4770
RAM	1GB	512MB	1GB	8GB
OS	32Bit Raspbian April 2019	32Bit Raspbian-Lite April 2019	32Bit Ubuntu-mate 16.04	64Bit Ubuntu 16.04
Network	LAN 100Mb/s	WiFi 72Mb/s	LAN 100Mb/s	LAN 100Mb/s
IP address	192.168.0.72	192.168.0.200	192.168.0.168	10.0.0.1
Software	Python with MQTT	Python with MQTT	Apache-Apollo with MQTT	Floodlight and Mininet

Table 4 The classification accuracy results

	Controller				*Switch*			
Accuracy results	*SVM*	*NB*	*KNN*	RF	*SVM*	*NB*	*KNN*	RF
Stateless only	0.972	0.793	0.972	0.974	0.904	0.659	0.894	0.908
Stateless&Stateful	0.999	0.798	0.999	1.000	0.999	0.659	1.000	1.000
Stateless&OF Stateful	0.999	0.790	0.999	0.999	0.999	0.658	0.999	0.999

to make sure T is larger than T_L^*, which means the conditional minimum of DoS period for the given L. The conditional minimum of W_k (as $W_L^* = B + 2dC + N/b$) and T_L^* can be resolved from Formula in Luo et al. (2014). Formula 1 defines the lowest average attack rate A_{min}. It ensures the attacking configuration from the malicious nodes to consume the minimum system resources. The maximum burst period T_{max} has no upper bound, but it does not mean longer burst period getting better attack effectiveness. In other words, a shrew attack will not succeed if $L < L_{min}$, regardless of its period T.

$$A_{min} = \frac{R \cdot L_{min}}{T_{max}} \tag{1}$$

Each malicious node uses the Linux Hping program to launch an attack on the IoT server, and network time to ensure that the attack is synchronous, as shown in the following python codes.

5.2 Experimental results

We tested four machine learning algorithms to distinguish normal and DoS attack traffic from Openflow packages in SDN Controller and Switch, such as the support vector machine with RBF kernel (SVM), the multinomial naive Bayes algorithm (NB), the k-nearest neighbours algorithm (KNN) and the random forest using Gini impurity scores (RF). We implemented these machine learning algorithms using the Python Scikit-learn library (Sklearn, n.d.) with the default values for all hyper-parameters.

Our dataset is generated at real time in our experimental network environment. The dataset collected on the controller and switch contains 204,888 and 48,509 records of data respectively. We train the classifier on a training set with 80% of combined normal and attack traffic dataset, and tested classification accuracy on a test set of remaining dataset. The normal traffic contains MQTT, HTTP, HTTPs, Ping messages etc., and the attack traffic contains a large number of TCP-syn and TCP-retransmission messages in addition to those contained in the normal stream.

Table 4 shows the classifier performance with and without specific stateful features by using classification accuracy indicator. The accuracies of our classifiers ranged from 0.798 to 1.000 in the controller and from 0.659 to 1.000 in the switch. Note that here in the controller almost four times as many Openflow packages as there are in the switch during the same attack. Thus, the testing results are better in the controller than in the switch, especially for using stateless feature dataset.

```
T <- 10 seconds
L <- "u1"
R <- "10000"

def synFlood(tIP, tPort) do
 /*tIP:target IP, tPort:target Port*/
 x <- randomFromTo(1, 254)
 y <- randomFromTo(41, 253)
 srcIP <- 192.168.x.y

 cmd <- "hping3 + -a + srcIP + -S + tIP
   +-p + tPort + -i + L + -c +R"
 call execute (cmd)
end

def doFlood() do
 for i<-0 to 8 setp 1 do /*>0.5s*/
  call synFlood(192.168.0.168,61613)
end

def run() do
 toRun <- False
  while 0<1
   do time <- current_time_second().
   if time = 0
    then toRun <- True
  break
  /*every time T run call doFlood */
   call schedule(T, doFlood)
  end
  while toRun = True
   do call pending()
   /*call schedule pending*/
  end
end

call run()
```

As mentioned earlier, stateful features cost more system resources to collect, especially those stateful features such

Table 5 The detailed results with stateless features

	Controller				*Switch*			
	SVM	*NB*	*KNN*	RF	*SVM*	*NB*	*KNN*	*RF*
Accuracy	0.97	0.79	0.97	0.97	0.90	0.66	0.89	0.91
Precision	0.96	0.72	0.96	0.97	0.94	0.92	0.93	0.95
Recall	0.97	0.84	0.97	0.97	0.95	0.67	0.95	0.94
F1-score	0.97	0.77	0.97	0.97	0.94	0.77	0.94	0.94

as bw and ΔN based on the raw IP packet. Using stateful features based only on Openflow provides the right balance between resource consumption and prediction accuracy. In both the controller and the switch, the prediction results using stateless features and only Openflow-based stateful features are very close to the results using all the features as shown in Table 4. The stateful features based only on Openflow are ΔN_{of}, $\frac{d\Delta N_{of}}{dt}$, $\frac{d^2\Delta N_{of}}{dt^2}$, N_{of} and S_{of}.

Moreover, the stateful feature dataset is often used with better results than the stateless feature dataset, but the NB algorithm is an exception. Although the NB algorithm is still valid when using a small dataset, it is more sensitive to input data preparation methods. The type of our data model is numeric, and the data type suitable for the NB algorithm is nominal, such as text. Our data model does not suit for the NB algorithm.

The SVM, KNN and RF algorithm achieve similar results in the controller and switch respectively. The ROC results of Figures 5(a) and (b) also show the same results.

The stateful datasets can provide better prediction accuracy, and they also cost more system resources for data collecting. Compared to the stateful dataset, the stateless dataset is more attractive to us. It is especially suitable for use on limited resource devices such as switches because it costs less system resources for data collection with acceptable prediction accuracy. Table 5 indicates the detailed results of the stateless features only. From the experimental results in the table, we can see that except for the NB algorithm, the accuracy, recall and f-score of other algorithms are all 0.97 in the controller. The precision results of the SVM and KNN algorithms are both 0.96, and the precision result of the RF algorithm is 0.97. Also, the NB algorithm has the worst effect in the switch, while the RF algorithm has the best effect in the switch. The accuracy, precision, recall and f1-score of RF algorithm are 0.91, 0.95, 0.94 and 0.94, respectively. SVM algorithm and KNN algorithm have the best recall rate of 0.95.

The RF algorithm not only gets the best grades results in the controller and switch with all features but also achieves excellent results with only the stateless features, suggesting that stateless features are also sufficient to identify both normal and attack traffic. Table 6 shows the importance of each stateless features in the RF algorithm by using the Gini scores (value between 1 and 0). Some features are more important than others for both the controller and switch, such as the Openflow package arriving time, type and size.

The distribution of Openflow package type and size varies significantly under normal and attack conditions, and it is indicated in Figures 6 and 7. Figure 6(a) shows the Openflow package size between 100 and 200 bytes is 48.2%, the package size between 200 and 300 bytes is 45% in total under the attack traffic in the controller. Figure 6(a) indicates the package size between 100 and 200 bytes increases to 86% and the package size between 200 and 300 bytes decreases to 6.3% in the normal traffic from the controller.

Table 6 Stateless feature importance using Gini impurity scores

Features	*Controller Gini score*	*Switch Gini score*
$Time$	0.30	0.66
IP_s	0.05	0.0001
IP_d	0.05	0.04
PT_s	0.29	0.009
PT_d	0.22	0.01
$Type$	0.04	0.09
S_i	0.03	0.18
ΔT	0.0002	0.0005
$\frac{d\Delta T}{dt}$	0.01	0.0009
$\frac{d^2\Delta T}{dt^2}$	0.002	0.002

It also happens on the switch, Figure 7(a) shows the Openflow package size between 100 and 200 bytes is 47.5%, the package size between 200 and 300 bytes is 50.5% in total under the attack traffic. Figure 7(a) indicates the package size less than between 100 and 200 bytes increases to 58.3% and the package size between 200 and 300 bytes decreases to 25.5% in the normal traffic. The attack traffic generates a large number of OF_Package_IN, OF_Package_Out, OF_Flow_Mod and Error messages due to the forged IPs and random ports attached to the attack packet. Moreover, The Openflow package with ARP message is around 152 bytes, and an attacking message is around 170 bytes. The Openflow package with the normal MQTT message is around 230 bytes, and the Openflow error message is round 228 bytes. The large number of 200 to 300 bytes packages in the attack stream exists because of the large number of error messages.

As we mentioned above, the most attractive types of Openflow messages are OF_Package_In, OF_Package_Out and OF_Flow_Mod. The packet type distribution under the attack and normal traffic in the controller and switch are indicated by Figure 6(b) (in the Controller) and Figure 7(b) (in the Switch). In the attack flow of the Controller, the proportion of Openflow package types change significantly compares to the normal flow, among which the OF_Flow_Mod messages increase from 10.5% to 38.9%, the OF_Package_Out messages increase from 10.1% to 31.2%, and the OF_Package_In, messages decrease from 78.3% to

Figure 5 Classifier ROC graph (see online version for colours)

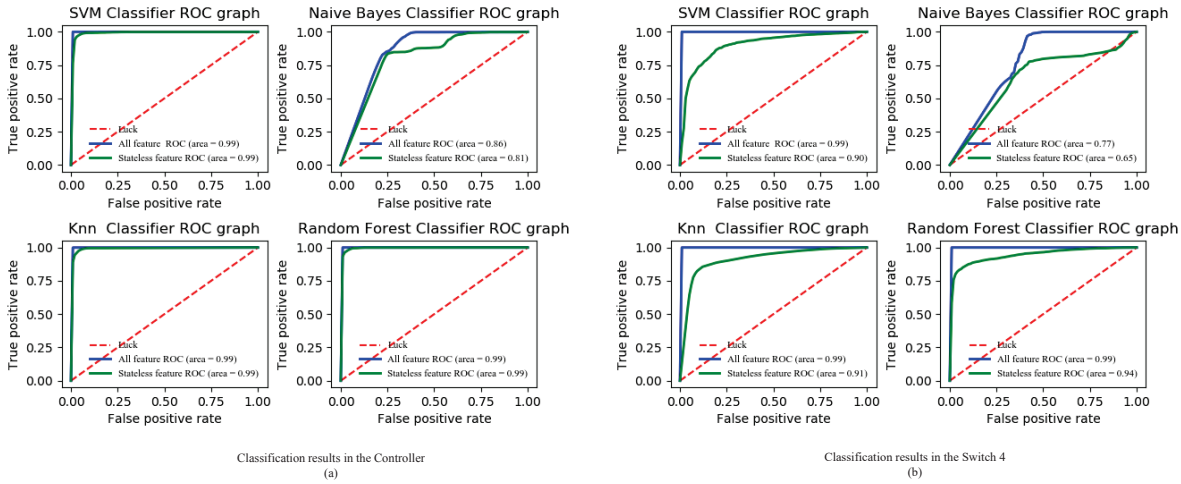
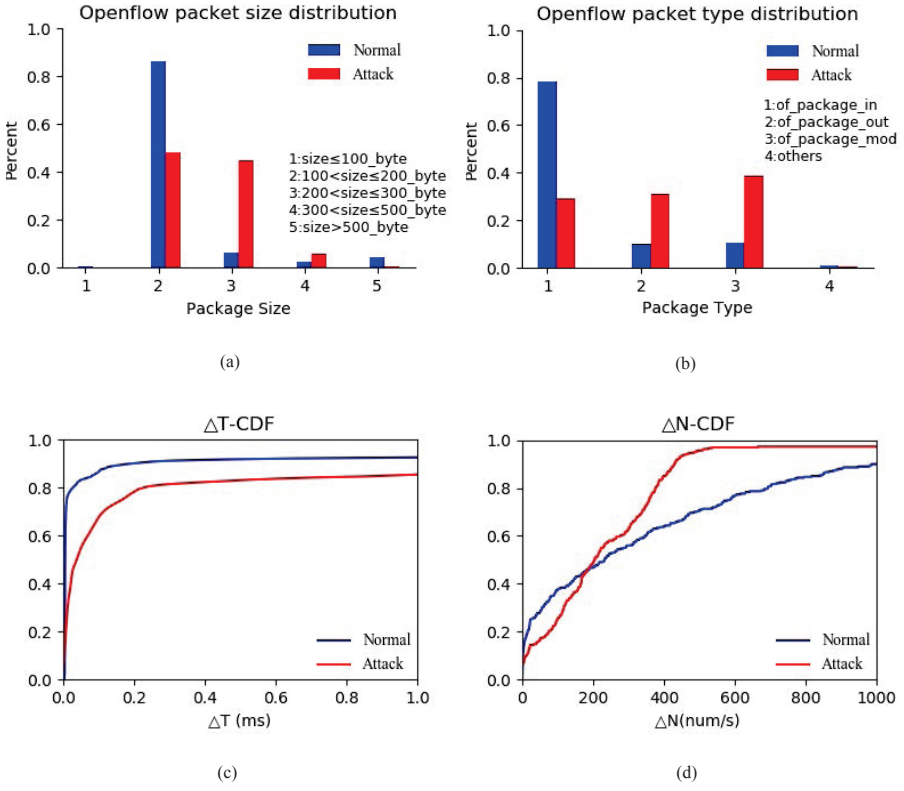


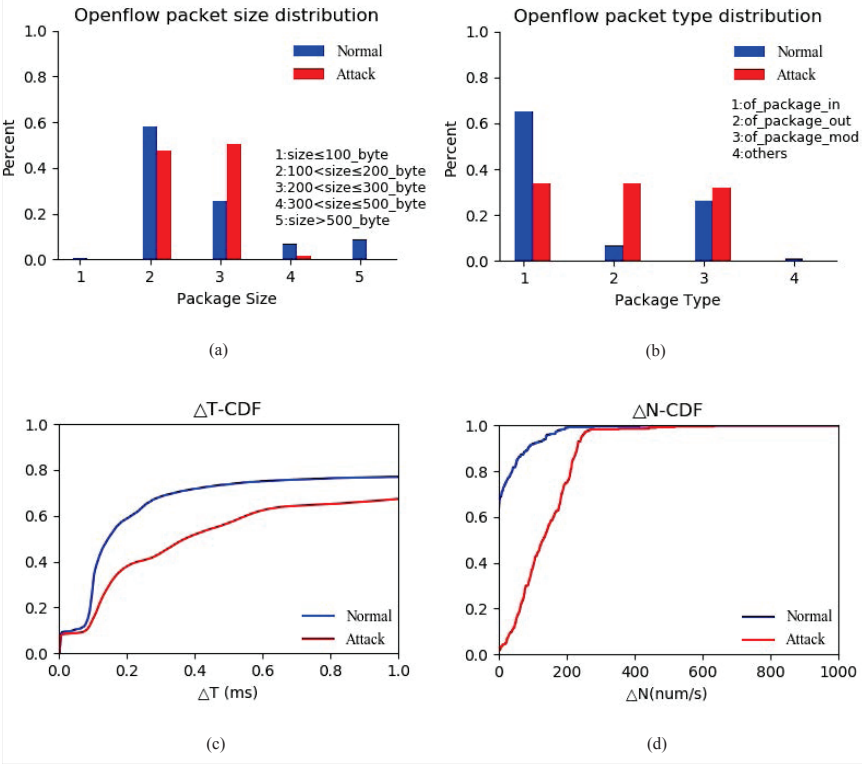
Figure 6 The distribution of feature data in the controller (see online version for colours)



29.3%. The same happens in the attack flow of the switch as, indicated by Figure 7(b).

Figures 6(c), (d) and 7(c), (d) indicate the comparison of features ΔT and ΔN for normal vs. attack traffic respectively. The DDoS attack streams affect not only the size distribution of Openflow message but also the type distribution of Openflow message. The ΔT is the time difference between the current arriving Openflow packets (all types) and the last arriving one. Compared with the two features mentioned above, the importance of the ΔT in the RF algorithm is not very high. On the contrary, the importance of feature $\frac{d\Delta T}{dt}$ is higher in the controller, which is related to the number of arriving Openflow packets and the correlation of different types of Openflow packets in the attack stream. In Figure 6(b) and 7(b), the most relevant Openflow package types to attack behaviour are OF_Package_In, OF_Package_Out and OF_Flow_Mod, the remaining types of Openflow packages are less relevant to attack flows. Therefore, If we replace any type of Openflow package interval, ΔT with a specific type (such as OF_Package_In package) package interval ΔT_{In}, and then both the importance of ΔT_{In} and the accuracy of the prediction will increase.

Figure 7 The distribution of feature data in the switch 4 (see online version for colours)



6 Conclusion

In this paper, we present a learning-based mechanism to detect the low-rate DDoS attack on SDN control and switch nodes under the IoT network environment. The proposed mechanism is based on two groups of features (stateless and stateful features) that extract from the Openflow package. The detection system uses learning algorithms to develop classifiers to distinguish normal and attack streams. We design and implement the experimental environment to evaluate our approach, which includes the low-rate DDoS attack module based on IoT devices, the physical and virtual heterogeneous SDN network, and the data flow capture and feature extraction model. Based on our platform, we verified the prediction results from different learning algorithms and the distribution of each features in the raw data. We also compared the results with traditional IP packet classification solution for the DDoS attack in IoT networks. The experiment results illustrate the effectiveness of our approach.

Acknowledgement

This work was partially supported by the Beijing Natural Science Foundation (No.4202036), the National Key R&D Program of China(No.2017YFB1400700), the National Natural Science Foundation of China(No. U11733115, No. 61972017, No. 61972018, No. 61932014, No. 61871023), and the Opening Project of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security (No.AGK2019001)

References

Ahmad, I., Namal, S., Ylianttila, M. and Gurtov, A. (2015) 'Security in software defined networks: a survey', *IEEE Communications Surveys and Tutorials*, Vol. 17, No. 4, pp.2317–2346.

Antikainen, M., Aura, T. and Särelä, M. (2014) 'Spook in your network: attacking an SDN with a compromised openflow switch', *Nordic Conference on Secure It Systems*, Vol. 8788, pp.229–244.

Apollo (n.d.) *Apache activemq (apollo)*, [online] https://activemq.apache.org/ (Accessed 3 March, 2019).

Behal, S. and Kumar, K. (2017) 'Detection of ddos attacks and flash events using information theory metrics-an empirical investigation', *Computer Communications*, Vol. 103, pp.18–28.

Cai, Z. and He, Z. (2019) 'Trading private range counting over big iot data', *In 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp.144–153.

Cai, Z. and Zheng, X. (2018) 'A private and efficient mechanism for data uploading in smart cyber-physical systems', *IEEE Transactions on Network Science and Engineering*, pp.1–1.

Cai, Z., He, Z., Guan, X. and Li, Y. (2018) 'Collective data-sanitization for preventing sensitive information inference attacks in social networks', *IEEE Transactions on Dependable and Secure Computing*, Vol. 15, No. 4, pp.577–590.

Cai, Z., Zheng, X. and Yu, J. (2019) 'A differential-private framework for urban traffic flows estimation via taxi companies', *IEEE Transactions on Industrial Informatics*, pp.1–1.

Cao, Y., Fei, S., Liu, Q., Huang, M., Hao, W. and You, I. (2017) 'A lddos-aware energy-efficient multipathing scheme for mobile cloud computing systems', *IEEE Access*, Vol. 5, No. 99, pp.21862–21872.

Chang, H.Y. (2017) 'A connectivity-increasing mechanism of zigbee-based IoT devices for wireless multimedia sensor neworks', *Multimedia Tools and Applications*, Vol. 78, pp.5137–5154.

Chen, J., Tang, X., Cheng, J., Wang, F. and Xu, R. (2019) 'DDOS attack detection method based on network abnormal behavior in big data environment', arXiv: Cryptography and Security, Retrieved 11 November, 2019, from http://arxiv.org/abs/1903.11844

Chistokhodova, A.A. and Sidorov, I.D. (2018) 'Novel method for low-rate DDOS attack detection', *Journal of Physics Conference Series*, p.032024.

Dayal, N., Maity, P., Srivastava, S. and Khondoker, R. (2016) 'Research trends in security and DDOS in SDN', *Security and Communication Networks*, Vol. 9, No. 18, pp.6386–6411.

Dhawan, M., Poddar, R., Mahajan, K. and Mann, V. (2015) 'Sphinx: Detecting security attacks in software-defined networks', *NDSS*, Vol. 15, pp.8–11.

Doshi, R., Apthorpe, N. and Feamster, N. (2018) 'Machine learning DDOS detection for consumer internet of things devices', *IEEE Symposium on Security and Privacy*, pp.29–35.

Floodlight (n.d.) *Floodlight SDN Controller*, [online] http://www.projectfloodlight.org/floodlight/ (Accessed 21 September, 2019).

Gao, Q., Huo, Y., Jing, T., Ma, L., Wen, Y. and Xing, X. (2019) 'An intermittent cooperative jamming strategy for securing energy-constrained networks', *IEEE Transactions on Communications*, Vol. 67, No. 11, pp.7715–7726.

Guirguis, M., Bestavros, A. and Matta, I. (2004) 'Exploiting the transients of adaptation for ROQ attacks on internet resources', *IEEE International Conference on Network Protocols*, Berlin, Germany, pp.184–195.

Hoque, N., Bhattacharyya, D.K. and Kalita, J.K. (2016) 'A novel measure for low-rate and high-rate ddos attack detection using multivariate data analysis', *2016 8th International Conference on Communication Systems and Networks, COMSNETS 2016*, Bangalore, India.

Huo, Y., Fan, X., Ma, L., Cheng, X., Tian, Z. and Chen, D. (2019) 'Secure communications in tiered 5g wireless networks with cooperative jamming', *IEEE Transactions on Wireless Communications*, Vol. 18, No. 6, pp.3265–3280.

Iii, A. F. H., Khanna, V., Tuncay, G., Want, R. and Kravets, R. (2016) 'Bluetooth low energy in dense iot environments', *IEEE Communications Magazine*, Vol. 54, No. 12, pp.30–36.

Jang, S.W., Kim, G.Y. and Byun, S. (2014) 'Clustering-based pattern abnormality detection in distributed sensor networks', *International Journal of Distributed Sensor Networks*, Vol. 2014, No. 3, pp.1–9.

Kolias, C., Kambourakis, G., Stavrou, A. and Voas, J. (2017) 'Ddos in the IoT: Mirai and other botnets', *Computer*, Vol. 50, No. 7, pp.80–84.

Kuerban, M., Tian, Y., Yang, Q., Jia, Y., Huebert, B. and Poss, D. (2016) Flowsec: Dos attack mitigation strategy on sdn controller', *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, IEEE, pp.1–2.

Kuzmanovic, A. and Knightly, E.W. (2006) 'Low-rate TCP-targeted denial of service attacks and counter strategies', *IEEE/ACM Transactions on Networking*, Vol. 14, No. 4, pp.683–696.

Luo, J., Yang, X., Jin, W., Jie, X., Jian, S. and Long, K. (2014) 'On a mathematical model for low-rate shrew DDOS', *IEEE Transactions on Information Forensics and Security*, Vol. 9, No. 7, pp.1069–1083.

Mahjabin, T., Xiao, Y., Li, T. and Chen, C. L. P. (2019) 'Load distributed and benign-bot mitigation methods for iot dns flood attacks', *IEEE Internet of Things Journal*, Vol. 7, No. 2, pp.986–1000.

Mahjabin, T., Xiao, Y., Sun, G. and Jiang, W. (2017) 'A survey of distributed denial-of-service attack, prevention, and mitigation techniques:', *International Journal of Distributed Sensor Networks*, Vol. 13, No. 12, pp.1–33.

Mehmood, A., Mukherjee, M., Ahmed, S. H., Song, H. and Malik, K. M. (2018) 'Nbc-maids: Naïve bayesian classification technique in multi-agent system-enriched ids for securing iot against ddos attacks', *Journal of Supercomputing*, Vol. 74, No. 10, pp.5156–5170.

Mohammadi, R., Conti, M., Lal, C. and Kulhari, S.C. (2019) 'SYN – Guard: an effective counter for SYN flooding attack in software-defined networking', *International Journal of Communication Systems*, Vol. 32, No. 18, Accessed 3 November, 2019, doi:10.1002/dac.4061.

Mqt (n.d.) *Mqtt Protocol*, [online] http://mqtt.org/ (Accessed 3 March, 2019).

Ping, D., Du, X., Zhang, H. and Tong, X. (2016) 'A detection method for a novel DDOS attack against sdn controllers by vast new low-traffic flows', *IEEE International Conference on Communications*, Kuala Lumpur, Malaysia.

Qiao, Y., Yu, R., Gong, Q. and Li, J. (2016) 'Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges', *IEEE Communications Surveys and Tutorials*, Vol. 18, No. 1, pp.1–1.

Schumny, H. (2004) 'Wireless personal area networks', *Computer Standards and Interfaces*, Vol. 26, No. 3, pp.157–158.

Sklearn (n.d.) *Scikit learn: Machine learning in Python*, [online] https://scikit-learn.org/stable/ (Accessed 3 March, 2019).

Spooner, J. and Zhu, S.Y. (2016) 'A review of solutions for sdn-exclusive security issues', *International Journal of Advanced Computer Sc ence and Applications (IJACSA)*, Vol. 7, No. 8, pp.113–122.

Sun, H., Lui, J.C.S. and Yau, D.K.Y. (2004) 'Defending against low-rate tcp attacks: dynamic detection and protection', *12th IEEE International Conference on Network Protocols*, Berlin, Germany, pp.196–205.

Toklu, S. and Simsek, M. (2018) 'Two-layer approach for mixed high-rate and low-rate distributed denial of service (DDOS) attack detection and filterin', *Arabian Journal for Science and Engineering*, Vol. 43, No. 12, pp.7923–7931.

Xin, Y., Yang, L., Wei, W., Li, W. and Xin, C. (2017) 'A novel interest flooding attacks detection and countermeasure scheme in ndn', *2016 IEEE Global Communications Conference, GLOBECOM 2016*, Washington, DC, USA.

Zhou, L., Liao, M., Yuan, C. and Zhang, H. (2017) 'Low-rate DDOS attack detection using expectation of packet size', *Security and Communication Networks*, Vol. 2017, pp.1–14.